UNIVERZITET U BEOGRADU

Srđan Ž. Verbić

Veza minimizacije slobodne energije, skrivenih Markovljevih lanaca i dekodovanja linearnih blok kodova

magistarska teza
(tekst teze ima 52 strane)

Beograd, 2000.

Veza minimizacije slobodne energije, skrivenih Markovljevih lanaca i dekodovanja linearnih blok kodova

APSTRAKT

Postoje dve osnovne klase algoritama za dekodovanje linearnih blok kodova: algoritmi koji minimizuju verovatnoću greške dekodovanja kodnih reči i algoritmi koji minimizuju verovatnoću greške dekodovanja za pojedinačne njene bitove. BCJR algoritam, (Bahl, Cocke, Jelinek, Raviv 1974), koji se zasniva na skrivenim Markovljevim lancima, spada u drugu pomenutu klasu. On daje tačne vrednosti za *a posteriori* verovatnoće pojedinačnih bitova, ali ga eksponencijalna složenost $O(2^{n-k})$ čini praktično neprimenljivim za linearne blok kodove sa većom dužinom kodne reči (n), odnosno većim brojem bitova provere parnosti (k). Sa druge strane, aproksimativni FEM algoritam (MacKay 1993), zasnovan na minimizaciji varijacione slobodne enegije, ima linearnu složenost $O(n)$ i često se upotrebljava u problemima dekodovanja linearnih blok kodova sa velikim k i n. Osnovni zadatak rada je poređenje ova dva algoritma i nalaženje uslova pod kojim je moguće izvođenje FEM kao specijalnog slučaja optimalnog BCJR algoritma.

Glavni rezultat teze je ostvaren u okviru poređenja BCJR algoritma i FEM algoritma u primeni na problem dekodovanja linearnog blok koda zadatog preko matrice provere parnosti. Ovaj problem se, takođe, javlja i u kriptoanalizi sekvencijalnih šifarskih sistema zasnovanih na linearnim pomeračkim registrima. Uveden je BCJR algoritam na redukovanom trelisu i pokazano je da su "*forward*" i "*backward*" rekurzije koje se javljaju u jednoj iteraciji FEM algoritma, za jednu vrstu matrice provere parnosti, ekvivalentne sa BCJR algoritmom na redukovanom trelisu za istu vrstu matrice. Saglasno tome, jedna iteracija FEM algoritma je ekvivalentna sa kompozitnom varijantom algoritma koji se sastoji od BCJR algoritama na redukovanim trelisima i čiji su izlazi na odgovarajući način agregirani. Time je verifikovana osnovna hipoteza ovoga rada da se FEM algoritam može dobiti pojednostavljenjem BCJR algoritma.

Predložen je i iterativni BCJR (IBCJR) algoritam u kojem se koristi određena funkcija agregiranja izlaza BCJR algoritama na redukovanim trelisima koja se razlikuje od one koja odgovara FEM algoritmu, čime se izbegava uvođenje virtuelnog šuma. Najzad, numeričkom simulacijom je ispitivan i uticaj virtuelnog šuma na uspešnost FEM algoritma i ustanovljen je optimalan nivo šuma za različite tipove linearnih blok kodova.

Ključne reči: BCJR algoritam, varijaciona minimizacija slobodne energije, skriveni Markovljevi lanci, složenost algoritama za dekodovanje, linearni pomerački registar, redukovani trelis, *forward-backward* algoritmi

A Connection between Free Energy Minimization, Hidden Markov Chains and Decoding of Linear Block Codes

ABSTRACT

There are two main classes of algorithms for decoding of linear block codes: algorithms that minimize probability of codewords' errors and algorithms that minimize probability of decoding errors of individual bits. BCJR algorithm (Bahl, Cocke, Jelinek, Raviv 1974), based on hidden Markov chains, belongs to the second mentioned class. That algorithm gives exact values for *a posteriori* probabilities of individual bits, but it's exponential complexity $O(2^{n-k})$ makes it practically impossible to apply that algorithm to linear block codes with greater length of codeword (n), i.e. greater number of parity check bits (k). On the other hand, approximate FEM algorithm (MacKay 1993), based on variational free energy, has linear complexity $O(n)$ and it is often used in problems of decoding of linear block codes with great k and n. Main task of this thesis is a comparison of those two algorithms and finding of conditions that allows inferring of FEM algorithm as a special case of optimal BCJR algorithm.

The prime result of this thesis is accomplished through comparison of BCJR and FEM algorithms on the problem of decoding linear block codes given by parity check matrix. This problem, also, arises in cryptanalysis of sequential cipher systems based on linear feedback shift registers. BCJR algorithm defined on reduced trellis is introduced. It is exhibited that "forward" and "backward" recursions, which appears in a single iteration of FEM algorithm for a single row of parity check matrix, are equivalent with BCJR algorithm on reduced trellis for the same row of matrix. According to that, a single iteration of FEM algorithm is equivalent to composite variant of algorithm that consists of BCJR algorithms on reduced trellises whose outputs are aggregated in a certain manner. This way main hypothesis of this work is verified, FEM algorithm can be inferred as an approximation of BCJR algorithm.

Iterative BCJR algorithm (IBCJR) is proposed. Aggregation function used for that algorithm differs from that used for FEM and that way we avoid introduction of virtual noise, immanent to FEM algorithm. Finally, influence of virtual noise to success of FEM algorithm is examined and optimal level of such noise is determined for different types of linear block codes.

Key words: BCJR algorithm, variational minimization of free energy, hidden Markov chains, complexity of decoding algorithms, linear feedback shift register, reduced trellis, forward-backward algorithms

Mentori:

prof. dr Zoran Ivković
Matematički fakultet, Beograd

prof. dr Milan Knežević
Fizički fakultet, Beograd

Članovi Komisije:

prof. dr Zoran Ivković
Matematički fakultet, Beograd

prof. dr Milan Knežević
Fizički fakultet, Beograd

prof. dr Žarko Mijajlović
Matematički fakultet, Beograd

Fiz. fak. 18.04. 2001.

1. IBJCR - eksp. slozenost?
2. BJCR - NP kompletan? Trelis - rešetke
3. FEM = Aniling metod!
4. Golic
5. Lenodovati: napad (dakle bez uljuđa) na kodovanu reč (informaciju)

Ovaj rad sam uradio pod budnim okom profesora Jovana Golića. Divim se njegovoj intuiciji. Rezultat koji sam dobio je upravo ono što smo očekivali sve vreme, a da nismo imali pojma kako se do toga dolazi. Razlozi zašto on nije u Komisiji za odbranu ove teze nemaju nikakve veze sa naukom. Možda je ovo način da mu se zahvalim za ono što je uradio.

SADRŽAJ

Uvod

Fundamentalni problem komunikacije je upravo onaj koji se tiče mogućnosti da tačno ili barem približno reprodukujemo poruke primljene sa nekog drugog mesta.

