Facultatea de Electronică şi Telecomunicații
Universitatea Politehnica Timişoara

Naforniță Corina

Algoritmi robuşti de marcare transparentă a imaginilor cu turbocoduri

Raport de cercetare din cadrul grantului CNCSIS, de tip TD, cod 47, numar 33385/29.06.04

Cuprins

1. Codorul si decodorul turbo 2
Introducere 2
Codorul turbo 2
Structura decodorului iterativ 4
2. Algoritmul MAP 5
Introducere şi elemente de matematică 5
Calculul valorilor $\alpha_k(s)$ prin recursivitate înainte (la ieşire) 8
Calculul recursiv al valorilor $\beta_k(s)$ 9
Calculul valorilor lui $\gamma_k(s',s)$ 9
Sumar al algoritmului MAP 10
3. Principiul decodării turbo iterative 11
Decodarea turbo. Preliminarii matematice 11
Decodarea turbo iterativă 12
4. Modificări ale algoritmului MAP 14
Introducere 14
Algoritmul Max-Log-MAP. Preliminarii matematice 15
Corectarea aproximărilor. Algoritmul Log-MAP 17
5. Algoritmul SOVA 18
Descrierea matematică a algoritmului SOVA 18
Implementarea algoritmului SOVA 21
6. Compararea algoritmilor decodoarelor componente 21
7. Utilizarea turbo codurilor în marcarea transparentă a imaginilor 23
Marcarea transparentă 25
Stabilirea structurii sistemului 26
Abordarea unei codări realizabile practic 27
BIBLIOGRAFIE 30

1. Codorul si decodorul turbo

Introducere

Codarea turbo a fost introdusă în 1993 de Berrou, Glavieux şi Thitimajasima [BG96], care raportau rezultate impresionante pentru coduri cu lungime mare a cadrelor. Din momentul apariției, turbocodarea a evoluat într-un ritm fără precedent, datorită eforturilor intense depuse de cercetătorii din domeniu. Ca urmare, turbo codurile au fost introduce şi în standarde, ca de exemplu standardul pentru comunicații mobile de generația a treia (3G). In sistemele video de difuzare, unde întârzierea asociată sistemului e mai puțin critică decât în sistemele interactive sensibile la întârzieri, câştigurile în performanță care pot fi atinse sunt şi mai impresionante.

In lucrarea lor, [BG96], autorii au folosit concatenarea paralelă a două coduri convoluționale recursive sistematice, RSC (Recursive Systematic Convolutional codes), cu un dispozitiv de aleatorizare (interleaver) plasat intre cele doua codoare. Rațiunea care a stat la baza folosirii codurilor RSC va fi expusă ulterior în lucrare. Pentru decodare, au folosit o structură iterativă având implementată o variantă modificată a algoritmului MAP (Maximum A Posteriori probability), propus de Bahl, Cocke, Jelinek şi Raviv [BCJR74]. Acest algoritm, clasic, conduce la o probabilitate minima a erorii pe bit, BER (Bit Error Rate). De la apariția turbocodurilor s-a depus un efort enorm în domeniu, pentru a reduce complexitatea decodorului, de către diverşi cercetatori, ca Robertson cu Villebrun şi Hoeher, Berrou cu Adde Angui şi Faudeuil, Bataille ş.a.[WH00, VY00]. S-a mai propus şi utilizarea turbocodurilor împreuna cu scheme de modulație care folosesc eficient banda de frecvențe. Lucrările lui Benedetto şi Montorsi [BM96a, BM96b] respectiv Perez ş.a.[PSC96] au facilitat înțelegerea cauzelor performanțelor excelente ale turbocodurilor. Hagenauer ş.a. [HOP96] a extins conceptul şi pentru codurile bloc concatenate. Jung şi Nasshan citați în [WH00] au analizat performanțele sistemelor codate în care se folosesc cadre scurte, ca de exemplu sistemele de transmisie a vocii. Ei au aplicat turbocodurile şi la sistemele CDMA (Code Division Multiple Acces), folosind detecția combinată cu diversitatea antenelor. Barbulescu şi Pietrobon s-au ocupat de proiectarea interleaverelor. In lucrarea sa, Sklar a descris decodorul iterativ şi componentele acestuia [SKL97].

Codorul turbo

Structura generală utilizată în turbo codoare este prezentată în Figura 1, [BG96]. Sunt utilizate două coduri componente pentru a coda biții de intrare, şi un interleaver plasat între cele două codoare.

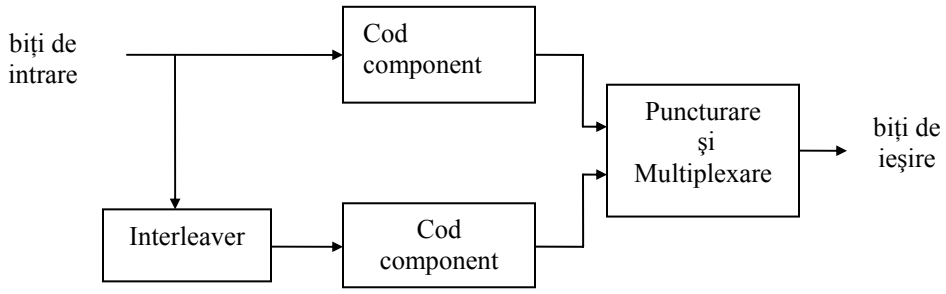

Fig. 1: Schema unui turbo codor

În general sunt utilizate codurile RSC ca şi coduri componente, dar este posibil de a se obţine performanţe bune, utilizând o structură ca cea din Figura 1, cu ajutorul altor coduri componente, cum ar fi codurile bloc. De asemenea, este posibil să se utilizeze mai mult de două coduri componente.

Ieşirile celor două codoare componente sunt mai apoi puncturate [THI93] şi multiplexate. De obicei ambele coduri componente RSC au rată 1/2, dând un bit de paritate şi un bit sistematic pentru fiecare bit de intrare.

Pentru a avea o rată de codare per total de jumătate, trebuiesc puncturaţi jumătate din biţii de ieşire ai fiecăruia dintre codoare. Astfel, se transmit toţi biţii sistematici ai primului codor RSC şi jumătate din biţii sistematici ai fiecărui codor. De precizat că biţii sistematici sunt foarte rar puncturaţi, deoarece aceasta degradează mult mai dramatic performanţa codului decât puncturând biţii de paritate. În Figura 2 s-a prezentat un cod RSC de rată 1/2 şi lungime de constrângere K=3, ce poate fi folosit ca şi cod component, în Figura 1.

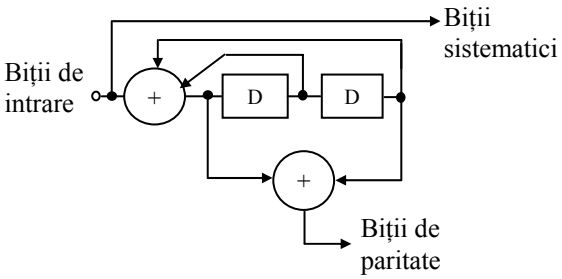

Fig. 2: Codor RSC (sistematic recursiv), K=3, R=1/2.

Codurile concatenate în paralel au fost investigate, la început, de Berrou ş.a. în articolul din 1993, [BGT93], dar îmbunătăţiri considerabile în performanţă ale turbo codurilor se datorează interleaverului utilizat între cele codoare şi a utilizării codurilor recursive ca şi coduri componente. Articole teoretice, publicate de Benedeto şi Montorsi [BM96a, BM96b] încearcă să explice remarcabilele performanţe ale turbo codurilor. Se pare că turbo-codurile pot avea un câştig al performanţei proporţional cu lungimea interleaver-ului utilizat. Însă complexitatea decodării per bit nu depinde de lungimea interleaverului. Aşadar pot fi obţinute perfomanţe foarte bune, cu o complexitate rezonabilă, prin utilizarea interleaverelor foarte lungi. Însă, în multe aplicaţii importante, precum

transmisiile vocale, lungimi cadru extrem de lungi nu sunt practice, datorită întârzierii rezultate.

Structura decodorului iterativ

Structura decodorului iterativ este prezentată în Figura 3.

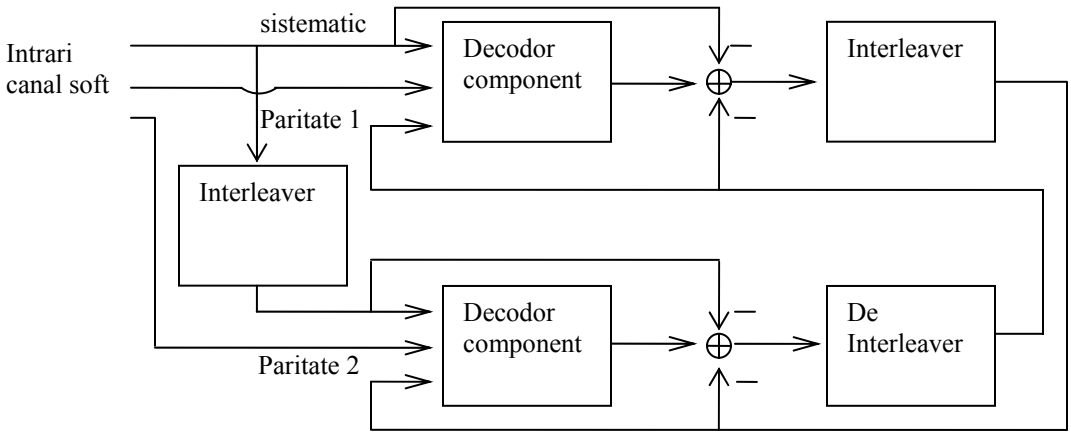

Fig.3: Schema turbodecodorului

Două decodoare componente sunt legate prin intermediul unor interleavere, într-o structură asemănătoare cu cea a codorului de la emisie. Fiecare codor primeşte trei intrări:
1) biţii codaţi sistematic, de la ieşirea canalului
2) biţii de paritate transmişi de la codorul component asociat şi
3) informaţia de la celălalt decodor component despre valorile plauzibile ale biţilor în cauză. Această informaţie de la celalalt decodor este numită informaţie *a priori.* Decodoarele componente trebuie să folosească atât intrările de la canal cât şi această informaţie a priori. Ele trebuie să ofere şi ceea ce se cunoaşte sub denumirea de ieşiri soft pentru biţii decodaţi. Aceasta înseamnă că, pe măsură ce livrează la ieşire secvenţa de biţi decodaţi, decodoarele componente trebuie să furnizeze, asociat fiecărui bit, şi probabilitatea ca acesta să fi fost decodat corect. Două decodoare adecvate sunt cele care folosesc algoritmul SOVA (Soft Output Viterbi Algorithm) [VAJ95] propus de Hagenauer şi Hoeher, respectiv algoritmul MAP propus de Bahl ş.a.

Ieşirile soft ale decodoarelor componente sunt de obicei reprezentate sub forma logaritmului raportului de plauzibilitate, LLR (Log Likelihood Ratio), al cărui semn reprezintă decizia hard asupra bitului [JH97] , iar mărimea sa reprezintă probabilitatea deciziei corecte sau decizia soft. Astfel, pentru o valoare u_k a bitului decodat, LLR este :

$$L(u_k) = \ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right) \qquad (1)$$

unde P(u_k = +1) este probabilitatea ca bitul u_k =+1 şi respectiv P(u_k = -1) probabilitatea ca u_k= -1. Pentru simplificarea deducerilor care urmeaza, cele doua valori posibile ale bitului u_k se iau =1 şi -1, şi nu 1 şi 0.

Decodorul din Figura 3 lucrează iterativ: în prima iteraţie primul decodor prelucrează doar valorile de ieşire ale canalului şi produce ieşirea soft ca estimat al biţilor de date. Ieşirea soft a primului decodor component e folosită apoi ca informaţie suplimentară pentru al doilea decodor component, care foloseşte această informaţie împreună cu ieşirile canalului, pentru a-şi calcula estimatul biţilor de date. Acum poate începe a doua iteraţie şi primul decodor component decodează din nou ieşirile canalului, dar împreună cu informaţia suplimentară referitoare la valoarea biţilor de intrare generată la ieşirea celui de al doilea decodor component la sfârşitul primei iteraţii. Aceasta informaţie suplimentară permite primului decodor să obţină un set mai precis de ieşiri soft, care apoi e folosit de al doilea decodor ca informaţie a priori. Acest ciclu se repetă şi, după fiecare iteraţie, BER al biţilor decodaţi descreşte. Totuşi se limitează la opt numărul de iteraţii, atât din cauza complexităţii cât şi datorită faptului că îmbunătăţirea performanţelor nu creşte liniar cu numărul de iteraţii.