Klod Šenon, 1948

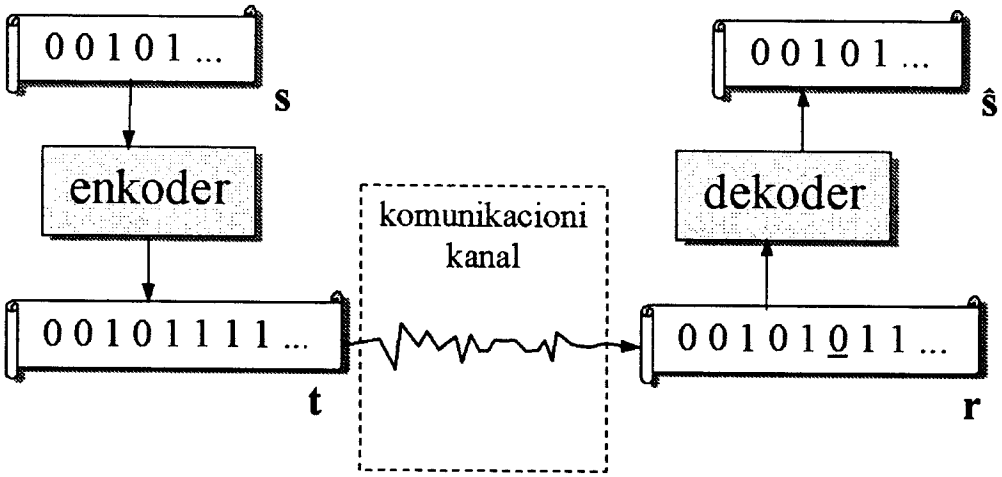
U zavisnosti od konkretne primene, razlikujemo dve osnovne klase algoritama za dekodovanje: algoritme koji kao izlaz daju najverovatniju transmitovanu kodnu reč i algoritme koji daju verovatnoće da je svaki od bitova transmitovane kodne reči jedinica. I u jednom i u drugom slučaju algoritam nastoji da maksimizuje relevantne *a posteriori* verovatnoće. Problem dekodovanja je NP kompletan i to znači da optimalne varijante algoritama obe pomenute klase imaju eksponencijalnu složenost. Takvi algoritmi, naravno, nisu praktični. Nas interesuju, pre svega, dovoljno dobre aproksimacije, odnosno odgovarajući algoritmi polinomijalne složenosti.

Algoritme za dekodovanje razlikujemo i u odnosu na njima svojstvene kodove. Tu, takođe, postoje dve osnovne klase. U jednu klasu spadaju algoritmi koji dekoduju konvolucione, a u drugu blok, kodove. Ova razlika je izraženija što su algoritmi praktičniji. Neki optimalni algoritmi, kao što je BCJR, mogu biti primenjeni na obe klase kodova. Nažalost, u konkretnim implementacijama ova opštost nestaje jer se "ekonomični" algoritmi zasnivaju baš na specifičnostima samog koda. Razlike između konvolucionih i blok kodova se najbolje mogu videti kroz njihove grafičke reprezentacije, trelise samih kodova. Na prvi pogled je jasno da trelis konvolucionog koda ima translacionu simetriju koja olakšava bilo kakva izračunavanja parametara trelisa, pa samim tim i koda. Ova osobina konvolucionih kodova je naročito dobro iskorišćena kod Viterbi algoritma. U slučaju linearnih blok kodova te simetrije nema, što nužno usložnjava izračunavanja. To, međutim, nije dovoljan razlog da u praksi koristimo samo konvolucione kodove. Najbolji kodovi koje danas poznajemo su turbo kodovi i kodovi sa retkim matricama provere parnosti. Ovi prvi se baziraju na konvolucionim, dok drugi spadaju u klasu blok kodova. Bilo kako bilo, iako smo sve bliže Šenonovom limitu, kodovi mogu biti još bolji. Koje kodove i algoritme koristiti, zavisi pre svega od prirode samog signala i od osobina komunikacionog kanala.

U ovom radu su analizirane neke osobine algoritama koji daju verovatnoće pojedinačnih bitova transmitovane kodne reči i koji se primenjuju, pre svega, na blok kodove. Prvi od razmatranih algoritama je BCJR, optimalni algoritam eksponencijalne složenosti. Drugi je varijaciona minimizacija slobodne energije, MekKejev algoritam skoro linearne složenosti baziran na analogiji sa čuvenim Fejnmanovim algoritmom prvi put primenjenim u statističkoj mehanici. Poređenje ta dva algoritma nameće osnovni zadatak ovog rada - Da li je, i pod kojim uslovima, moguće MekKejev algoritam posmatrati kao uprošćenje optimalnog BCJR algoritma?

1. KOMUNIKACIONI SISTEM

Formalno govoreći, svaki komunikacioni sistem se sastoji iz istih elemenata: od izvora informacija, od pošiljaoca koji poruku prevodi u oblik pogodan za slanje, od komunikacionog kanala koji predstavlja medijum za prenos poruke, od primaoca koji iz dobijenog signala rekonstruiše originalnu poruku i od odredišta kome je poruka namenjena (Shannon 1948). Na slici 1.1. je dat shematski prikaz diskretnog komunikacionog sistema. Izvor informacija originalnu poruku (**s**) predaje pošiljaocu (enkoderu) koji u poruku ugrađuje kontrolne bitove i takav signal (**t**) šalje kroz komunikacioni kanal. Signal trpi uticaj šuma, što ima za posledicu da neki od bitova budu promenjeni. Primalac (dekoder) dobija šumom izmenjeni signal (**r**) na osnovu koga rekonstruiše originalnu poruku (**ŝ**). Naravno, cilj je da **s** i **ŝ** budu identični, ali to nije uvek moguće.



Slika 1.1. - Shematski prikaz diskretnog komunikacionog sistema

Pošiljalac i primalac, odnosno odredište, ne moraju nužno biti na različitim mestima u prostoru. Primeri za to su, recimo, Kamen iz Rozete ili hard disk. Tamo gde je poruka zapisana, tamo je i čitamo. Za razliku od "stereotipnog" komunikacionog kanala gde se poruka prenosi kroz prostor, kao u slučaju telefonske linije ili komunikacije svemirske sonde i baze na Zemlji, ovaj drugi vid kanala bi trebalo da očuva poruku u vremenu. U skladu sa ovim, "mesta" iz čuvene Šenonove sentence valja shvatiti kao tačke u prostor-vremenu, nešto opštijem medijumu same komunikacije.

Bez obzira na to kako izgleda komunikacioni kanal, on sigurno sadrži elemente šuma koji manje ili više menjaju sadržaj poruke. Najčešće je izvor tog šuma u prirodnom fonu sredine kroz koju šaljemo signal. U slučaju poruka sa Vojadžera ili Kazinija, to je elektromagnetno polje Sunca i planeta Sunčevog sistema. Kod Kamena iz Rozete, to je erozija, dok je kod poruka relevantnih za kriptologiju to, ponekad, čovek.