Trebuie avută în vedere şi aleatorizarea folosită la codare. Deci e necesar să se ia măsuri pentru aleatorizarea şi de-aleatorizarea corectă a LLR-urilor care reprezintă valorile soft ale biţilor (figura 3). In plus, din cauza decodării iterative, trebuie luate măsuri ca o informaţie să fie folosită o singură dată în fiecare pas de decodare. Din acest motiv au fost introduse noţiunile de informaţie extrinsecă respectiv informaţie intrinsecă, în lucrarea lui Berrou, Glavieux şi Thitimajashima.

2. Algoritmul MAP

Introducere şi elemente de matematică

Bahl, Cocke, Jelinek şi Raviv au propus, în 1974 [BCJR74], un algoritm cunoscut sub numele de algoritmul MAP (Maximum A Posteriori probability) sau BCJR, pentru a determina probabilităţile *a posteriori* ale stărilor şi tranziţiilor unei surse Markov, afectată de un zgomot necorelat. Ei au arătat cum poate fi folosit algoritmul pentru decodarea atât a codurilor convoluţionale cat şi a codurilor bloc. Când e folosit pentru decodarea codurilor convoluţionale, algoritmul este optimal, în sensul minimizării BER la decodare, spre deosebire de algorimul Viterbi [VAJ95] care minimizează probabilitatea de cale incorectă prin trellis selectată de decodor. Astfel, algoritmul Viterbi poate fi privit ca minimizând numărul de *grupuri* de biţi asociaţi căii incorecte şi nu al numărului de biţi decodaţi incorect. De altfel, în majoritatea aplicaţiilor, performanţele celor doi algoritmi sunt aproape identice [BCJR74]. Algorimul MAP examinează fiecare din căile posibile prin trellisul decodorului convoluţional şi de aceea este extrem de complex. Astfel, complexitatea sa nu a justificat folosirea sa în majoritatea sistemelor. Situaţia s-a schimbat însă odată cu descoperirea turbocodurilor.

Algoritmul MAP generează nu numai secvenţa estimată de biţi ci şi probabilităţile ca fiecare bit să fi fost corect decodat. Acest lucru a fost esenţial pentru decodarea iterativă propusă de Berrou ş.a., astfel că în prima lor lucrare a fost folosit algoritmul MAP.

Ulterior au fost depuse mari eforturi pentru reducerea la un nivel rezonabil a complexității algoritmului MAP. In cele ce urmează, vor fi expuse bazele teoretice ale algoritmului MAP folosit pentru decodarea ieşirilor soft ale codurilor convoluționale ce compun turbocodul. Se presupune ca se folosesc coduri binare.

Algoritmul MAP generează pentru fiecare bit u_k decodat, probabilitatea ca acest bit să fi fost +1 sau -1, având dată secvența de simboluri recepționate $\underline{y}$. Acest lucru este echivalent cu găsirea lui $L\left(u_k \mid \underline{y}\right)$ sau LLR a posteriori, adică

$$L(u_k|\underline{y}) = \ln\left(\frac{P\left(u_k = +1|\underline{y}\right)}{P\left(u_k = -1|\underline{y}\right)}\right) \tag{2}$$

Dacă în trellis sunt cunoscute starea precedentă $S_{k-1} = s'$ şi starea prezentă $S_k = s$ atunci va fi cunoscut şi bitul de intrare u_k care determină tranziția între aceste stări. Acest lucru, împreună cu regula lui Bayes şi faptul că tranzițiile dintre starea precedentă S_{k-1} şi starea curentă S_k sunt mutual exclusive (în codor poate apărea doar una dintre ele), permite rescrierea ecuației (2) sub forma :

$$L(u_k|\underline{y}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}{\sum_{(s',s)\Rightarrow u_k=-1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}\right) \tag{3}$$

unde $(s',s) \Rightarrow u_k = +1$ este setul tranzițiilor din starea precedentă $S_{k-1} = s'$ în starea prezentă $S_k = s$ care poate apărea daca bitul de intrare $u_k = +1$ şi la fel $(s',s) \Rightarrow u_k = -1$. Pentru simplificare se rescrie $P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})$ ca $P(s' \wedge s \wedge \underline{y})$.

Vom considera acum probabilitățile individuale $P(s' \wedge s \wedge \underline{y})$ de la numărătorul şi numitorul ecuației (3). Secvența recepționată $\underline{y}$ poate fi separată în trei părți: cuvântul de cod recepționat asociat tranziției $\underline{y}_k$ prezente, secvența recepționată anterior tranziției prezente $\underline{y}_{j<k}$ şi secvența recepționată ulterior tranziției prezente $\underline{y}_{j>k}$. Putem astfel scrie astfel probabilitatile individuale $P(s' \wedge s \wedge \underline{y})$:

$$P(s' \wedge s \wedge \underline{y}) = P(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k \wedge \underline{y}_{j>k}) \tag{4}$$

In figura 4 este prezentată diagrama trellis pentru un cod RSC, cu 4 stări şi lungimea de constrângere K=3. Liniile continue reprezintă tranzițiile pentru bitul de intrare -1, iar liniile întrerupte reprezintă tranzițiile pentru u_k= +1. Valorile $\alpha_{k-1}(s')$, $\gamma_k(s',s)$ şi $\beta_k(s)$ sunt calculate de algoritmul MAP.

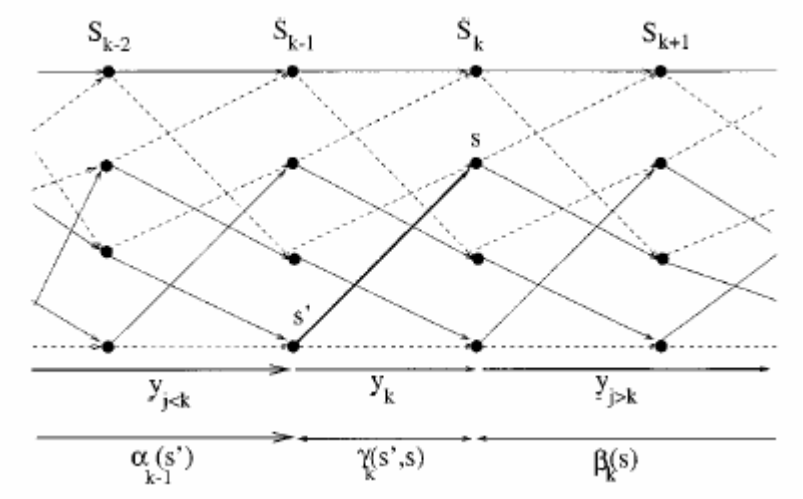

Fig. 4: Diagrama trellis MAP pentru un cod RSC cu K=3

Folosim relația $P(a \wedge b) = P(a \mid b) P(b)$ consecință a regulii lui Bayes şi considerăm un canal fără memorie. Rezultă că viitoarea secvență recepționata $\underline{y}_{j>k}$ va depinde doar de starea prezentă *s* şi nu de starea anterioară *s'* sau de secvența recepționată anterior $\underline{y}_{j<k}$ respectiv secvența prezentă $\underline{y}_k$, deci putem scrie :

$$P(s' \wedge s \wedge \underline{y}) = P(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k \wedge \underline{y}_{j>k}) = P(\underline{y}_{j>k} \mid s) P(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k)$$
$$= P(\underline{y}_{j>k} \mid s) P(\{\underline{y}_k \wedge s\} \mid s') P(s' \wedge \underline{y}_{j<k}) = \beta_k(s) \gamma_k(s', s) \alpha_{k-1}(s') \quad (5)$$

unde:

$$\alpha_{k-1}(s') = P(S_{k-1} = s' \wedge \underline{y}_{j<k}) \quad (6)$$

este probabilitatea ca trellisul să fie în starea *s'* la momentul *k-1* şi secvența recepționată din canal până în acest punct este $\underline{y}_{j<k}$

$$\beta_k(s) = P(\underline{y}_{j>k} \mid S_k = s) \quad (7)$$

este probabilitatea ca, dacă la momentul *k* trellisul se află în starea *s*, viitoarea secvență recepționață din canal să fie $\underline{y}_{j>k}$, şi în final,

$$\gamma_k(s', s) = P(\{\underline{y}_k \wedge S_k = s\} \mid S_{k-1} = s') \quad (8)$$

este probabilitatea ca, dacă trellisul se află în starea *s'* la momentul *k-1*, el să treacă în starea *s* şi secvența receptionată din canal pentru această tranziție să fie $\underline{y}_k$.

Ecuația (5) arată că probabilitatea $P\left(s'\wedge s\wedge \underline{y}\right)$ de trecere a codorului din starea $S_{k-1}=s'$ în starea $S_k=s$ şi de recepționare a secvenței $\underline{y}$, poate fi împărțită într-un produs de factori, $\alpha_{k-1}\left(s'\right)$, $\gamma_k\left(s',s\right)$ şi $\beta_k\left(s\right)$. Semnificația acestor trei factori este dată în figura 2, unde e marcată cu linie îngroşată tranziția din $S_{k-1}=s'$ în $S_k=s$. Din ecuațiile (3) şi (5) putem scrie pentru LLR condiționată a lui u_k, fiind dată secvența $\underline{y}_k$:

$$L(u_k|\underline{y})=\ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1}P(S_{k-1}=s'\wedge S_k=s\wedge\underline{y})}{\sum_{(s',s)\Rightarrow u_k=-1}P(S_{k-1}=s'\wedge S_k=s\wedge\underline{y})}\right)$$
$$=\ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1}\alpha_{k-1}\left(s'\right)\gamma_k\left(s',s\right)\beta_k\left(s\right)}{\sum_{(s',s)\Rightarrow u_k=-1}\alpha_{k-1}\left(s'\right)\gamma_k\left(s',s\right)\beta_k\left(s\right)}\right) \qquad (9)$$

Algoritmul MAP găseşte $\alpha_k\left(s\right)$ şi $\beta_k\left(s\right)$ pentru toate stările *s* prin trellis, de exemplu pentru *k=0,1,...,N-1* şi $\gamma_k\left(s',s\right)$ pentru toate tranzițiile posibile din starea $S_{k-1}=s'$ în starea S_k=*s* , din nou pentru *k=0,1,...,N-1* . Aceste valori sunt apoi folosite cu ecuația (9), pentru a obține LLR-urile condiționate $L\left(u_k\mid\underline{y}\right)$ pe care decodorul MAP le furnizează. In cele ce urmează e descris modul de calcul al valorilor $\alpha_k\left(s\right)$, $\beta_k\left(s\right)$ şi $\gamma_k\left(s',s\right)$.

Calculul valorilor $\alpha_k(s)$ prin recursivitate înainte (la ieşire)

Din definiția lui $\alpha_{k-1}\left(s'\right)$, (ecuația 6), putem scrie :

$$\alpha_k\left(s\right)=P\left(S_k=s\wedge\underline{y}_{j<k+1}\right)=P\left(s\wedge\underline{y}_{j<k}\wedge\underline{y}_k\right)$$
$$=\sum_{toti\_s'}P\left(s\wedge s'\wedge\underline{y}_{j<k}\wedge\underline{y}_k\right)=\sum_{toti\_s'}P(s\wedge s'\wedge\underline{y}_{j<k}\wedge\underline{y}_k) \qquad (10)$$

unde, în ultima linie am împărțit probabilitatea $P\left(s\wedge\underline{y}_{j<k+1}\right)$ într-o sumă de probabilități mutuale $P\left(s\wedge s'\wedge\underline{y}_{j<k+1}\right)$ pe toate stările posibile precedente s'. Folosind regula lui Bayes şi ipoteza că avem din nou un canal fără memorie, putem proceda astfel:

$$\alpha_k\left(s\right)=\sum_{toti\_s'}P(s\wedge s'\wedge\underline{y}_{j<k}\wedge\underline{y}_k)$$
$$=\sum_{toti\_s'}P\left(\left\{s\wedge\underline{y}_k\right\}\middle|\left\{s'\wedge\underline{y}_{j<k}\right\}\right)P\left(s'\wedge\underline{y}_{j<k}\right)$$
$$=\sum_{toti\_s'}P\left(\left\{s\wedge\underline{y}_k\right\}\middle|s'\right)P\left(s'\wedge\underline{y}_{j<k}\right)$$

$$= \sum_{toti\_s'} \alpha_{k-1}(s')\gamma_k(s',s) \tag{11}$$

Astfel, odată ce valorile $\gamma_k(s',s)$ sunt cunoscute, valorile $\alpha_k(s)$ pot fi calculate recursiv. Presupunând că trellisul are starea inițială $S_0 = 0$, condițiile inițiale pentru calculul recursiv sunt:

$$\alpha_0(S_0=0) = 1$$
$$\alpha_0(S_0=s) = 0 \text{ pentru toți } s \neq 0 \tag{12}$$

Calculul recursiv al valorilor $\beta_k(s)$

Valorile lui $\beta_k(s)$ pot fi calculate recursiv, în mod asemănător. Folosind o deducere similară cu cea a ecuației (11), se poate arăta că

$$\beta_{k-1}(s') = P\left(\underline{y}_{j>k-1} \middle| S_k = s'\right) = \sum_{toti\_s} \beta_k(s)\gamma_k(s',s) \tag{13}$$

Astfel că, odată ce sunt cunoscute valorile $\gamma_k(s',s)$ poate fi folosită o recursivitate înapoi / la intrare pentru a calcula valorile lui $\beta_{k-1}(s')$ din valorile lui $\beta_k(s)$ folosind relația (13).