U svim ovim slučajevima, ako kroz kanal šaljemo poruku, postoji verovatnoća da će informacija koju primimo na izlazu komunikacionog kanala biti različita od one koju smo poslali. Ovu neprijatnu osobinu komunikacionog sistema možemo ublažiti na dva načina. Prvi je da pojačamo signal ili uzmemo "veću antenu". Ovo je, obično, skuplja varijanta. Drugi način je da nađemo bolji kod koji

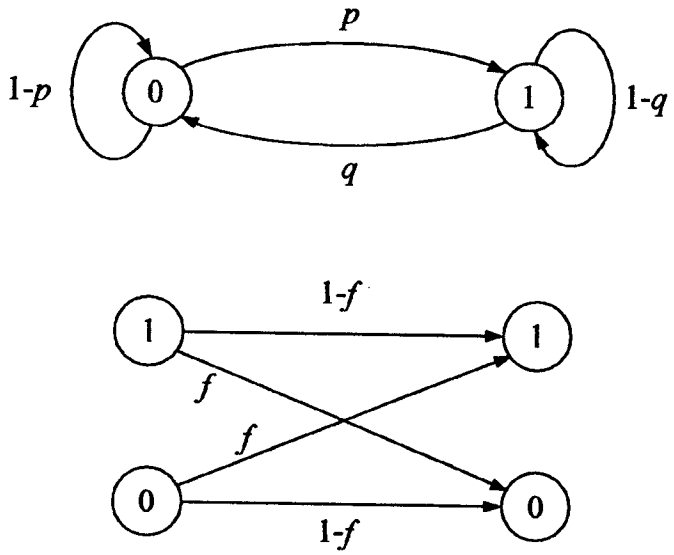
nam omogućava da rekonstruišemo početni signal sa većom pouzdanošću za isti nivo šuma. Možda toga nismo svesni, ali mi u svakodnevnom životu koristimo i jedno i drugo.

Kodiranje predstavlja svesno uvođenje redundantne informacije koja će nam omogućiti da rekonstruišemo originalnu poruku. Moglo bi se reći da mi na ovaj način kodiramo sve poruke u svakodnevnoj komunikaciji upravo sa ciljem da nas sagovornik što bolje razume. Gestikulacija, na primer, često izgleda suvišna, ali ona značajno povećava šanse da onaj kome je poruka upućena shvati šta smo zapravo hteli da kažemo. U pisanoj komunikaciji, takođe, postoji redundansa. Svi tekstovi pisani na prirodnim jezicima predstavljaju prilično "neekonomičan" zapis. Za srpski jezik redundansa je oko 85%. To znači da bi ovaj magistarski rad mogao biti "spakovan" i na šest puta manje strana, a da opet sve informacije budu sačuvane. Naravno, takav tekst bi bilo drastično teže čitati. Drugim rečima, *mgl b d zstvm sv smglsnk, td b tkst b dpl krć, l b, tkđ, b mng mnj čtlju.*

Kada je u pitanju diskretni komunikacioni sistem, redundansu je najlakše uvesti korišćenjem bitova provere parnosti. To su bitovi kojim originalnu poruku (**s**) dopunjujemo do sekvence koja se transmituje kroz komunikacioni kanal (**t**). Svaki od tih bitova predstavlja sumu po modulu 2 tačno određenih bitova originalne poruke. Bitovi provere parnosti postoje i kod konvolucionih i kod linearnih blok kodova. Razlika je samo u načinu na koji se ti bitovi dodaju originalnoj poruci.

1.1. Šum u komunikacionom kanalu

U ovom radu je razmatran samo diskretni kanal bez pamćenja (DMC). Takav komunikacioni kanal karakteriše nezavisno delovanje šuma na sve bitove transmitovane sekvence. U najjednostavnijem slučaju DMC-a, binarno simetričnom kanalu (BSC) postoji samo jedna veličina koja određuje verovatnoću da bit bude primljen u svom promenjenom stanju, nivo šuma f. Nešto složeniji slučaj kanala koji nije simetričan razlikuje promenu bita iz nule u jedinicu i iz jedinice u nulu. Na slici 1.2. su dati grafički prikazi obe vrste DMC-a. Oznake uz strelice predstavljaju verovatnoće da bitovi na izlazu iz kanala budu promenjeni.



Slika 1.2. - Grafički prikaz DMC-a u opštem slučaju i u slučaju BSC kanala

2. LINEARNI BLOK KODOVI

Porukom nazivamo uređenu k-torku bitova $\mathbf{s}=(s_1, s_2,..., s_k)$ dobijenu od određenog izvora informacija. Kako svaki bit može uzeti vrednosti nule ili jedinice, postoji ukupno 2^k različitih vektora **s**. Enkoder transformiše poruku u n-torku $\mathbf{t}=(t_1, t_2,..., t_n)$ koja predstavlja kodnu reč. Kod velike većine kodova koje danas upotrebljavamo n je veće od k, te se na taj način uvodi redundansa. Taj "višak" od $n-k$ bitova u kodnoj reči zavisi od stanja k bitova poruke, tako da je broj različitih kodnih reči 2^k kao u i slučaju poruka. Ovaj skup od 2^k kodnih reči dužine n zovemo (n,k) blok kod. Ukoliko je suma po modulu 2 bilo koje dve kodne reči kodna reč iz istog koda onda takav blok kod nazivamo linearnim. Odnos $R=k/n$ nazivamo brzinom prenosa informacija.

Najčešće korišćeni primer blok koda je Hemingov kod (7,4). Odgovarajući enkoder uzima sekvence dužine četiri bita i njima pridružuje tri bita provere parnosti. Skupu od 2^4 poruka odgovara isti toliki broj kodnih reči, odnosno Hemingov kod.

0	0	0	0	→	0	0	0	0	0	0	0
0	0	0	1	→	1	0	1	0	0	0	1
0	0	1	0	→	1	1	1	0	0	1	0
0	0	1	1	→	0	1	0	0	0	1	1
0	1	0	0	→	0	1	1	0	1	0	0
0	1	0	1	→	1	1	0	0	1	0	1
0	1	1	0	→	1	0	0	0	1	1	0
0	1	1	1	→	0	0	1	0	1	1	1
1	0	0	0	→	1	1	0	1	0	0	0
1	0	0	1	→	0	1	1	1	0	0	1
1	0	1	0	→	0	0	1	1	0	1	0
1	**0**	**1**	**1**	→	**1**	**0**	**0**	**1**	**0**	**1**	**1**
1	1	0	0	→	1	0	1	1	1	0	0
1	1	0	1	→	0	0	0	1	1	0	1
1	1	1	0	→	0	1	0	1	1	1	0
1	1	1	1	→	1	1	1	1	1	1	1

Tabela 1.1. - Poruke dužine 4 i njima odgovarajuće kodne reči

U tabeli 1.1. su date sve moguće poruke dužine 4 i njima odgovarajuće kodne reči. Treba napomenuti da kodne reči nisu jednoznačno određene za ovaj skup poruka. Postoji nekoliko načina na koji se jednoj poruci može pridružiti kodna reč, odnosno tačno onoliko načina na koliko možemo formirati generatorsku matricu **G**. U slučaju koda datog u tabeli 1.1. korišćena je matrica

$$\mathbf{G}=\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\1&1&1&0&0&1&0\\1&0&1&0&0&0&1\end{bmatrix}.$$

Tako poruci **s**=[1 0 1 1] odgovara kodna reč **t**=**s**·**G**, tj. [1 0 0 1 0 1 1]. Interesantno je da su poslednja četiri bita kodne reči, u ovom našem slučaju, jednaki samoj poruci.

To nije karakteristika samog koda već konkretno izabrane generatorske matrice. Za matrice koje imaju osobinu da u kodnoj reči razdvoje samu poruku od bitova provere parnosti kažemo da su date u sistematskoj formi.

Svakoj generatorskoj matrici **G** možemo pridružiti odgovarajuću matricu provere parnosti **H** sa osobinom $\mathbf{G} \cdot \mathbf{H}^T = 0$. Algoritmi za dekodovanje se mogu bazirati na bilo kojoj od ove dve matrice.

Kad su u pitanju blok kodovi, kodovanje, očigledno, ima linearnu složenost. Postupak u suprotnom smeru, tj. dekodovanje, međutim, predstavlja daleko veći problem. Kao i kod drugih kombinatornih problema jedan praktičan pristup rešavanju sastoji se u sekvencijalnom traženju kodne reči sa najvećom *a posteriori* verovatnoćom. To u slučaju Hemingovog koda znači da bi trebalo "proći" kroz sve kodne reči i za svaku naći verovatnoću da na izlazu komunikacionog kanala dobije primljenu kodnu reč, **r**. Složenost optimalnog dekodovanja će, u svakom slučaju, biti proporcionalna sa 2^k.

Kod dekodovanja linearnih blok kodova značajnu pomoć predstavlja činjenica da znamo raspodelu težina za kodne reči i da nam je poznato minimalno Hemingovo rastojanje između njih. Nažalost, za veliku većinu linearnih kodova još uvek ne znamo kako izgleda njihova raspodela težina, što drastično otežava dekodovanje (Lin, Costello 1983).

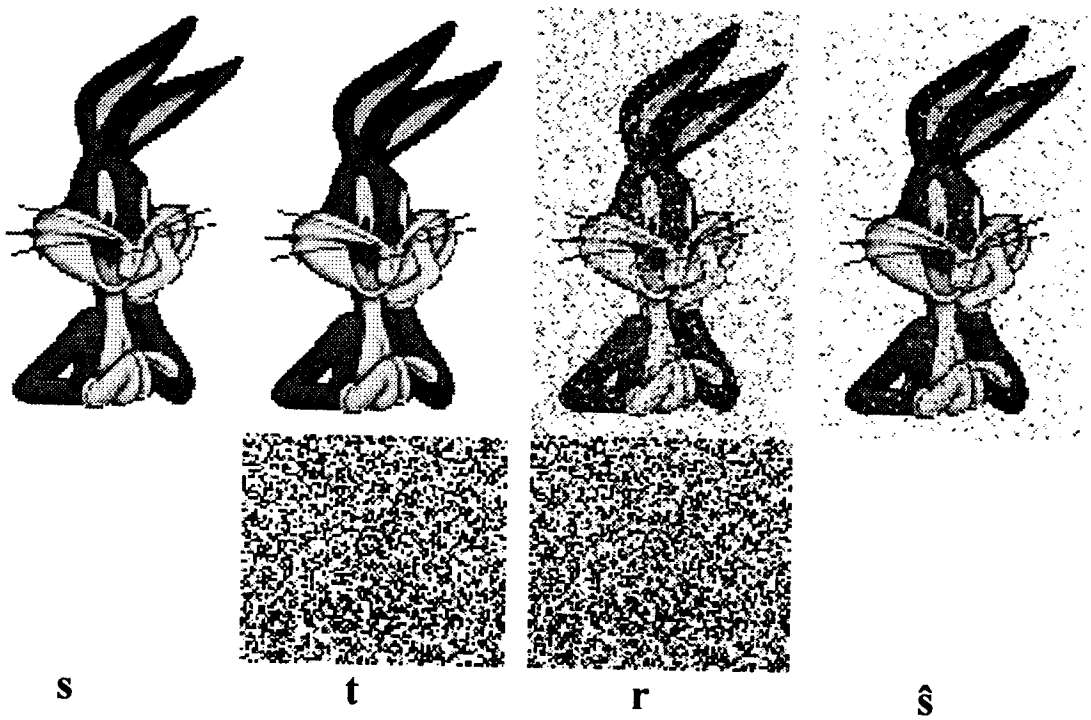
Težinu neke sekvence bitova definišemo kao broj jedinica koje ona sadrži. Hemingovo rastojanje između dve sekvence je težina njihove razlike po modulu 2. Za Hemingov kod (7,4) minimalno rastojanje je tri bita, što nam omogućava da korektno rekonstruišemo originalnu poruku ako je šum promenio bilo koji, ali tačno jedan bit transmitovane kodne reči. Drugim rečima, ako je razlika između vektora **r** i **t** samo jedan bit, Hemingovo rastojanje između njih je jedan. Međutim, postoje tačno tri kodne reči za koje je **r** na Hemingovom rastojanju dva, četiri na rastojanju tri, itd. Ako pretpostavimo da je nivo šuma dovoljno mali, onda je najverovatnije da je promenjen samo jedan bit, pa za najverovatniju kodnu reč možemo uzeti onu koja je na rastojanju jedan. Sa druge strane, da bi našli verovatnoće da su pojedinačni bitovi jedinice u obzir moramo uzeti i kodne reči koje se nalaze na nešto većem Hemingovom rastojanju. Ovo je bitna razlika između dekodera koji maksimizuju *a posteriori* verovatnoće za kodne reči i pojedinačne simbole.

Sindrom $\boldsymbol{\sigma}$ definišemo kao proizvod primljenog signala i transponovane matrice provere parnosti **H**, tj. $\boldsymbol{\sigma} = \mathbf{r} \cdot \mathbf{H}^T$. Vektor $\boldsymbol{\sigma}$ ima onoliko bitova koliko i provera parnosti, tj. n-k. Ako su svi ti bitovi nule onda je primljeni signal neka od kodnih reči[1]. Obrnuto, takođe, važi - ako je primljeni signal kodna reč, onda je sindrom sigurno nulti vektor. U slučaju Hemingovog koda (7,4) možemo ispraviti svih sedam različitih grešaka gde je promenjen tačno jedan bit kodne reči. Svaka od ovih grešaka ima različit sindrom i sve njih možemo prepoznati u vrstama matrice $\mathbf{H}^T$. Ako je promenjen samo m-ti bit kodne reči, onda je sindrom upravo m-ta vrsta transponovane matrice provere parnosti. To znači da je dovoljno memorisati $2^{n-k}-1$ različitih sindroma, pa da otkrijemo na kom je bitu došlo do greške, a samim tim i da rekonstruišemo početnu poruku. Nažalost, kod dužih blok kodova traženje greške na osnovu sindroma postaje nemoguće, jer $2^{n-k}-1$ vektora dužine n-k više ne mogu da stanu u memoriju računara.