Calculul valorilor lui $\gamma_k(s',s)$

Vom vedea acum cum pot fi calculate valorile $\gamma_k(s',s)$ ale probabilității tranziției, din relația (5), din secvența recepționată din canal şi orice informație disponibilă a priori. Folosind definiția lui $\gamma_k(s',s)$ (relația 8) şi regula lui Bayes, avem:

$$\begin{aligned}\gamma_k(s',s) &= P\left(\left\{\underline{y}_k \wedge s\right\} \middle| s'\right) = P\left(\underline{y}_k \middle| \left\{s' \wedge s\right\}\right) P\left(s \middle| s'\right) \\ &= P\left(\underline{y}_k \middle| \left\{s' \wedge s\right\}\right) P(u_k) = P\left(\underline{y}_k \middle| \underline{x}_k\right) P(u_k)\end{aligned} \tag{14}$$

unde:

u_k este bitul de intrare necesar pentru a determina tranziția din starea $S_{k-1} = s'$ în starea $S_k = s$

$P(u_k)$ este probabilitatea a priori a acestui bit

$\underline{x}_k$ este cuvântul de cod transmis asociat acestei tranziții.

Astfel, probabilitatea tranziției $\gamma_k(s',s)$ e dată de produsul dintre probabilitatea a priori a bitului de intrare u_k necesar pentru această tranziție şi probabilitatea ca având dat

cuvântul de cod $\underline{x}_k$ care s-a transmis, asociat acestei tranziții, să recepționăm secvența de canal $\underline{y}_k$. Probabilitatea a priori $P(u_k)$ e dedusă într-un decodor iterativ din ieşirea decodorului component precedent, şi probabilitatea condiționată a secvenței recepționate $P(\underline{y}_k \mid \underline{x}_k)$ e dată, în ipoteza canalului gaussian fără memorie cu modulație BPSK, ca:

$$P\left(\underline{y}_k \middle| \underline{x}_k\right) = \prod_{l=1}^{n} P\left(y_{kl} \middle| x_{kl}\right) = \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{E_b R}{2\sigma^2}\left(y_{kl} - a x_{kl}\right)^2\right)} \tag{15}$$

unde:

x_{kl} *şi* y_{kl} sunt biții individuali, din cuvintele de cod $\underline{x}_k$ transmis şi $\underline{y}_k$ recepționat

n este numărul biților din fiecare cuvânt de cod
E_b este energia per bit transmisă
σ^2 este dispersia zgomotului (varianța)
a este amplitudinea fadingului; pentru canale fără fading, de tip AWGN a=1

Sumar al algoritmului MAP

Din cele prezentate se vede că decodarea MAP a secvenței recepționate $\underline{y}$ pentru a obține LLR a posteriori, $L(u_k \mid \underline{y})$ poate fi rezumată după cum urmează. Pe măsura recepționării valorilor y_{kl}, ele sunt folosite împreună cu LLR-urile a priori $L(u_k)$ (care într-un decodor turbo iterativ sunt furnizate de celălalt decodor component) pentru a calcula $\gamma_k(s',s)$ conform relațiilor (14) şi (15). Pe măsura recepționării din canal a valorilor y_{kl} şi calculării valorilor $\gamma_k(s',s)$, se poate folosi recursivitatea înainte (relația 11), pentru a calcula $\alpha_k(s',s)$. Odată ce toate valorile din canal au fost recepționate şi au fost calculate $\gamma_k(s',s)$ pentru toți $k=1,2,...,N$, poate fi folosită recursivitatea înapoi (relația 13) pentru a calcula valorile $\beta_k(s',s)$. In final, toate valorile calculate pentru $\alpha_k(s',s)$, $\beta_k(s',s)$ şi $\gamma_k(s',s)$ sunt folosite în (9) pentru a calcula $L(u_k \mid \underline{y})$. Trebuie luate măsuri pentru a evita problemele de depăşire negativă în calculul lui $\alpha_k(s',s)$ şi $\beta_k(s',s)$, dar asemenea probleme pot fi evitate printr-o normalizare adecvată a acestor valori. Aceste normalizări dispar, din cauza raportului existent în relația (9), astfel încât nu vor modifica LLR-urile furnizate de algoritm.

Algoritmul MAP este extrem de complex în forma descrisă, din cauza înmulțirilor necesare în relațiile (11) şi (13), pentru calculul recursiv al valorilor $\alpha_k(s',s)$ şi $\beta_k(s',s)$, a înmulțirii şi exponențialelor necesare la calculul lui $\gamma_k(s',s)$ folosind relația (15), precum şi a înmulțirilor şi logaritmului natural necesare pentru calculul lui $L(u_k \mid \underline{y})$, folosind relația (9). De aceea, s-a depus un efort continuu pentru reducerea

complexității algoritmului, şi s-a ajuns la algoritmul Log-MAP propus de Robertson, ş.a. având aceleaşi performanțe ca şi algoritmul MAP, dar cu o complexitate semnificativ mai mică şi fără problemele de scalare menționate.

3. Principiul decodării turbo iterative

Decodarea turbo. Preliminarii matematice

In cele ce urmează vor fi descrise conceptele de informație extrinsecă şi intrinsecă folosite de Berrou ş.a. [BGT93] şi modul în care algoritmul MAP respectiv alte decodoare cu intrare soft şi ieşire soft, SISO (Soft-In-Soft-Out) se pot utiliza în decodarea iterativă a turbo codurilor.
In [BGT93] se arată că, pentru un cod sistematic cum e codul RSC, ieşirea decodorului MAP, dată de relația (9), poate fi rescrisă ca:

$$L(u_k \mid \underline{y}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\Rightarrow u_k=-1} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right) = L(u_k) + L_c y_{ks} + L_e(u_k) \qquad (16)$$

unde :

$$L_e(u_k) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} \alpha_{k-1}(s')\chi_k(s',s)\beta_k(s)}{\sum_{(s',s)\Rightarrow u_k=-1} \alpha_{k-1}(s')\chi_k(s',s)\beta_k(s)}\right) \qquad (17)$$

Aici, $L(u_k)$ este LLR a priori dat de ecuația (1), iar L_c e numit măsura gradului de încredere sau a siguranței canalului, şi se calculează cu relația

$$L_c = \frac{4a}{2\sigma^2} \qquad (18)$$

y_{ks} este versiunea recepționată a bitului sistematic transmis $x_{ks}=u_k$, iar

$$\chi_k(s',s) = \exp\left(\frac{L_c}{2}\sum_{l=2}^{n} y_{kl}x_{kl}\right) \qquad (19)$$

Astfel putem vedea că LLR a posteriori $L(u_k \mid \underline{y})$ calculat cu algoritmul MAP poate fi văzut ca format din trei termeni, $L(u_k)$, $L_c y_{ks}$ şi $L_e(u_k)$. Termenul $L(u_k)$ din LLR a priori derivă din $P(u_k)$ din expresia (14) pentru probabilitatea tranziției ramurii $\gamma_k(s',s)$. Aceasta probabilitate ar putea proveni dintr-o sursă independentă. In majoritatea

cazurilor nu vom avea cunoştinţe independente sau a priori despre valoarea cea mai plauzibilă a bitului u_k, şi astfel $L(u_k)$ =0 (LLR a priori), corespunzând unei probabilităţi a priori $P(u_k)$=0,5. Totuşi, în cazul unui decodor turbo iterativ, fiecare decodor component poate furniza celuilalt decodor un estimat a priori a LLR, $L(u_k)$, aşa după cum se va vedea în continuare [WH00].

Al doilea termen, $L_c\, y_{ks}$ din relaţia (16) este ieşirea soft a canalului pentru bitul sistematic u_k, care a fost transmis direct în canal şi recepţionat ca y_{ks}. Când raportul semnal pe zgomot, SNR (Signal to Noise Ratio) al canalului este mare, valoarea gradului de siguranţă al canalului în ecuaţia (18) va fi mare şi acest bit sistematic va avea o influenţă mare asupra LLR a posteriori $L(u_k|\underline{y})$. Când, dimpotrivă, canalul este de calitate scăzută şi L_c este mic, ieşirea soft a canalului pentru bitul sistematic recepţionat y_{ks} va avea o influenţă mai redusă asupra LLR a posteriori generat de algoritmul MAP.

Ultimul termen din ecuaţia (16), $L_e(u_k)$ este dedus folosind condiţiile impuse de codul folosit, din secvenţa $L(u_n)$ de informaţie a priori şi secvenţa $\underline{y}$ de informaţie recepţionată din canal, *excluzând* biţii sistematici y_{ks} recepţionaţi şi informaţia a priori $L(u_k)$ pentru bitul u_k. De aceea, el este denumit LLR *extrinsec* pentru bitul u_k. Ecuaţia (16) arată că informaţia extrinsecă de la decodorul MAP poate fi obţinută scăzând informaţia a priori $L(u_k)$ şi intrarea sistematică recepţionată a canalului $L_c\, y_{ks}$ din ieşirea soft $L(u_k|\underline{y})$ a decodorului. Acesta este motivul pentru scăderea din Figura 3. Pentru celălalt decodor component folosit în decodarea turbo, se pot deduce ecuaţii similare cu (16).

Acum pot fi expuse pe scurt noţiunile de *informaţie a priori, a posteriori şi extrinsecă*, care stau la baza decodării iterative a turbo codurilor.

a priori informaţia a priori despre un bit este informaţia cunoscută înainte de a începe decodarea, de la o altă sursă decât secvenţa recepţionată sau constrângeri ale codului. Ea este uneori numită şi informaţie intrinsecă, în contrast cu informaţia extrinsecă, descrisa mai jos.

extrinsec informaţia extrinsecă despre un bit u_k este informaţia furnizată de decodor pe baza secvenţei recepţionate şi a informaţiei a priori *excluzând* bitul sistematic y_{ks} recepţionat şi informaţia a priori $L(u_k)$ pentru bitul u_k. Tipic, această informaţie e furnizată de decodorul component folosind constrângerile impuse secvenţei transmise de codul utilizat. El procesează biţii recepţionaţi şi informaţia a priori referitoare la bitul sistematic u_k şi foloseşte aceste informaţii şi constrângerile codului pentru a furniza informaţii despre valoarea lui u_k.

a posteriori informaţia a posteriori despre un bit e informaţia luată în considerare de către decodor de la *toate* sursele de informaţie, referitoare la u_k. Este LLR a posteriori şi anume $L(u_k|\underline{y})$ generat de algoritmul MAP la ieşire.

Decodarea turbo iterativă

Considerăm la început primul decodor component, la prima iteraţie. Acesta primeşte secvenţa din canal $L_c\underline{y}^{(1)}$ conţinând versiunea recepţionată a biţilor sistematici transmişi $L_c\, y_{ks}$ şi a biţilor de paritate $L_c\, y_{kl}$ de la primul codor. De obicei, pentru a reduce la jumătate rata codului, jumătate din aceşti biţi de paritate vor fi eliminaţi la transmiţător şi

astfel decodorul turbo va trebui să insereze zerouri în ieşirea soft a canalului $L_c\, y_{kl}$ pentru aceşti biţi strecurați. Primul decodor component poate apoi procesa intrările soft de la canal şi să producă estimatul său $L_{11}\left(u_k \left| \underline{y} \right.\right)$ a LLR-urilor condiționate a biților de date u_k, k=1,2,...,N. Cu notația folosită, indicele 11 din $L_{11}\left(u_k \left| \underline{y} \right.\right)$ indică că acesta este LLR a posteriori din prima iterație a primului decodor component. De remarcat este faptul ca în aceasta prima iterație, primul decodor component nu are informație a priori despre bit şi astfel $P\left(u_k\right)$ din (16) care da $\gamma_k\left(s', s\right)$ va fi 0,5.

Apoi al doilea decodor component începe să lucreze. El recepționează secvența de canal $\underline{y}^{(2)}$ conținând versiunea aleatorizată a biților sistematici recepționați şi biții de paritate de la al doilea codor. Turbo decodorul va trebui să insereze zerouri din nou în aceasta secvență, dacă biții de paritate generați de codor au fost strecurați înaintea transmisiei. Acum însă, pe lângă $\underline{y}^{(2)}$ secvența de canal recepționată, decodorul poate utiliza LLR condiționată $L_{11}\left(u_k \left| \underline{y} \right.\right)$ furnizată de primul decodor component pentru a genera LLR-urile a priori $L(u_k)$ care vor fi utilizate de al doilea decodor component. Metaforic vorbind acest LLR a priori $L(u_k)$, care se referă la bitul u_k, este dat de o „comportare independentă a informației, cu scopul de a avea două opinii independente despre calitatea canalului" referitoare la bitul u_k. Acesta va furniza o „a doua opinie despre calitatea canalului" în ceea ce priveşte bitul u_k. Intr-un decodor turbo iterativ, informația extrinsecă $L_e(u_k)$ de la celălalt decodor component e folosită ca LLR-uri a priori, după ce a fost aleatorizată pentru a rearanja biții decodați $\underline{u}$ în aceeaşi ordine în care au fost codați de al doilea codor. Al doilea decodor component foloseşte astfel secvența de canal recepționată $\underline{y}^{(2)}$ şi LLR-urile a priori $L(u_k)$ (rezultate prin aleatorizarea LLR-urilor extrinseci $L_e(u_k)$ ale primului decodor component) pentru a produce LLR-ul său a posteriori $L_{12}\left(u_k \left| \underline{y} \right.\right)$. Acesta este sfârşitul primei iterații.

Pentru a doua iterație primul decodor component procesează din nou secvența sa recepționată din canal $\underline{y}^{(1)}$, dar acum el are şi LLR-urile a priori $L(u_k)$ date de partea extrinsecă $L_e(u_k)$ a LLR-urilor a posteriori $L_{12}\left(u_k \left| \underline{y} \right.\right)$ calculate de al doilea decodor component, şi astfel el poate produce un LLR a posteriori îmbunătățit $L_{21}\left(u_k \left| \underline{y} \right.\right)$. A doua iterație continuă apoi cu al doilea decodor component folosind LLR-urile a posteriori îmbunătățite $L_{21}\left(u_k \left| \underline{y} \right.\right)$ de la primul codor, pentru a deduce cu relația (16) un LLR a priori îmbunătățit $L(u_k)$ pe care îl foloseşte împreună cu secvența de canal recepționată $\underline{y}^{(2)}$ pentru a calcula $L_{22}\left(u_k \left| \underline{y} \right.\right)$.

Acest proces iterativ continuă şi cu fiecare iterație, BER mediu a biților decodați scade. Dar, după un anumit număr de iterații [SKL97] performanța nu se mai îmbunătăţeşte la fiecare iterație, odată cu creşterea numărului de iterații. Astfel, deoarece nu mai apare o îmbunătățire de performanță semnificativă, după un anumit număr de iterații şi din motive de complexitate, se limitează numărul de iterații la opt.

La decodarea iterativă a turbo codurilor, informația a priori folosită de decodorul component în contextul bitului u_k, ar trebuie să fie, în mod ideal, bazată pe o „conduită a informației", independentă de ieşirile canalului folosite de acel decodor. Mai explicit, bitul de informație sistematic original şi biții de paritate aferenți sunt afectați diferit de canal, din cauza constrângerilor impuse de cod. Astfel, chiar dacă bitul de informație sistematic a fost foarte corupt de canal, biții de paritate aferenți pot ajuta decodorul să obțină un estimat, cu un mare grad de siguranță, asupra bitului afectat de canal. Totuşi, după cum s-a explicat anterior, în decodoarele turbo $L_e(u_k)$ pentru bitul u_k folosesc toți biții de paritate recepționați disponibili şi toți biții sistematici recepționați, cu excepția valorii y_{ks} recepționată, asociată bitului u_k. Aceeaşi biți sistematici recepționați sunt folosiți şi de celelalte părți ale decodorului, care folosesc versiuni aleatorizate sau de-aleatorizate ale LLR-urilor a priori $L_e(u_k)$. Astfel, LLR-urile a priori $L(u_k)$ nu sunt cu adevărat independente față de $\underline{y}$, ieşirile canalului folosite de decodoarele componente. Totuşi, deoarece codurile convoluționale au o memorie redusă, de obicei doar patru biți sau mai puțini, LLR extrinsec $L_e(u_k)$ este singurul afectat semnificativ de biții sistematici recepționați relativ apropiați de bitul u_k . Când LLR extrinsec $L_e(u_k)$ e folosit ca LLR a priori $L(u_k)$ de celălalt decodor component, din cauza aleatorizării folosite, bitul u_k şi vecinii săi vor fi probabil separați mult. Astfel că dependența LLR-urilor a priori $L(u_k)$ față de valorile sistematice de canal recepționate folosite de celalalt decodor component, va avea un efect relativ redus şi decodarea iterativă dă rezultate bune.

Un alt motiv pentru utilizarea decodării iterative este dat de cât de bine lucrează: decodarea optimală a turbocodurilor conduce la o performanță cu doar câteva fracțiuni de dB, (0,35-0,5) dB mai bună decât performanța obținută cu decodarea iterativă folosind algoritmul MAP [WH00]. Apoi au găsite diferite scheme de codare care permit apropierea de limita lui Shannon, la câteva fracțiuni de dB, limita care ne indică cea mai buna performanță obtenabilă teoretic. Astfel că decodarea iterativă a turbo codurilor se pare că duce la performanță optimă.

După ce a fost descrisă folosirea algoritmului MAP la decodarea iterativă a codurilor turbo, vom descrie şi alte decodoare cu intrare soft şi ieşire soft, mai puțin complexe, care pot fi folosite în locul algoritmului MAP, şi anume Max-Log-MAP, Log-MAP şi SOVA.

4. Modificări ale algoritmului MAP

Introducere

Algoritmul MAP este mult mai complex decât algoritmul Viterbi, iar pentru decizii hard cei doi algoritmi dau performanțe aproape identice. De aceea nu a fost folosit timp de aproape două decenii. Interesul față de el a reapărut odată cu apariția turbo codurilor şi s-a realizat ca i se poate reduce foarte mult complexitatea fără a-i afecta performanța. Inițial a fost propus algoritmul Max-Log-MAP de către Koch şi Baier, respectiv Erfanian ş.a. [WH00]. Această tehnică simplifică algoritmul MAP prin translatarea recursivității în domeniul logaritmic şi folosirea unei aproximări. Datorită acestei aproximări performanța algoritmului este suboptimală față de cea a algoritmului MAP. Astfel că în 1995 a apărut o nouă propunere, a lui Robertson ş.a., [VY00], şi anume algoritmul Log-MAP, care

corectează aproximarea menționată, păstrând însă performanțele algoritmului MAP, şi scăzând dramatic complexitatea față de acesta.

Algoritmul Max-Log-MAP. Preliminarii matematice

Algoritmul MAP calculează LLR-urile a posteriori $L\left(u_k \left| \underline{y} \right.\right)$ folosind relația (9), lucru ce necesită următoarele valori:

1- valorile $\alpha_{k-1}(s')$, care se calculează printr-o recursivitate înainte (relația 11),
2- valorile $\beta_k(s)$, care se calculează printr-o recursivitate înapoi (relația 13)
3- probabilitățile $\gamma_k(s',s)$ de tranziție a ramurii, care se calculează cu relația (14).

Algoritmul Max-Log-MAP simplifică aceste operații prin transferarea în domeniul logaritmic şi folosirea apoi a aproximării:

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i (x_i) \tag{20}$$

unde $\max_i (x_i)$ înseamnă valoarea maximă a lui x_i. Se definesc apoi $A_k(s)$, $B_k(s)$ şi $\Gamma_k(s',s)$ astfel:

$$A_k(s) \triangleq \ln\left(\alpha_k(s)\right) \tag{21}$$

$$B_k(s) \triangleq \ln\left(\beta_k(s)\right) \tag{22}$$

$$\Gamma_k(s',s) \triangleq \ln \gamma_k(s',s)) \triangleq \ln(\gamma_k(s',s)) \tag{23}$$

Cu aceste notații, relația (11) devine :

$$\begin{aligned} A_k(s) &\triangleq \ln\left(\alpha_k(s)\right) \\ &= \ln\left(\sum_{toti\_s'} \alpha_{k-1}(s')\gamma_k(s',s)\right) \\ &= \ln\left(\sum_{toti\_s'} \exp\left[A_{k-1}(s') + \Gamma_k(s',s)\right]\right) \\ &\approx \max_s \left(A_{k-1}(s') + \Gamma_k(s',s)\right) \end{aligned} \tag{24}$$

Conform ecuației (24), pentru fiecare cale dintre starea precedentă şi starea prezentă $S_k = s$, algoritmul adună metricile de ramură $\Gamma_k(s',s)$ la valoarea precedentă $A_{k-1}(s')$, pentru a găsi, pentru acea cale, noua valoare $\tilde{A}_k(s)$. Conform cu relația (24), noua valoare $A_k(s)$ este maximum valorilor $\tilde{A}_k(s)$ a diverselor căi care ajung în starea $S_k = s$. Acest lucru poate fi văzut că selectarea unei căi ca „supraviețuitor" şi renunțarea

la celelalte căi care ajung la starea respectivă. Valoarea lui $A_k(s)$ ar trebui să dea logaritmul natural al probabilității ca trellisul să se afle în starea $S_k = s$ în etapa *k*, având dată secvența de canal recepționată până la acest punct $\underline{y}_{j<k}$. Totuşi, din cauza aproximării (relația 20) folosite pentru a deduce relația (24), se ia în considerare doar calea de plauzibilitate maximă ML (Maximum Likelihood) spre $S_k = s$, când se calculează această plauzibilitate. Astfel că valoarea lui A_k din algoritmul Max-Log-MAP dă de fapt probabilitatea celei mai plauzibile căi prin trellis spre starea $S_k = s$, nu probabilitatea oricărei cai prin trellis spre starea $S_k = s$. Această aproximare este una din cauzele performanței suboptimale a algoritmului Max-Log-MAP față de cea a algoritmului MAP.

Din relația (24) se vede ca în algoritmul Max-Log-MAP, recursivitatea înainte folosită pentru calculul lui $A_k(s)$ este aceeaşi ca cea din algoritmul Viterbi – pentru fiecare pereche de căi care ajung la aceeaşi stare, găsirea supraviețuitorului implică două adunări şi o comparație. De remarcat este ca pentru trellisul binar, sumarea şi maximizarea față de toate stările precedente $S_{k-1} = s'$ din relația (24), vor fi de fapt doar pentru două stări precedente $S_{k-1} = s'$ care au cale spre starea actuală $S_k = s$. Pentru toate celelalte valori ale lui *s'* vom avea $\gamma_k(s',s) = 0$.

Asemănător cu relația (24) folosită pentru a calcula $A_k(s)$ printr-o recursivitate înainte, relația (13) poate fi rescrisă astfel:

$$B_{k-1} \triangleq \ln\left(\beta_{k-1}(s')\right) \approx \max_s \left(B_k(s) + \Gamma_k(s',s)\right) \tag{25}$$

pentru a calcula valorile $B_{k-1}(s')$ printr-o recursivitate înapoi. Se vede asemănarea cu algoritmul Viterbi, cu excepția sensului de parcurgere a trellisului.

Folosind relațiile (14) şi (15), se poate scrie metrica de ramură $\Gamma_k(s',s)$ ca:

$$\Gamma_k(s',s) \triangleq \ln\left(\gamma_k(s',s)\right) = C + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl} \tag{26}$$

unde *C* nu depinde de u_k sau de cuvântul de cod transmis $\underline{x}_k$, astfel că va fi considerat o constantă şi va fi omis. Astfel metrica de ramură este echivalentă cu cea din algoritmul Viterbi, cu adunarea LLR a priori $u_k L(u_k)$. In plus, termenul $\sum_{l=1}^{n} y_{kl}x_{kl}$ care reprezintă corelația, este ponderat cu L_c, gradul de siguranță al canalului (relația 18).

In final, din relația (9) putem scrie pentru LLR-urile a posteriori $L(u_k|\underline{y})$ calculate cu algoritmul Max-Log-MAP :

$$L(u_k|\underline{y}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\Rightarrow u_k=-1}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right) \tag{27}$$

$$\approx \max_{(s',s)\Rightarrow u_k=+1}(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s))-\max_{(s',s)\Rightarrow u_k=-1}(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s))$$

Aceasta înseamnă că în algoritmul Max-Log-MAP, pentru fiecare bit u_k se calculează LLR a posteriori $L(u_k|\underline{y})$ prin considerarea fiecărei tranziții între nivelurile S_{k-1} şi S_k ale trellisului. Aceste tranziții sunt apoi grupate în cele care pot apărea dacă $u_k=+1$ şi cele care pot apărea daca u_k = -1. Pentru ambele grupuri se găseşte tranziția care dă valoarea maximă pentru $A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)$ şi LLR a posteriori se calculează doar pe baza acestor două tranziții, cele mai bune.