---

[1] Nažalost, to ne mora biti ista kodna reč koja je i transmitovana. U slučaju Hemingovog (7,4) koda, ako je šum pokvario tri ili čak šest bitova sindrom će opet biti nula. Naravno, za razumno nizak nivo šuma ovo je malo verovatno. Konkretno za BSC kanal i f=0.1 verovatnoća višestruke greške na jednoj kodnoj reči iznosi 15%.

Na slici 1.3. je dat primer četiri osnovne sekvence komunikacionog sistema: originalne poruke **s** (slika u BMP formatu), kodovanog signala **t**, primljene sekvence **r** i, koliko je to moguće, dobro rekonstruisane poruke $\hat{\mathbf{s}}$. Slika je transmitovana kroz BSC kanal sa šumom nivoa f=0.1 korišćenjem Hemingovog (7,4) koda. Verovatnoća grešaka na dekodovanim bitovima je oko 7%.



Slika 1.3. - Primer četiri osnovne sekvence komunikacionog sistema

3. GRANICE PRIMENLJIVOSTI KODOVA I ALGORITAMA ZA DEKODOVANJE

U praksi, nesumnjivo, postoji kompromis između realizovane verovatnoće greške dekodovane kodne reči, koju nastojimo da smanjimo, i brzine prenosa informacija R, za koju želimo da bude što je moguće veća. Postavlja se ključno pitanje - Koja je maksimalno moguća brzina prenosa za očekivanu i podnošljivo malu verovatnoću greške? Odgovor na ovo pitanje je dao Klod Šenon u svom čuvenom radu iz 1948. godine formulišući fundamentalnu teoremu teorije informacija - teoremu o kodovanju u zašumljenom kanalu. Neočekivani rezultat te teoreme kaže da je za svaki komunikacioni kanal moguće konstruisati kod sa nenultom brzinom odašiljanja R i pri čemu je moguće dostići proizvoljno malu verovatnoću greške. Najveća moguća brzina odašiljanja je pri tom određena kapacitetom samog kanala:

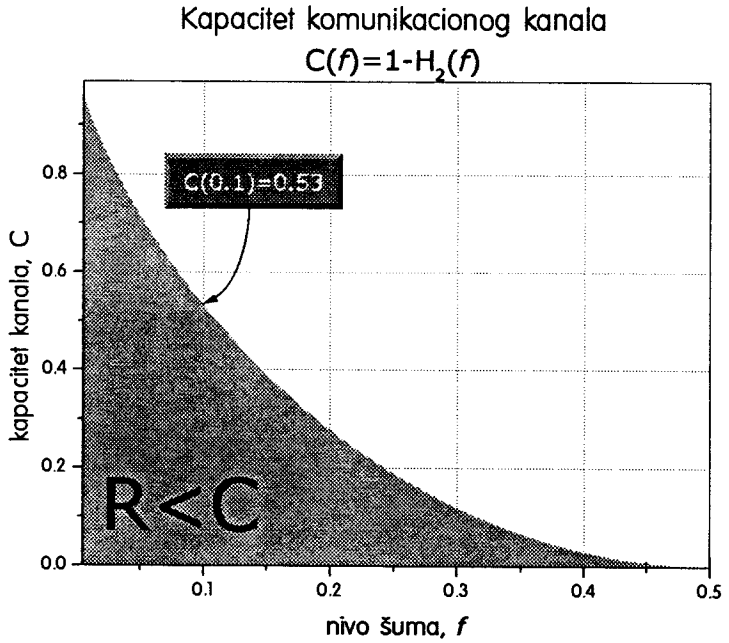
$$C(f) = 1 - H_2(f).$$

H_2 u prethodnoj formuli označava informacionu entropiju samog šuma koja se računa kao

$$H_2 \equiv f \cdot \log_2 \frac{1}{f} + (1-f) \cdot \log_2 \frac{1}{1-f}.$$

Komunikacija je moguća i na brzinama R>C, ali tada ne možemo zahtevati proizvoljno malu verovatnoću greške dekodovanog bita.

Na slici 1.4. je data zavisnost kapaciteta komunikacionog kanala C od nivoa šuma u njemu f. U oblasti $R \leq C$, osenčenoj na slici, možemo očekivati kodove sa proizvoljno malom verovatnoćom greške.



Slika 1.4. - Kapacitet komunikacionog kanala

Bitno je napomenuti da to što znamo da postoje "dobri" kodovi za jako veliko R ne znači da smo ih i otkrili. Danas znamo za dosta kodova koji se po svojim karakteristikama približavaju Šenonovom limitu, ali još uvek ima prostora i za bitno bolje kodove. Drugo, čak i kada bismo znali kako da ih konstruišemo, niko ne tvrdi da bismo bili u stanju da ih primenimo. U principu, za blok kodove važi da što je željena verovatnoća greške manja, to dužina koda mora biti veća.

Očigledno, postoji prilično velika razlika između onoga što je moguće i onoga što znamo da konstruišemo. Poseban je problem kako u praksi primeniti neki od tih "jako dobrih" kodova. Sa Hemingovim (7,4) kodom nema naročitih problema jer ni kodovanje ni dekodovanje ne iziskuju obimna izračunavanja. Kod iole dužih blok kodova to može biti i nepremostiv problem. Zbog toga u praksi koristimo razne tipove aproksimativnih algoritama za dekodovanje koji imaju svoje dobro definisane domene primenljivosti. Nesumnjivo, bez obzira na to koliko sam kod bio dobar, uspeh u rekonstrusanju originalne poruke pre svega zavisi od performansi konkretnog algoritma za dekodovanje.

Optimalno dekodovanje linearnih blok kodova za minimizaciju verovatnoće greške bitova

Za razliku od Viterbi algoritma koji optimizuje dekodovanje po verovatnoći kodne reči, BCJR algoritam (Bahl *et al.* 1974) daje rešenje sa minimalnim greškama pojedinačnih bitova. Ovo, drugim rečima, znači da kao izlaz dekodera dobijamo n-dimenzionalni vektor sa komponentama koje predstavljaju verovatnoće da su odgovarajući bitovi kodne reči jednaki jedinici. Simbol n ovde označava dužinu kodne reči.

Algoritam se zasniva na formiranju trelisa[1] karakterističnog za sam kod i propagaciji verovatnoće sekvenci kodne reči kroz njegova unapred definisana stanja. Svaka sekvenca može biti u jednom od 2^l stanja trelisa, pri čemu l predstavlja memoriju u slučaju konvolucionih ili broj bitova provere parnosti u slučaju blok kodova. Linearni kodovi se na taj način mogu predstaviti odgovarajućim trelisima pri čemu je za svaku dužinu sekvence definisano 2^l stanja. U slučaju transmitovanog koda, tj. izvora, očigledno se pretpostavlja da vrednosti pojedinih bitova zavise od prethodnih elemenata niza. BCJR algoritam se zasniva baš na pretpostavci da umesto niza bitova radimo sa nizom stanja gde ne postoji efekat pamćenja, odnosno Markovljevom svojstvu da je sva prethodna istorija sadržana u jednom stanju.