Algoritmul Max-Log-MAP poate fi rezumat după cum urmează. Se foloseşte, la fel ca la algoritmul Viterbi, un calcul recursiv înainte, respectiv înapoi, pentru determinarea lui $A_k(s)$ cu relația (24), şi $B_k(s)$ cu relația (25). Metrica de ramura $\Gamma_k(s',s)$ se calculează cu relația (26) în care poate fi omisă constanta C. Apoi, după executarea ambelor recursivități înainte şi înapoi, se poate calcula LLR a posteriori, cu relația (27). In acest fel, complexitatea algoritmului Max-Log-MAP nu este semnificativ mai mare decât cea a algoritmului Viterbi, adică în loc de o recursivitate apar două, metrica de ramură din (26) are un termen suplimentar a priori $u_k L(u_k)$ adunat la ea, şi pentru fiecare bit trebuie folosită relația (27) pentru a da LLR-urile a posteriori. Viterbi a constatat că algoritmul Max-Log-MAP are o complexitate de doar trei ori mai mare decât a decodorului Viterbi [WH00]. Din păcate memoria necesară este mult mai mare, deoarece trebuie memorate atât metricile de cale $A_k(s)$ calculate prin recursivitate înainte, cât şi $B_k(s)$ calculate prin recursivitate înapoi, înainte de a putea calcula $L(u_k|\underline{y})$. Dar, aşa cum tot Viterbi a arătat, printr-o creştere relativ redusă a volumului de calcul, de aproximativ patru ori față de cea a algoritmului Viterbi, necesarul de memorie scade foarte mult, devenind practic egal cu cel al decodorului Viterbi.

Corectarea aproximărilor. Algoritmul Log-MAP

Algoritmul Max-Log-MAP are o mică scădere de performanță față de algoritmul MAP, din cauza aproximării dată de relația (20), care în cazul turbo codurilor este în jur de 0,35 dB [WH00]. Folosind însă logaritmul Jacobian, poate fi scrisă o relație exactă, evitând aproximarea din relația (20), astfel:

$$\begin{aligned}\ln\left(e^{x_1}+e^{x_2}\right) &= \max\left(x_1,x_2\right)+\ln\left(1+e^{-|x_1-x_2|}\right)\\ &= \max\left(x_1,x_2\right)+f_c\left(|x_1-x_2|\right)=g\left(x_1,x_2\right)\end{aligned} \qquad (28)$$

unde $f_c(x)$ poate fi considerat ca un termen de corecție. Acest lucru stă la baza algoritmului Log-MAP propus de Robertson, Villebrun şi Hoeher [.....vucetici....]. In acest algoritm, se calculează valorile pentru $A_k(s)\triangleq\ln(\alpha_k(s))$ şi $B_k(s)\triangleq\ln(\beta_k(s))$ asemănător ca în algoritmul Max-Log-MAP, folosind recursivitatea înainte şi înapoi, dar maximizările din relațiile (24) şi (25) sunt corectate de termenul $f_c(x)$ din relația (28). Termenul de corecție nu trebuie calculat pentru fiecare valoare a lui x, ci poate fi

memorat într-o tabelă. Autorii algoritmului Log-MAP au arătat că sunt suficiente opt valori de corecție, cuprinse între 0 şi 5. Algoritmul Log-MAP este puțin mai complex decât Max-Log-MAP, dar are aceleaşi performanțe ca algoritmul MAP, lucru care îl face adecvat şi atractiv pentru folosirea sa în decodoarele componente ale unui decodor turbo iterativ.

5. Algoritmul SOVA

Descrierea matematică a algoritmului SOVA

Algoritmul SOVA, un algoritm cu intrări soft şi ieşiri soft, este construit pe baza algoritmului Viterbi clasic. Față de acesta, algoritmul are două modificări care permit utilizarea sa în decodoarele turbo [VAJ95]. Mai întâi, metricile de cale sunt modificate pentru a ține cont de informația a priori când se selectează calea prin trellis. In al doilea rând, algoritmul e modificat astfel încât să genereze ieşiri soft sub forma de LLR a posteriori $L(u_k|\underline{y})$ pentru fiecare bit decodat.

Prima modificare e uşor de realizat. Considerăm secvența de stări $\underline{s}_k$ care dă stările de-a lungul căii supraviețuitoare la starea $S_k = s$ la momentul *k* de pe trellis. Probabilitatea ca aceasta să fie calea corectă prin trellis este:

$$p\left(\underline{s}_k^s \middle| \underline{y}_{j\le k}\right) = \frac{p\left(\underline{s}_k^s \wedge \underline{y}_{j\le k}\right)}{p\left(\underline{y}_{j\le k}\right)} \tag{29}$$

Cum numitorul este constant, adică probabilitatea secvenței recepționate $\underline{y}_{j\le k}$ pentru tranzițiile până la şi inclusiv a *k*-a tranziție, pentru toate căile $\underline{s}_k^s$, este constantă, rezultă că probabilitatea ca $\underline{s}_k^s$ să fie corectă este proporțională cu $p(\underline{s}_k^s \wedge y_{j\le k})$. Deci metrica ar trebui să fie astfel definită încât maximizarea să conste în maximizarea lui $p(\underline{s}_k^s \wedge y_{j\le k})$. De asemenea, metrica trebuie să fie uşor de calculat într-un mod recursiv, la trecerea de la nivelul *k*-1 la *k*. Dacă $\underline{s}_k^s$, calea de la nivelul *k*, are pentru primele *k*-1 tranziții calea $\underline{s}_{k-1}^{s'}$, atunci, presupunând că avem un canal fără memorie şi folosind definiția lui $\gamma_k(s',s)$ din relația (8), vom avea :

$$p\left(\underline{s}_k{}^s \wedge \underline{y}_{j\le k}\right) = p\left(\underline{s}_{k-1}^{s'} \wedge \underline{y}_{j\le k-1}\right) p\left(s \wedge \underline{y}_k \middle| s'\right) = p\left(\underline{s}_{k-1}^{s'} \wedge \underline{y}_{j\le k-1}\right) \gamma_k(s',s) \tag{30}$$

Metrica adecvată $M\left(\underline{s}_k^s\right)$ pentru calea $\underline{s}_k^s$ este definită ca:

$$M\left(\underline{s}_k{}^s\right) \triangleq \ln\left(p\left(\underline{s}_k^s \wedge \underline{y}_{j\le k}\right)\right) = M\left(\underline{s}_{k-1}{}^{s'}\right) + \ln\left(\gamma_k(s',s)\right) \tag{31}$$

Folosind relația (26) şi omițând constanta *C*, vom avea :

$$M(\underline{s}_k^s) = M(\underline{s}_{k-1}^{s'}) + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl} \tag{32}$$

Astfel, la algoritmul SOVA, metrica este actualizată ca la algoritmul Viterbi, cu termenul $u_k L(u_k)$ adăugat suplimentar, astfel încât să poată fi luată în considerare informația a priori disponibilă. Trebuie observat că acest lucru este echivalent cu recursivitatea înainte (relația 24) folosită pentru calculul lui $A_k(s)$ în algoritmul Max-Log-MAP.

Acum vom analiza a doua modificare a algoritmului, pentru a obține ieşiri soft. In trellisul binar există două căi care ajung în starea $S_k=s$ în nivelul k al trellisului. Algoritmul Viterbi modificat, care ține cont de informația a priori $u_k L(u_k)$, calculează metrica (relația 32) pentru ambele căi şi *renunță la calea de metrică minimă*. Dacă cele două căi $\underline{s}_k^s$ şi $\underline{\hat{s}}_k^s$ au metricile $M(\underline{s}_k^s)$ şi $M(\underline{\hat{s}}_k^s)$ şi este selectată calea $\underline{s}_k^s$ ca supraviețuitor, deoarece metrica sa este mai mare, atunci putem defini diferența metricilor Δ_k^s ca:

$$\Delta_k^s = M(\underline{s}_k^s) - M(\underline{\hat{s}}_k^s) \geq 0. \tag{33}$$

Probabilitatea de a fi luat o decizie corectă selectând calea $\underline{s}_k^s$ ca supraviețuitor şi renunțarea la calea $\underline{\hat{s}}_k^s$ este deci:

$$P(\text{decizie corectă la } S_k = s) = \frac{P(\underline{s}_k^s)}{P(\underline{s}_k^s) + P(\underline{\hat{s}}_k^s)}. \tag{34}$$

Ținând cont de definirea metricii de cale (relația 31) avem :

$$P(\text{decizie corectă la } S_k = s) = \frac{e^{M(\underline{s}_k^s)}}{e^{M(\underline{s}_k^s)} + e^{M(\underline{\hat{s}}_k^s)}} = \frac{e^{\Delta_k^s}}{1 + e^{\Delta_k^s}} \tag{35}$$

şi LLR ca aceasta să fie decizia corectă e dată pur şi simplu de Δ_k^s.

In figura 5 e prezentată o secțiune simplificată a trellisului pentru codul RSC, cu K=3 şi diferențele metricilor Δ_k^s marcate în diferite puncte ale trellisului.

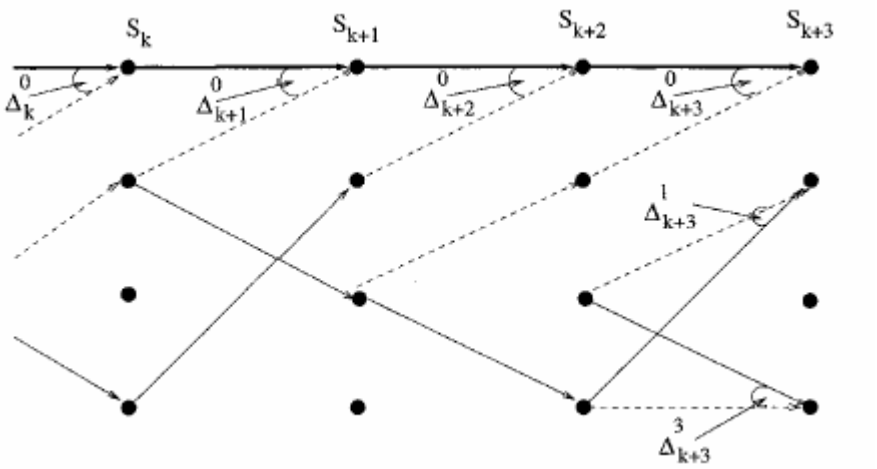

Fig. 5: Secțiune simplificată a trellisului pentru codul RSC, K=3, cu decodare SOVA.

Când se ajunge la sfârşitul trellisului şi am identificat calea ML prin trellis trebuie să găsim LLR-urile care dau gradul de încredere al deciziilor asupra biților, de-a lungul căii ML. Cu privire la algoritmul Viterbi s-a făcut constatarea că toți supraviețuitorii de la nivelul l din trellis provin, în mod normal, de la aceeaşi cale, dintr-un punct plasat cu l puncte înainte în trellis. Acest punct e la maxim δ tranziții înainte de l, unde de obicei δ este de cinci ori lungimea de constrângere a codului convoluțional. Astfel, valoarea bitului u_k asociat tranziției din starea $S_{k-1}=s$ pe calea ML, poate diferi, dacă în loc de calea ML algoritmul Viterbi selectează una din căile ce se uneşte cu calea ML ulterior, după maxim δ tranziții, adică în nivelul $k+\delta$. Din cele expuse rezultă că bitul u_k nu va fi afectat dacă algoritmul selectează oricare din căile care se unesc cu calea ML după acest punct, deoarece o asemenea cale va diverge de la calea ML după tranziția dintre $S_{k-1}=s'$ şi $S_k=s$. Astfel, când se calculează LLR pentru bitul u_k , algoritmul SOVA trebuie să țină cont de probabilitatea ca să fi fost incorect descărcate căile care se unesc cu calea ML din nivelul k în nivelul $k+\delta$. Acest lucru se face considerând valorile diferențelor metricilor $\Delta_i^{s_i}$ pentru toate stările s_i existente de-a lungul căii ML între nivelurile $i=k$ şi $i=k+\delta$. Hagenauer a arătat că [JH95] acest LLR poate fi aproximat de:

$$L\left(u_k \middle| \underline{y}\right) \approx u_k \min_{i=k\ldots k+\delta; u_k \neq u_k^i} \Delta_i^{s_i} \tag{36}$$

unde u_k este valoarea bitului dată de calea ML, iar u_k^i este valoarea acestui bit pentru calea care se uneşte cu calea ML, dar va fi descărcată la nivelul i al trellisului. Astfel, minimizarea din relația (36) este făcută doar pentru căile care se unesc cu calea ML, care ar putea da o valoare diferită pentru bitul u_k dacă au fost alese ca supraviețuitori. Căile care se unesc cu calea ML, care ar fi dat aceeaşi valoare pentru u_k, evident că nu afectează gradul de siguranță al deciziei asupra lui u_k.

Liniile continue din figura 5 reprezintă tranzițiile atunci cand bitul de intrare este -1 iar liniile punctate reprezinta tranzițiile pentru u_k = +1. Calea ML se presupune a fi calea nulă, formată numai din zero-uri şi este trasată cu linie îngroşată. Sunt reprezentate şi căile care se unesc cu calea ML. Se arată şi cum calea ML dă o valoare de -1 pentru bitul u_k, dar căile care se unesc cu calea ML la nivelurile S_k, S_{k+1}, şi S_{k+3} ale trellisului, dau

toate valorile +1 pentru bitul u_k. Astfel, dacă se presupune $\sigma = 3$, din relația (36), LLR-ul $L\left(u_k \left| \underline{y} \right.\right)$ va fi dat de -1 înmulțit cu minimul dintre diferențele metricilor Δ_k^0, Δ_{k+1}^0 şi Δ_{k+3}^0.

Implementarea algoritmului SOVA

Pentru fiecare stare, de la fiecare nivel, se calculează metrica $M\left(\underline{s}_k^s\right)$ pentru ambele căi ce intră într-o stare, folosind relația (32). Calea cu cea mai mare metrică e selectată ca supraviețuitor şi, pentru această stare, la acest nivel din trellis se memorează un pointer spre calea precedentă de-a lungul căii supraviețuitoare. Totuşi, se memorează şi informația folosită în relația (36), pentru a permite şi calculul gradului de siguranță pentru biții decodați şi anume $L\left(u_k \left| \underline{y} \right.\right)$. Astfel, diferența Δ_k^s dintre metricile supraviețuitorilor şi a căilor descărcate sunt memorate, împreună cu vectorul binar conținând δ+1 biți, care indică, dacă calea descărcată ar fi dat sau nu aceeaşi serie de biți u_l pentru $l=k$ înapoi la $l=k-\delta$, la fel cum ar fi făcut supraviețuitorul. Această serie de biți este numită actualizarea secvenței în [JH95] şi, după cum a observat Hagenauer, e rezultatul sumării modulo 2 (SAU-EXCLUSIV) dintre δ+1 biți anteriori decodați de-a lungul supraviețuitorului şi căile descărcate. Când algoritmul SOVA a identificat calea ML, secvența de actualizare memorată şi diferențele metricilor de-a lungul acestei căi sunt folosite în relația (36) pentru a calcula valorile lui $L\left(u_k \left| \underline{y} \right.\right)$.

Algoritmul SOVA este cel mai puțin complex dintre toți algoritmii folosiți în decodoarele prezentate aici. Robertson ş.a. au arătat că algoritmul are o complexitate pe jumătate față de cea a algoritmului Max-Log-MAP. In schimb algoritmul SOVA este cel mai puțin precis dintre algoritmii prezentați şi, când e folosit într-un decodor turbo iterativ, duce la performanțe cu aproximativ 0,6 dB mai scăzute decât prin folosirea algoritmului MAP.

6. Compararea algoritmilor decodoarelor componente

Algoritmul MAP este un decodor component optimal pentru turbo coduri. El găseşte probabilitatea fiecărui bit u_k de a fi +1 sau -1, calculând probabilitatea tranziției din starea $S_{k-1} = s'$ la $S_k = s$ care poate apărea daca bitul de intrare a fost +1 şi la fel pentru fiecare tranziție care poate apărea dacă bitul de intrare a fost -1. Deoarece tranzițiile sunt mutual exclusive, probabilitatea ca să apară oricare dintre ele este suma probabilităților individuale şi astfel LLR-ul pentru bitul u_k e dat de raportul celor două sume de probabilități, ca în relația (3).

Din cauza naturii Markov a trellisului şi presupunerii că ieşirea trellisului este observată în zgomot fără memorie, probabilitățile individuale ale tranzițiilor din relația (3), pot fi exprimate ca produsul a trei termeni $\alpha_{k-1}\left(s'\right)$, $\beta_k\left(s\right)$ şi $\gamma_k\left(s', s\right)$ ca în relația

(5). Termenii $\gamma_k(s',s)$ pot fi calculaţi din probabilităţile a priori ale biţilor decodaţi şi informaţia de canal recepţionată, ca în relaţiile (14) şi (15). Algoritmul MAP este optimal pentru decodarea turbo codurilor dar este extrem de complex. In plus, din cauza înmulţirilor necesare la calculul recursiv a lui $\alpha_{k-1}(s')$ şi $\beta_k(s)$ apar deseori probleme practice de calcul. Algoritmul Log-MAP este teoretic identic cu algoritmul MAP, dar transferă operaţiile în domeniul logaritmic. Astfel, înmulţirile sunt înlocuite cu sume şi problemele de calcul numeric de la algoritmul MAP sunt evitate, complexitatea algoritmului fiind redusă spectaculos.

Algoritmul Max-Log-MAP reduce în continuare complexitatea algoritmului Log-MAP folosind aproximarea prin maximizare (relaţia 20). Acest lucru are, comparativ cu algoritmul Log-MAP, două efecte la operare. In primul rând, după cum se poate vedea din relaţia (27), înseamnă ca sunt luate în considerare doar doua tranziţii când se găseşte LLR $L(u_k \mid \underline{y})$ pentru fiecare bit u_k – cea mai bună tranziţie de la $S_{k-1} = s'$ la $S_k = s$ care ar da u_k=+1 şi cea mai buna care ar da u_k = -1. Asemănător, în calculul recursiv al termenilor $A_k(s)$=ln ($\alpha_k(s)$) şi $B_k(s)$= ln ($\beta_k(s)$) din relaţiile (24) şi (25), aproximarea înseamnă că doar o tranziţie, cea mai plauzibilă, e considerată când se calculează termenii $A_k(s)$ din $A_{k-1}(s')$ şi $B_{k-1}(s')$ din $B_k(s)$. Aceasta înseamnă ca deşi $A_{k-1}(s')$ ar trebui să dea logaritmul probabilităţii ca trellisul să ajungă în starea $S_{k-1} = s'$ de-a lungul *oricărei* căi pornind din starea S_0 =0, de fapt dă logaritmul probabilităţii doar a *celei mai plauzibile căi* spre starea $S_{k-1} = s'$. Asemănător, $B_k(s)$= ln ($\beta_k(s)$) ar trebui să dea logaritmul probabilităţii secvenţei recepţionate $\underline{y}_{j>k}$ doar dacă trellisul se află în starea $S_k = s$ la nivelul *k*. Totuşi, maximizarea din relaţia (25) folosită în calculul recursiv al termenului $B_k(s)$ înseamnă că doar calea cea mai plauzibilă dintre $S_k = s$ şi sfârşitul trellisului este luată în considerare, şi nu toate căile.

Astfel, algoritmul Max-Log-MAP găseşte LLR $L(u_k \mid \underline{y})$ pentru un bit u_k dat, comparând probabilitatea celei mai plauzibile căi care dă u_k = +1 cu probabilitatea celei mai plauzibile căi care dă u_k = -1. Pentru următorul bit, u_{k+1} , se compară din nou cea mai bună cale care ar da u_k = +1, respectiv cea mai bună cale care ar da u_k =-1. Una dintre aceste „cele mai bune căi” va fi întotdeauna calea ML, şi astfel nu se va schimba de la un nivel la altul, în timp ce cealaltă s-ar putea schimba. Algoritmul Log-MAP, spre deosebire de MAP, consideră fiecare cale în calculul LLR pentru fiecare bit. Tot ceea ce se schimbă de la un nivel la altul este împărţirea căilor în cea care dă u_k = +1 şi cea care dă u_k = -1. Astfel , algoritmul Max-Log-MAP are performanţe mai scăzute faţă de MAP şi Log-MAP.

In algoritmul SOVA, calea ML e găsită maximizând metrica din relaţia (32). Recursivitatea folosită pentru a găsi aceasta metrică e identică cu cea folosită pentru găsirea termenului $A_k(s)$ din relaţia (24), la algoritmul Max-Log-MAP. Odată aflată calea ML, decizia hard pentru un bit dat u_k e determinată de tranziţia care a avut loc pe calea ML între nivelurile S_{k-1} şi S_k ale trellisului. LLR pentru acest bit $L(u_k \mid \underline{y})$ e determinat analizând căile care se unesc cu calea ML şi ar avea o decizie hard diferită pentru bitul u_k. LLR se ia ca fiind minimul diferenţei metricilor pentru aceste căi. Folosind notaţiile de la algoritmul Max-Log-MAP, odată ce o cale s-a unit cu calea ML, ea va avea aceeaşi valoare $B_k(s)$ ca şi calea ML. Astfel, cum metrica de la algoritmul SOVA e identică cu valorile $A_k(s)$ de la algoritmul Max-Log-MAP, luând diferenţa

dintre metricile celor două căi care se unesc, în algoritmul SOVA e echivalent cu luarea diferenței dintre cele două valori ale $(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s))$ în algoritmul Max-Log-MAP, ca în relația (27). Singura diferență este ca în algoritmul Max-Log-MAP una din căi va fi calea ML, şi cealaltă va fi cea mai plauzibilă cale care dă o decizie hard diferită pentru u_k. In algoritmul SOVA, una din căi va fi din nou calea ML, dar cealaltă cale care dă o decizie hard diferită pentru u_k, s-ar putea să nu fie calea de plauzibilitate maximă. In schimb ea va fi cea mai plauzibilă cale care dă o decizie hard diferită pentru u_k, şi supraviețuieşte ca să se unească cu calea ML. Alte căi, mai plauzibile, care dau o decizie hard diferită pentru bitul u_k față de calea ML, ar putea fi descărcate înainte de a se uni cu calea ML. Astfel că algoritmul SOVA are o performanță mai slabă comparativ cu algoritmul Max-Log-MAP. Totuşi cei doi algoritmi furnizează aceleaşi decizii hard, care în ambele cazuri sunt determinate de calea ML, decizii calculate folosind aceeaşi metrică la ambii algoritmi.

Complexitatea celor trei algoritmi depinde de lungimea de constrângere K a codurilor convoluționale folosite, dar oricum algoritmul Max-Log-MAP e de două ori mai complex decât algoritmul SOVA. Algoritmul Log-MAP este cu ceva mai complex decât algoritmul Max-Log-MAP din cauza căutării în tabela de corecții a factorilor $f_c(x)$. Când se foloseşte decodarea turbo iterativă, performanța algoritmilor e de acelaşi ordin de mărime ca şi complexitatea lor, astfel că algoritmul Log-MAP are cea mai bună performanță, fiind urmat de algoritmul Max-Log-MAP şi la sfârşit de algoritmul SOVA, cu cea mai slabă performanță.

E posibilă decodarea optimală a turbo codurilor într-un singur pas, non-iterativ [BH00], din cauza complexității se preferă decodarea iterativă neoptimală. Un asemenea decodor iterativ are două decodoare componente cu intrări şi ieşiri soft, care pot folosi algoritmul MAP, Log-MAP, Max-Log-MAP sau SOVA. Algoritmul MAP e optimal dar foarte complex. Algoritmul Log-MAP e o simplificare a lui MAP având aceeaşi performanță optimă la o complexitate rezonabilă. Ceilalți doi algoritmi, Max-Log-MAP şi SOVA sunt mai puțin complecşi, dar au o performanță cu ceva mai scăzută.

Cercetările continuă pentru îmbunătățirea performanței şi scăderea complexității. De exemplu, Hagenauer ş.a. lucrând la coduri turbo bazate pe coduri bloc a obținut performanțe aproape de limita Shannon, la o rată de codare aproape de 1, deşi cu un efort de decodare mare. Eforturi se depun şi în sistemele de transmisiuni video sau a vocii, folosind turbo codurile. Multe aplicații sunt la sistemele video interactive [HCS], precum şi în rețelele locale. Se aşteaptă ca cercetările să conducă la aceleaşi performanțe ale turbo codurilor şi pe canale dispersive afectate de fading.

7. Utilizarea turbo codurilor în marcarea transparentă a imaginilor

Marcarea transparentă WM (WaterMarking) este un proces prin care un semnal este ascuns sau înglobat într-alt semnal, de obicei o fotografie, voce sau video. Există o serie de tehnici de înglobare a semnalizării care cuprind de la steganografia clasică până la aplicațiile actuale referitoare la interesele comerciale cu privire la drepturile de autor şi

controlul la copiere al conținutului. In acest ultim caz, un avantaj al înglobării semnalului este că informația de control prin copiere este înglobată direct în datele media care trebuie protejate şi astfel este independentă de difuzare şi/sau formatul transmisiei, rămânând prezentă şi după decriptare [CMM99].

Marcarea transparentă este o formă de comunicație. Condiția ca fidelitatea conținutului documentului media să nu fie afectată de către semnalul de marcare, implică faptul că semnalul de marcare să fie foarte mic comparativ cu cel asociat conținutului documentului, asemănător cu condițiile impuse puterii din comunicațiile tradiționale. Acest lucru, asociat cu faptul că din punctul de vedere al detectării semnalului de marcare, semnalul care reprezintă documentul apare ca un zgomot, a condus la concluzia că marcarea transparentă este o formă de comunicație cu spectru împrăştiat, SS (Spread-Spectrum) [PL02,CMM99]. Pentru ca marcajul să fie imperceptibil, el trebuie să aibă o putere foarte mică, astfel încât transmiterea să apară ca o formă de comunicație printr-un canal extrem de zgomotos. Se ştie faptul că tehnicile SS facilitează comunicația prin medii zgomotoase, şi de aceea, sunt mult folosite în marcarea transparentă. Majoritatea schemelor WM reprezintă informația (payload) sub forma unei secvențe de zgomot pseudo-aleator, PN (Pseudo-Noise), aşa numita secvență directă, DS (Direct Sequence) din SS. Astfel, modul de generare a secvenței aleatoare devine cheia WM. Pentru decodarea marcajului WM, decodorul trebuie să cunoască cheia, astfel că aceste scheme sunt, prin excelență, private. Hartung ş.a., citat în [PL02], a propus ca secvența PN să fie publică, pentru a permite decodarea publică a marcajului. Detectorul/decodorul mai trebuie, de asemenea, să se sincronizeze cu secvența PN, înainte ca să aibă loc detectarea/decodarea. Acest lucru reprezintă un dezavantaj major al multor scheme existente. O variație față de principiul SS de bază, este filtrarea trece bandă/trece jos a secvenței, anterior înglobării marcajului, astfel încât el va avea o energie foarte mică în componentele de frecvență înaltă, care tinde să fie eliminată de către sistemele de compresie.