1. BCJR ALGORITAM

Pretpostavimo da je izvor vremenski diskretan Markovljev proces konačnih stanja. U tom slučaju imamo $M=2^l$ različitih stanja Markovljevog izvora, $m=0, 1, ..., M-1$. Stanje izvora u trenutku t označavamo sa S_t, a odgovarajući izlaz sa X_t. Sekvence stanja i izlaza označavamo sa $\mathbf{S}_t^{t'} = S_t, S_{t+1}, ..., S_{t'}$, odnosno $\mathbf{X}_t^{t'} = X_t, X_{t+1}, ..., X_{t'}$.

Prelaze između stanja karakterišu verovatnoće prelaza

$$p_t(m|m') = \Pr\{S_t = m | S_{t-1} = m'\}$$

i verovatnoće izlaza[2]

$$q_t(X|m', m) = \Pr\{X_t = X | S_{t-1} = m'; S_t = m\}.$$

Markovljev izvor počinje od stanja $S_0=0$ i proizvodi izlaznu sekvencu $\mathbf{X}_1^n$ završavajući se u konačnom stanju $S_n=0$. $\mathbf{X}_1^n$ predstavlja ulaznu sekvencu zašumljenog

[1] Trelis je grafička interpretacija strukture koju dobijamo za Markovljev lanac. Struktura sadrži čvorove određene indeksima stanja m i vremenskim indeksima t. Čvorovi predstavljaju stanja Markovljevog lanca, a veze između njih nenulte verovatnoće prelaza. Graf koji dobijamo povezivanjem svih definisanih čvorova zovemo trelisom koda. Vremenski indeks govori o redosledu bitova koji izlaze iz enkodera i tako definiše smer Markovljevog lanca. Za svaku sekvencu simbola $\mathbf{S}_0^n$ postoji jedinstvena putanja kroz trelis. Obrnuto, takođe, važi.

[2] Verovatnoća izlaza ustvari označava verovatnoću da je odgovarajući bit transmitovanog signala nula, odnosno jedinica ako su stanja m i m' fiksirana.

diskretnog kanala bez pamćenja (DMC) čiji izlaz označavamo sa $\mathbf{Y}_1^n$. Verovatnoću prelaza za DMC definišemo kao proizvod verovatnoća prelaza za pojedinačne bitove:

$$\Pr\{Y_1^t \big| X_1^t\} = \prod_{j=1}^{t} \mathrm{R}(Y_j, X_j),$$

gde je $1 \le t \le n$. Zadatak dekodera je da ispita sve izlazne sekvence DMC-a i proceni *a posteriori* verovatnoće (APP) stanja i prelaza u Markovljevom izvoru, tj. uslovne verovatnoće

$$P_t(m) = \Pr\{S_t = m \big| \mathbf{Y}_1^n\} = \Pr\{S_t = m; \mathbf{Y}_1^n\} / \Pr\{\mathbf{Y}_1^n\} \text{ i}$$

$$\Pr\{S_{t-1} = m'; S_t = m \big| \mathbf{Y}_1^n\} = \Pr\{S_{t-1} = m'; S_t = m; \mathbf{Y}_1^n\} / \Pr\{\mathbf{Y}_1^n\}.$$

Da bismo našli ove uslovne verovatnoće, naročito $P_t(m)$, potrebno je definisati neke pomoćne veličine:

$$\alpha_t(m) = \Pr\{S_t = m; \mathbf{Y}_1^t\}, \qquad \beta_t(m) = \Pr\{\mathbf{Y}_{t+1}^n \big| S_t = m\}.$$

Veličina $\alpha_t(m)$ predstavlja združenu verovatnoću određenog stanja izvora m u trenutku t i dela izvorne sekvence $\mathbf{Y}_1^t$. Jasno, t je tekući indeks i uvek je $1 \le t \le n$. $\beta_t(m)$ predstavlja verovatnoću pojavljivanja drugog dela sekvence, od $t+1$ do n, ako je poznato stanje m u trenutku t.

$$\gamma_t(m', m) = \Pr\{S_t = m; Y_t \big| S_{t-1} = m'\}$$

Veličina $\gamma_t(m', m)$ označava združenu verovatnoću stanja m i pojavljivanja simbola Y_t kao izlaza DMC-a u trenutku t, ako je poznato stanje procesa u trenutku $t-1$.

Veze između pomenutih verovatnoća su date relacijama

$$\alpha_t(m) = \sum_{m'=1}^{M} \Pr\{S_{t-1} = m'; S_t = m; \mathbf{Y}_1^t\} = \sum_{m'} \Pr\{S_{t-1} = m'; \mathbf{Y}_1^{t-1}\} \cdot \Pr\{S_t = m; Y_t \big| S_{t-1} = m'\} =$$

$$= \sum_{m'} \alpha_{t-1}(m') \cdot \gamma_t(m', m) \qquad \text{i} \qquad (2.1)$$

$$\beta_t(m) = \sum_{m'=1}^{M} \Pr\{S_{t+1} = m'; \mathbf{Y}_{t+1}^n \big| S_t = m\} = \sum_{m'} \Pr\{S_{t+1} = m'; Y_{t+1} \big| S_t = m\} \cdot \Pr\{\mathbf{Y}_{t+2}^n \big| S_{t+1} = m'\} =$$

$$= \sum_{m'} \beta_{t+1}(m') \cdot \gamma_{t+1}(m, m'), \qquad (2.2)$$

uz početne uslove

$$\alpha_0(1) = 1 \text{ i } \alpha_0(m) = 0 \quad \forall m \ne 1 \text{ i } \beta_n(1) = 1 \text{ i } \beta_n(m) = 0 \quad \forall m \ne 1.$$

U srednjem koraku jednakosti (2.1) i (2.2) smo iskoristili osobinu Markovljevih procesa da događaji nakon trenutka t ne zavise od $\mathbf{Y}_1^t$ ukoliko nam je poznato stanje S_t. Odavde sledi da se α_t i β_t mogu rekurzivno izračunavati, ako znamo odgovarajuće vrednosti γ_t. Na osnovu definicije imamo da je izraz za γ_t

$$\gamma_t(m, m') = \sum_X \Pr\{S_t = m \big| S_{t-1} = m'\} \cdot \Pr\{X_t = X \big| S_{t-1} = m', S_t = m\} \cdot \Pr\{Y_t \big| X\} =$$

$$= \sum_X p_t(m \big| m') \cdot q_t(X \big| m', m) \cdot \mathrm{R}(Y_t, X). \qquad (2.3)$$

Veličine $\alpha_t(m)$ i $\beta_t(m)$ se mogu predstaviti i kao verovatnoće da putanja slučajno izabrane kodne reči prođe baš kroz m-to stanje trelisa sa vremenskim indeksom t. Razlika je u tome što u slučaju $\alpha_t(m)$ polazimo od nultog elementa unapred kroz trelis, dok je za $\beta_t(m)$ obrnuto, od poslednjeg elementa unazad. Ove verovatnoće u opštem slučaju nisu jednake. Ono što nas zapravo interesuje je veličina $\Pr\{S_t = m|\mathbf{Y}_1^n\}$, *a posteriori* verovatnoća određenog stanja trelisa za poznati primljeni signal. Ova verovatnoća se može dobiti iz poznatih $\alpha_t(m)$ i $\beta_t(m)$. Definišimo prvo veličinu

$$\lambda_t(m) = \Pr\{S_t = m; \mathbf{Y}_1^n\}$$

koja predstavlja združenu verovatnoću stanja m i sekvence $\mathbf{Y}_1^n$.

$$\begin{aligned}\lambda_t(m) &= \Pr\{S_t = m; \mathbf{Y}_1^t\} \cdot \Pr\{\mathbf{Y}_{t+1}^n | S_t = m; \mathbf{Y}_1^t\} \\ &= \alpha_t(m) \cdot \Pr\{\mathbf{Y}_{t+1}^n | S_t = m\} = \alpha_t(m) \cdot \beta_t(m)\end{aligned} \tag{2.4}$$

U srednjoj jednakosti je iskorišćena osobina Markovljevih lanaca da ukoliko je poznato S_t, događaji posle vremena t ne zavise od $\mathbf{Y}_1^t$.