Pentru a îmbunătăți robustețea marcajului, se utilizează deseori, coduri de control a erorilor, pentru a coda informația utilă înainte de înglobare. Dintre aceste coduri sunt de menționat turbo codurile. Valenti [VAL96] a analizat turbo codurile ca şi coduri liniare şi a obținut o limită superioară a erorii pe cuvânt, WER (Word Error Rate) aproximativă, în condițiile AWGN:

$$P_{err,turbo} \approx\leq \sum_{w=d_{\min}}^{w=d_{\max}} A_w \; Q\left(\sqrt{wR_c SNR}\right),$$

unde d_{min} şi d_{max} sunt distanțele Hamming minimă şi maximă ale codului, A_w este numărul de cuvinte de cod cu ponderea w , R_c este rata codului iar SNR este raportul semnal pe zgomot. Curbele BER au un gradient negativ. Dacă BER la intrare este mai mic decât un prag de zgomot gaussian, respectiv SNR la intrare este mai mare decât un prag, se pot obține erori de decodare foarte mici, utilizând turbo-codurile. Acest lucru are o importanță deosebită în marcarea transparentă [KMKM00].

Majoritatea cercetărilor în marcarea transparentă efectuate până acum s-au axat pe căi noi de ascundere/înglobare a informației şi pe căi de detecție şi/sau extragere a acestei informații ascunse. Se creează în prezent o teorie care stabileşte limitele fundamentale ale

oricărei scheme WM de marcare transparentă. Problema WM poate fi văzută ca o problema modificată de codare canal-sursă, cu condiții impuse ratei şi distorsiunii.

Rolul principal al codării canalului este de fapt de a îngloba semnalul de marcare în datele originale, cu o distorsiune redusă şi de a oferi o protecție față de atacuri, care distorsionează, atât pentru datele originale, cât şi pentru semnalul de marcare.

In cele ce urmează e introdusă mai întâi, problema marcării transparente în general, apoi sunt abordate strategiile optime de urmat de către transmițător şi atacator. Este analizată şi abordarea acestei probleme prin cuantizare, combinată cu codarea canalului. Apoi atenția este îndreptată spre tehnici de decodare iterativă şi aplicabilitatea codurilor decodabile iterativ la marcarea transparentă. Sunt apoi prezentate graficele performanțelor diferitelor coduri şi dezvoltări posibile în viitor.

Marcarea transparentă

Recent a fost dezvoltată o teorie care stabileşte limitele fundamentale în cazul general al marcării transparente, descrise mai jos. Un mesaj M trebuie comunicat unui receptor. Mesajul e înglobat într-o secvență de lungime N, $\tilde{X}^N = (\tilde{X}_1, ..., \tilde{X}_N)$ numit *setul gazdă al datelor*, tipic date de la gazdă ca o imagine/video/semnal audio. Înglobarea se face folosind o cheie criptografică $K^N = (K_1, ..., K_N)$ care este disponibilă şi la decodor. Rezultatul reprezintă *datele marcate transparent* sau *datele compuse* şi ele sunt supuse *atacurilor*, care încearcă să înlăture orice urmă a lui M din X^N. Procesul de înglobare a datelor trebuie să fie *transparent*: X^N trebuie să fie similar cu $\tilde{X}^N$, conform cu o măsură a distorsiunii adecvată (fig.6).

Sistemul trebuie şi să fie robust: mesajul înglobat trebuie să reziste oricărui atac (dintr-o clasă rezonabilă de atacuri). O restricție tipică impusă atacatorului este o limită a gradului de distorsionare care se introduce.

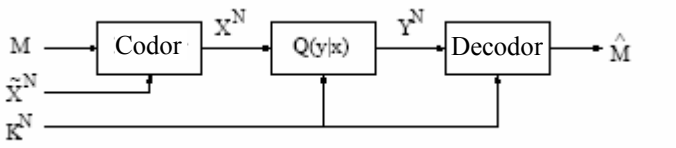

Fig. 6: O schemă generală de marcare transparentă

Sistemul poate fi analizat definind modelul statistic pentru M, $\tilde{X}^N$ şi K^N, o funcție de distorsiune, specificând condițiile pentru nivelurile de distorsiune admisibile, D_1 şi D_2 pentru cel care înglobează datele respectiv pentru atacator, şi specificând informația disponibilă tuturor părților. Apoi, se poate căuta *rata maximă pentru o transmisie sigură* pentru M, față de *orice* strategie posibilă de înglobare a datelor, şi față de *orice* atac care satisface condițiile specificate. Acest lucru se face prin aplicarea principiilor din teoria informației şi, în particular, conform conceptului fundamental des folosit în literatura referitoare la marcarea transparentă: chiar dacă semnalul gazdă $\tilde{X}^N$ nu e disponibil la decodor (*marcare transparentă blind/oarbă*), faptul că la codor e cunoscut N, înseamnă că rata care poate fi atinsă este mai mare decât dacă $\tilde{X}^N$ ar fi fost o interferență

necunoscută. Această problemă cade în categoria problemelor de comunicații în care codorul şi decodorul au acces la informații suplimentare (side informations). *Capacitatea de înglobare*/*ascundere* limitează ratele unei transmisii sigure şi depinde de alegerea funcției de distorsiune şi a nivelurilor de distorsionare admisibile D_1 şi D_2.

Atât pentru marcarea transparentă oarbă (blind) cat şi pentru marcarea non-oarbă au fost găsite formule concise pentru semnalele gazdă gaussiene i.i.d. (infinit şi identic distribuite) şi funcțiile de distorsiune de tip eroare medie pătratică. Se presupune că atacatorul nu cunoaşte funcțiile de codare şi de decodare, ca de exemplu funcții care depind de o cheie criptografică. In ambele cazuri, atacul este canalul de test gaussian din teoria rată-distorsiune [KMKM00]. Capacitatea are expresia:

$$C = \frac{1}{2}\log\left(1 + \frac{D_1}{\beta D_2}\right),$$

unde $\beta \geq 1$ este un „factor de creştere a zgomotului" care tinde la 1 pentru nivele de distorsiune D_1 mici, $D_2 \ll \sigma^2$, unde σ^2 este dispersia semnalului gazdă. De remarcat este faptul că C este aceeaşi atât pentru marcarea oarbă cât şi pentru marcarea non-oarbă, şi valoarea sa limită e independentă de σ^2, dacă D_1, $D_2 \ll \sigma^2$.

Stabilirea structurii sistemului

Formula capacității, prezentată anterior, sugerează o strategie de codare şi decodare. In această secțiune sunt descrise procedura de marcare transparentă şi blocurile principale. In figura 7 este descrisă procedura generală de înglobare. Presupunem că semnalul gazdă $\tilde{X}$ este distribuit conform cu o distribuție gaussiană de medie nulă şi dispersie σ^2, adică $\tilde{X} \sim N(0, \sigma^2)$. Colecția dicționarelor cuvintelor de cod $CB_1, CB_2, ..., CB_M$, i.e. subseturi discrete ale spațiului euclidian $\mathbf{R}^N$ folosit pentru cuantizarea vectorială a semnalului gazdă $\tilde{X}$. Se presupune că alegerea dicționarelor este aleatoare. Transmițătorul marcajului introduce informația în semnalul gazdă $\tilde{X}$, alegând unul din cele din cele M dicționare posibile, pentru etapa de cuantizare vectorială. Cele M dicționare sunt astfel alese încât distorsiunea estimată cauzată de aceasta etapă să nu depăşească D_1 impusă. Cuantizarea se face asupra unei versiuni scalate $\alpha\tilde{X}$ a semnalului gazdă, generându-se vectorul U. Aici, parametrul α este un factor de scalare $\alpha = D_1/(D_1 + \beta D_2)$, unde $\beta = (\sigma^2 + D_1)/(\sigma^2 + D_1 - D_2)$. In final se transmite semnalul $X = U + (1-\alpha)\tilde{X}$.

Semnalul X e transmis printr-un *canal de atac*, un dispozitiv asupra căruia atacatorul are un control total, obținându-se semnalul recepționat Y. Singura condiție impusă asupra canalului de atac este ca el să ducă la o distorsiune dintre X şi Y mai mică decât D_2. Sarcina receptorului este estimarea sigură a indexului dicționarului de coduri folosit pentru cuantizare. Pentru a îndeplini aceasta sarcină, vectorul recepționat este mai întâi

scalat cu factorul $\gamma = (D_1 + \alpha\sigma^2)/(D_1 + D_2 + \beta\sigma^2)$. Semnalul rezultat este apoi decodat cu codul *C*, care este reuniunea celor *M* dicţionare de cod CB_i.

Dacă transmiterea marcajului a fost făcută cu succes, receptorul îl va decoda ca şi cuvânt de cod din dicţionarul ales la procedura de codare şi astfel el va putea reconstrui mesajul *E*. Se arată că această codare aleatoare determină o transmisie sigură a marcajului, la orice rata $R = \log M / N$ care satisface condiţia $R < C$.

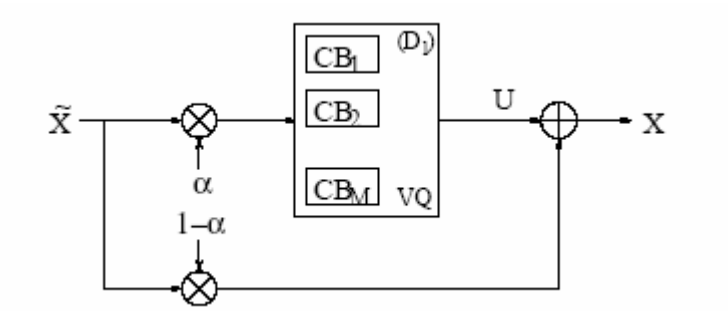

Fig. 7: Structura codorului pentru schema de watermarking

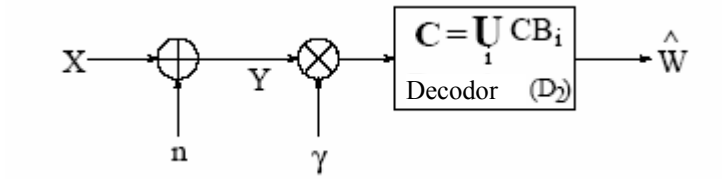

Fig. 8: Structura decodorului pentru schema de watermarking

Abordarea unei codări realizabile practic

Schema de codare prezentată în secţiunea precedentă are dezavantajul oricărei scheme de codare aleatoare, adică e nefezabilă din cauza complexităţii. Pentru a obţine o soluţie realizabilă practic a problemei marcării transparente, sunt necesare scheme de codare de complexitate redusă, care să menţină totuşi elementele de bază ale schemelor de codare aleatoare. In acest context, sarcina principală este găsirea unui dicţionar de coduri *CB* care constă din reuniunea unor subcoduri disjuncte CB_i, cu proprietatea că *CB* este un dicţionar de coduri bun şi *practic* pentru *cuantizarea vectorială*. Rezolvarea simultană a acestor două cerinţe se dovedeşte a fi sarcina principală în obţinerea unor scheme de codare pentru marcarea transparentă.