Veza sa uslovnom verovatnoćom je

$$\Pr\{S_t = m|\mathbf{Y}_1^n\} = \lambda_t(m)/\Pr\{\mathbf{Y}_1^n\},$$

što ustvari znači da bi veličinu $\lambda_t(m)$ samo trebalo normalizovati. Kako je verovatnoća $\Pr\{\mathbf{Y}_1^n\} = \lambda_n(0)$ dobijamo izraz za tražene verovatnoće svih stanja u trelisu:

$$P_t(m) = \lambda_t(m)/\lambda_n(0) \tag{2.5}$$

Veličina $P_t(m)$ nam daje verovatnoće pojedinih stanja u trelisu. Kako kodna reč dužine n mora "proći" kroz tačno jedno stanje za svaki vremenski indeks $0 \le t \le n$ imamo

$$\sum_{m=0}^{M-1} P_t(m) = 1.$$

2. KONSTRUKCIJA TRELISA ZA LINEARNE BLOK KODOVE

Jedan mogući način za konstrukciju trelisa za linearni blok kod je dat u istom radu gde je predložen i BCJR algoritam (Bahl *et al.* 1974). Ovde se trelis konstruiše korišćenjem matrice provere parnosti odgovarajućeg koda.

Neka je $\mathbf{H}$ matrica provere parnosti linearnog blok koda (n,k) i neka su $\mathbf{h}_i$, i=1, 2, ..., n kolone matrice $\mathbf{H}$. Neka je $\mathbf{C}$=(c_1, c_2, ..., c_n) kodna reč. Stanja S_t koja odgovaraju kodnoj reči, gde je t=0, 1, ..., n, i definišemo kao:

$$S_0 = \mathbf{0} \text{ i } S_t = S_{t-1} + c_t\mathbf{h}_t = \sum_{i=1}^{t} c_i\mathbf{h}_i, \quad t=1, 2, \ldots, n. \tag{2.6}$$

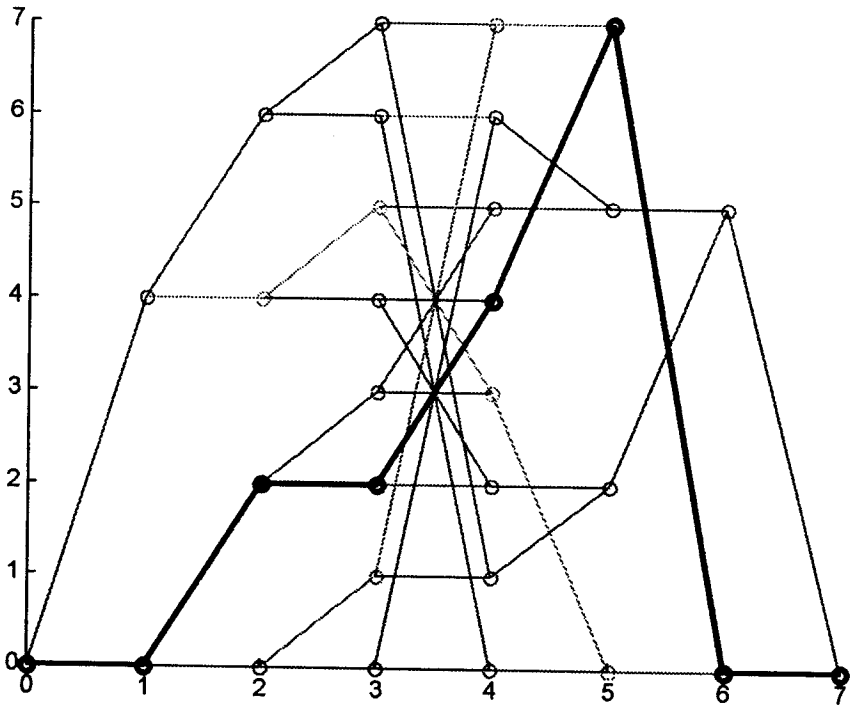
Naravno, ovde sabiranje podrazumeva kongruenciju po modulu 2. Tekuća vrednost stanja S_t je funkcija samo prethodnog stanja S_{t-1} i tekućeg izlaza c_t. S_n=$\mathbf{0}$ za sve kodne reči[3].

[3] Ako je $\mathbf{C}$ kodna reč onda je S_n jednako proizvodu $\mathbf{C}\cdot\mathbf{H}^T$ što je nulti vektor i odgovara stanju 0.

U slučaju Hemingovog (*Hamming*) koda (7,4) imamo generator i odgovarajuću matricu provere parnosti date u sistematskoj formi:

$$\mathbf{G} = \begin{bmatrix} 1\,1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0 \\ 1\,0\,1\,0\,0\,0\,1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1\,0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,1 \end{bmatrix}.$$

Grafički prikaz trelisa Hemingovog koda (7,4) izgleda ovako:



Slika 2.1. - Trelis Hemingovog (7,4) koda

Stanja trelisa na crtežu označena brojevima od 0 do 7 predstavljaju dekadne vrednosti binarnih brojeva zapisanih u vektorima stanja, odnosno kolonama S_r. Podebljana izlomljena linija na grafiku predstavlja jednu od mogućih putanja kroz trelis, odnosno kodnu reč 0101110.

Očigledno, način konstruisanja trelisa nije jedinstven jer ni matrica provere parnosti nije jedinstvena. Linearne kombinacije vrsta u **H** matrici daju drugačiju matricu provere parnosti i samim tim drugačiji trelis.

2. 1. BCJR algoritam i trelis u slučaju binarnog kanala

Kada BCJR algoritam primenimo na binarni komumikacioni kanal, svakom Markovljevom izvoru $\mathbf{S}_0^n$ odgovara izlazna sekvenca enkodera $\mathbf{X}_1^n$, koja predstavlja binarnu kodnu reč. Veza između binarnog koda i odgovarajućeg skupa stanja Markovljevog procesa se ostvaruje konstrukcijom trelisa za sam kod.