Atenţia va fi centrată doar pentru cazul distorsiunilor mici, adică $(D_1, D_2) \ll \sigma^2$. In plus, se presupune ca dicţionarele de coduri CB_i şi *CB* au o structura latice. Fie **C** un cod binar liniar de lungime *N* şi dimensiune *K*. Laticea *CB* obţinută din construcţia A [KMKM00] şi din codul **C** e definită ca $CB=\{x \mid x \in \mathbf{Z}^N : x \bmod 2 \in \mathbf{C}\}$. Fie **C'** un subcod liniar al codului **C.** Se pot găsi sublatice ale lui *CB* ca $CB' = \{x \mid x \in \mathbf{Z}^N : x \bmod 2 \in \mathbf{C'}\}$. Subcodul **C'** generează coseturile |**C**| /|**C'**| ale lui **C'** ţinând cont de **C.** Construcţia A aplicată acestor coseturi generează |**C**| /|**C'**| translaţii disjuncte ale laticei *CB'*. Se alege subcodul **C'** ca fiind codul cel mai simplu ce constă din cuvinte nule (cu toate componentele nule), şi obţinem sublaticea $CB'=2\mathbf{Z}^N$. Coseturile lui **C'** ţinând cont de **C**

sunt date chiar de cele 2^K cuvinte de cod ale lui **C** şi luăm 2^K translaţii ale laticei $2\mathbf{Z}^N$ să fie dicţionare pentru partea de cuantizare vectorială a schemei de codare pentru marcarea transparentă. Alegerea lui **C'** ca şi cod nul e tentantă din mai multe motive. Scopul este de a găsi coduri bune şi practice pentru cuantizarea vectorială şi codarea canalului, calităţi care depind foarte mult de codul **C** ales. Turbo codurile şi alte coduri decodabile iterativ reprezintă o alegere excelentă pentru acest cod. Pentru cuantizarea vectorială s-a ales o latice extrem de simplă şi practică, $2\mathbf{Z}^N$, pentru care schema de codare vectorială poate fi implementată uşor cu un cuantizor scalar. In figura 9 e prezentată procedura de codare pentru schema aleasă. Mesajul W dat este codat în cuvânt de cod al codului binar **C**, obţinând cuvântul de cod **C**(W).

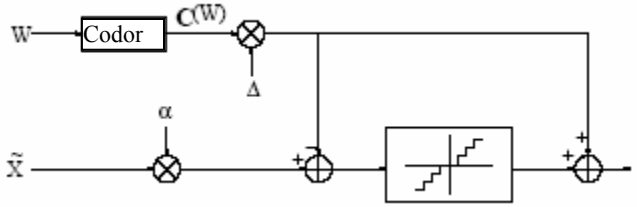

Fig. 9: Codor posibil pentru schema de marcare transparentă.

Cuantizarea vectorială constă din găsirea celui mai apropiat punct de latice din laticea translatată $\Delta(2\mathbf{Z}^N+\mathbf{C}(W))$, unde Δ este un parametru ales pentru ajustarea distorsiunii D_1. O cale de implementare eficientă a acestei proceduri este translatarea semnalului $\alpha\tilde{X}$ înainte de a face cuantizarea scalară (care este echivalentă cu un cuantizor latice $\Delta 2\mathbf{Z}^N$). Această translaţie a lui $\alpha\tilde{X}$ este compensată pentru versiunea cuantizată a lui $\alpha\tilde{X}$.

Regăsirea marcajului din semnalul distorsionat X se face prin decodarea dicţionarului latice. Structura decodorului e prezentata în figura 8. Semnalul care transportă informaţia distorsionată e recepţionat din canal şi scalat cu parametrul γ, pentru a găsi estimatul afectat de zgomot al ieşirii actuale a cuantizorului vectorial. Se ştie că $Y = X + n$ unde $X=U+(1-\alpha)\tilde{X}$. De asemenea, notând cu E eroarea de cuantizare introdusă în etapa de cuantizare vectorială, se obţine $\tilde{X} = (U + E)/\alpha$ şi $U = \alpha Y - \{E(1-\alpha) + n\alpha\}$. Deoarece avem nevoie de un estimat al lui U ca intrare în decodor, se scalează mai întâi cu parametrul α ieşirea canalului (cele expuse sunt valabile doar pentru distorsiuni mici).

Decodarea laticei CB este relativ simplă dacă alegem codul binar **C** pentru care e disponibil un algoritm de decodare eficient. Astfel, cea mai naturală alegere e dată de turbo coduri şi coduri produs decodabile iterativ. Prima sarcină este obţinerea cuvântului de cod **C**(W) care a fost folosit în translatarea cuantizorului scalar din figura 7. Trebuie să obţinem un estimat soft pentru fiecare bit individual din **C**(W) pentru a porni algoritmul de decodare turbo. Schema de cuantizare scalară ne permite să obţinem aceste estimate pentru fiecare poziţie individuală. Notăm cu Y_i valoarea recepţionată pe poziţia i a cuvântului de cod Y.
Se definesc $d_{0,i}$ şi $d_{1,i}$ ca fiind
$d_{0,i}=\min_j(Y_i - \Delta 2j)^2$ şi
$d_{1,i}=\min_j(Y_i - \Delta(2j+1))^2$
unde j este întreg în ambele cazuri . Astfel, $d_{0,i}$ şi $d_{1,i}$ sunt distanţele euclidiene pătratice ale valorilor recepţionate, faţă de cel mai apropiat nivel al cuantizorului scalar, în ipoteza

ca $\mathbf{C}(W)$ a fost egal cu 0 sau 1. Pentru a translata aceste distanțe în logaritmul raportului de plauzibilitate, presupunem că distribuția lui Y_i, fiind dată $\mathbf{C}(W)_i$, este gaussiană de medie nulă şi dispersie $\sigma^2 = \left(D_1 D_2\right)/(D_1 + D_2)$. In această ipoteză, logaritmul raportului de plauzibilitate este bine aproximat cu :

$$\ln\left(\frac{\Pr\left\{C(W)=0\middle|Y_i\right\}}{\Pr\left\{C(W)=1\middle|Y_i\right\}}\right) = \frac{d^2{}_{0j} - d^2{}_{1j}}{2\tilde{\sigma}^2}$$

In [KMKM00] au fost făcute simulări pentru câteva coduri binare: un cod produs [256,25,64] format din doua coduri Reed-Muller[16,5,8], un turbo cod de lungime 252 şi rata 1/9 rezultat prin concatenarea serială a două coduri convoluționale cu rata 1/3 şi matricea generatoare g=[100;111;101], şi un cod Hamming folosit ca element de referință. Cel mai bun dintre ele s-a dovedit a fi turbo codul, dar şi pentru acesta a rămas o distanță de 4,5 dB în raportul D_2/D_1 față de capacitatea obtenabilă, care poate fi acoperită în mare parte prin folosirea unor coduri mai elaborate şi de lungime mai mare.

Scopul a fost găsirea unor coduri bune şi realizabile practic din schema de marcare transparentă. Dar autorii constată ca partea de codare vectorială a schemei de codare este o sursă serioasă de pierderi, în particular datorită utilizării cuantizării scalare. Soluția propusă de aceştia este găsirea unor cuantizări vectoriale bune, care să permită transmiterea la capacitatea apropiată de limita teoretică. Altă sursă de pierderi este calculul logaritmului raportului de plauzibilitate în ipoteza zgomotului gaussian, ipoteza care este încălcată prin folosirea cuantizării scalare, deoarece zgomotul de cuantizare este uniform distribuit în bandă.

BIBLIOGRAFIE

[BCJR74] L.R.Bahl, J.Cocke, F.Jelinek, J.Raviv,„*Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate",*IEEE Trans. Info. Theory, vol. IT-20, Mar. 1974, pp.248-287

[BG96] C. Berrou, A.Glavieux, „ *Near Optimum Error Correcting Coding and Decoding: Turbo-Codes*", IEEE Trans. Commun., vol. 44, no. 10, Oct.1996, pp. 1262-1271

[BGT93] C. Berrou, A. Glavieux, P. Thitimajashima, *"Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes",* Proc.ICC'93, Geneva, Switzerland, May 1993, pp. 1064-1070

[BH00] M.Breiling, L.Hanzo, „*The Super-Trellis Strucure of Turbo Codes"*, IEEE Trans. Info. Theory, vol. 46, no. 6, Sept. 2000, pp.2212-2227

[BM96a] S.Benedetto, G.Montorsi „*Unveiling Turbo-Codes: Some Results on Parallel Concatenated Coding Schemes"* IEEE Trans. Info. Theory, vol. 42, pp. 409-428, Mar.1996

[BM96b] S.Benedetto, G.Montorsi, „*Design of Parallel Concatenated Convolutional Codes",* IEEE Trans. Commun., vol.44, pp. 591-600, May 1996

[BPGS02] Félix Balado, Fernando Pérez-González, and Sandro Scalise, „*Turbo coding for sample-level watermarking in the DCT domain",* Proc. of the IEEE International Conference on Image Processing (ICIP), pages 1003-1006, Thessaloniki, Greece, October 2001.

[CMM99] I.J.Cox, M.L.Miller, A.L.MCKellis, „*Watermarking as Communications with Side Effects"*, Proc. of IEEE ,vol. 87,no. 7,July 1999,pp1127-1141

[CPR02] J.Chou, S.S.Pradhan, K.Ramchandran, „*Turbo Coded Trellis Based Constructions for Data Embedding: Channel Coding with Side Information"*, In Proc. of Asilomar Conference on Signals, Systems and Computers, Pacific Grove (USA), October 2001

[DB03] C.Douillard, C.Berrou, „*M-Binary Turbo Codes",* March 31, Submitted to IEEE Trans.Communication

[DDP] S.Dolinar, D.Divsalar, E.Pollara, „*Code Performance as a Function of Block Size",* SPL, NASA,[on line], http://tmo.jpl.nasa.gov/tmo/progress_report/42-133/title.htm

[FB97] P.Perry, C.Berrou, „*Comparison on Pseudo-Syndrome and Syndrome Techniques for Synchronising Viterbi Decoders",* Intern. Symposium on Turbo Codes, Brest, France, 1997

[FV97] W.Feng, B.Vucetici, „*A List Bidirectional Soft Output Decoder of Turbo Codes",* Intern. Symposium on Turbo Codes, Brest, France, 1997

[HLY02] L.Hanzo, T.H.Liew, B.L.Yeap, „*Turbo Coding,Turbo Equalisation and Space-Time Coding"*, John Wiley & Sons, England, 2002

[HOP96] J. Hagenauer, E.Offer, L. Papke, „*Iterative decoding of Binary Block and Convolutional Codes*", IEEE Trans. Info. Theory, vol. 42, no. 2, Mar. 1996, pp 429- 445.

[HW99] C.Heegard, S.B. Wicker, „*Turbo Coding",* Kluwer Academic Publishers,1999

[JBDP97] M.Jezequel, C.Berrou, C.Douillard, P.Penard, „*Characteristics of a Sixteen-State Turbo-Encoder/Decoder (Turbo 4)", ",* Intern. Symposium on Turbo Codes, Brest, France, 1997

[JH95] J.Hagenauer, „*Source Controlled Channel Decoding",* Ieee Trans. Communic., vol. 43, pp.2449-2457, Sept. 1995

[JH97] J.Hagenauer, „*The Turbo Principle: Tutorial Introduction and State of the Art”,* Intern. Symposium on Turbo Codes, Brest, France, 1997
[KMKM00] M. Kesal, M. K. Mihcak, R. Koetter, and P. Moulin, *"Iteratively decodable codes for watermarking applications,"* in Proc. 2nd Int. Symp. on Turbo Codes and Related Topics, Sept. 2000 , Brest, France
[LYHH93] J.Lodge, R.Young, P.Hoeher, J.Hageanauer, „*Separable MAP Filters for the Decoding of Product and Concatenated Codes”,* Proc.IEEE Intern.Conference on Communic (ICC) Geneva, Switzerland, May 1993, pp. 1740-1745
[MDC02] M.L.Miller, G.J.Doerr, I.J.Cox, „*Dirty-Paper Trellis Codes for Watermarking*”, IEEE Int.Conf.on Image Processing,vol II,pp 129-132, 2002
[PL02] Patrick Loo, „*Digital Watermarking using Complex Wavelets*”, Ph.D. Thesis., March 2002, Univ. of Cambridge,USA
[PRO00] J.G.Proakis, „*Digital Communications”,* editia4-a, New-York, Edit. Mc Graw-Hill, 2000
[PSC96] L.C.Perez, J. Seghers, D.J.Costello, „*A Distance Spectrum Interpretation of Turbo-Codes”,* IEEE Trans. Inform. Theory, vol. 42, pp. 1698-1709, Nov. 1996
[SKL97] B.Sklar, „*A Primer on Turbo Code Concepts*”, IEEE Communic. Magazine, pp. 94-102, Dec. 1997
[TC00] O.Y.Takeshita, D.J.Costello Jr., „*New Deterministic Interleaver Design for Turbo Codes”,* IEEE Trans. Info. Theory, vol. 46, no. 6, Sept. 2000, pp.1988-2006
[THI93] P. Thitimajshima, “Les codes Convolutifs Récursifs Systématiques et leur application à la concaténation parallèle”, These, 21.12.1993
[VAJ95] Viterbi A.J, „*CDMA. Principles of Spread-Spectrum Communications”*, Addisson Wesley Wireless Communic.,1995
[VAL96] M.C.Valenti, „*An Introduction to Turbo Codes”* , http://www.csee.wvu.edu/~mvalenti/documents/valenti1996.pdf , May 1996
[VY00] B.Vucetici, J.Yuan, „*Turbo Codes. Principles and Applications”,* Kluwer Academic Publishers, 2000
[WH00] J.Woodard, L.Hanzo, „*Comparative Study of Turbo Decoding Techniques: An Overview”* , IEEE Trans. Vehic. Techn.,vol. 49, no. 6, Nov. 2000