Osobina trelisa je da za svaku kodnu reč $\mathbf{X}_1^n$ postoji tačno jedna putanja kroz trelis, tj. jedan niz stanja $\mathbf{S}_0^n$, pri čemu ovaj niz ima početno i krajnje stanje S_0=0 i S_n=0. Postojanje putanje kroz trelis može biti kriterijum da li je primljena sekvenca u komunikaciom kanalu kodna reč ili ne, tj. da li je šum promenio neki od bitova transmitovane poruke. Izlazna sekvenca enkodera $\mathbf{X}_1^n$ je ulaz diskretnog kanala bez pamćenja (DMC). Odgovarajući izlaz DMC-a, odnosno primljeni signal označavamo sa $\mathbf{Y}_1^n$.

Trelisi se mogu konstruisati i za konvolucione i za linearne blok kodove. Kod konvolucionih trelis, barem u svom središnjem delu, ima osobinu translacione simetrije, što omogućava jednostavnije dekodovanje. U oba slučaja BCJR algoritam daje uslovne verovatnoće pojedinačnih stanja u trelisu za datu primljenu sekvencu, odnosno

$$P_t(m) = \Pr\{S_t = m | \mathbf{Y}_1^n\} = \Pr\{S_t = m; \mathbf{Y}_1^n\} / \Pr\{\mathbf{Y}_1^n\}.$$

Ako znamo kako izgleda trelis i ako znamo verovatnoće svih njegovih stanja onda je moguće naći i verovatnoće svih kodnih reči, kao i verovatnoće da su pojedinačni bitovi transmitovane kodne reči jednaki jedinici.

2. 2. Pojednostavljenje algoritma u slučaju BSC šuma

Ako BCJR algoritam primenimo na binarni kod transmitovan kroz binarno simetrični kanal (BSC) formula za verovatnoću (2.3)

$$\gamma_t(m, m') = \sum_X p_t(m|m') \cdot q_t(X|m', m) \cdot \mathrm{R}(Y_t, X)$$

dobija jednostavniju formu. U ovom slučaju sumiranje po X predstavlja sumiranje po svim mogućim izlaznim simbolima, tj. {0, 1}. Funkcija R daje verovatnoću spontane promene simbola usled šuma, što je u slučaju binarno simetričnog kanala dato binomnom raspodelom. U konkretnom slučaju sekvenci od samo jednog simbola, imamo još jednostavniji izraz:

$$\mathrm{R}(Y_t, X) = \begin{cases} p_c, & Y_t \neq X \\ 1 - p_c, & Y_t = X \end{cases},$$

gde p_c predstavlja verovatnoću greške na jednom bitu, odnosno nivo šuma binarno simetričnog kanala. Izraz (2.3) u slučaju binarnog signala za BSC i linearne blok kodove ima oblik:

$$\gamma_t(m,m') = p_t(m|m') \cdot q_t(0|m',m) \cdot \mathrm{R}(Y_t|0) + p_t(m|m') \cdot q_t(1|m',m) \cdot \mathrm{R}(Y_t|1), \qquad (2.7)$$

gde $p_t(m|m')$ uzima vrednost 1 ukoliko postoji kod za koji imamo $S_t=m$ i $S_{t-1}=m'$. U suprotnom p_t uzima vrednost 0. Ako je $m=m'$ onda $q_t(0|m',m)$ ima vrednost 1, dok je $q_t(1|m',m)=0$. Za $m\neq m'$ važi obrnuto[4].

3. IZRAČUNAVANJE VEROVATNOĆA POJEDINAČNIH BITOVA

U originalnom članku o BCJR algoritmu nije dat način kako iz *a posteriori* verovatnoća stanja u trelisu dobiti verovatnoće pojedinih bitova transmitovane kodne reči. Ono što se nameće samo po sebi kao algoritam za ovo izračunavanje podrazumeva prebrojavanje po svim kodnim rečima, određivanje njihovih relativnih verovatnoća i njihovo sumiranje po svim kodnim rečima u kojima je odgovarajući bit jednak jedinici. Ovo znači da iz poznatih vrednosti $P_t(m)$ možemo dobiti ne samo verovatnoće pojedinačnih bitova već i pojedinačnih kodnih reči[5]. Štaviše, u ovoj varijanti izračunavanja, to je neophodno.

Kako ima 2^k kodnih reči onda je za dobijanje svih verovatnoća vreme izračunavanja proporcionalno sa $(n+1)2^k$. Iako je prethodni način dobijanja APP vrednosti za jedinične bitove najjednostavniji za implementaciju, sigurno nije optimalan u pogledu vremenske kompleksnosti. U nastavku je predložen jedan rekurzivni algoritam manje složenosti.

Kako svakoj kodnoj reči $\mathbf{X}$ iz koda χ odgovara tačno jedna putanja kroz trelis $\mathbf{S}$, i kako znamo verovatnoće svih čvorova u njemu $P_t(m)$, umesto verovatnoća celih kodnih reči možemo definisati verovatnoće njihovih sekvenci, tj. parcijalne verovatnoće $pp_t(\mathbf{S}_0^t)$. Ova veličina je verovatnoća da sekvenca stanja $\mathbf{S}_0^t$ odgovara transmitovanom kodu. Ako je $t=n$ onda je $pp_t(\mathbf{S}_0^n) = \Pr\{\mathbf{X}\}$. Druga veličina koju bi trebalo definisati je uslovna verovatnoća $PB_t(S_t) = \Pr\{X_{t+1} = 1 | S_t\}$.

$$PB_t(S_t) = \begin{cases} 0, & \nexists T_1 \\ \dfrac{P_{t+1}(T_1)}{P_{t+1}(T_1) + P_{t+1}(T_0)}, & \exists T_1 \end{cases}$$

Ovde T_1 i T_0 predstavljaju stanja trelisa sa vremenskim indeksom $t+1$ direktno povezana sa stanjem S_t. Prelaz $S_t \rightarrow T_1$ odgovara bitu $X_{t+1}=1$, dok prelazu $S_t \rightarrow T_1$ odgovara nula.

Parcijalna verovatnoća pp_t se definiše rekurzivno na osnovu PB_t.

$$pp_t(S_t) = pp_{t-1}(S_{t-1}) \cdot \begin{cases} PB_{t-1}(S_{t-1}), & X_t = 1 \\ 1 - PB_{t-1}(S_{t-1}), & X_t = 0 \end{cases}, \quad pp_0(0) = 1$$

[4] Ovo je posledica načina na koji smo formirali trelis blok koda u kom sve horizontalne linije predstavljaju nulte, a kose jedinične bitove.

[5] Da bi dobili rešenje koje minimizuje verovatnoću greške kodne reči, dovoljno je iz verovatnoća stanja naći verovatnoće kodnih reči i izabrati onu sa najvećom verovatnoćom.

Konačno, verovatnoće pojedinačnih bitova dobijamo izrazom:

$$\pi_t = \sum_{S_t} pp_t(S_t) \cdot PB_t(S_t), \qquad (2.8)$$

gde S_t prebrojava sva moguća stanja sa vremenskim indeksom t. Ovaj način dobijanja π_t zahteva manji broj izračunavanja. Parcijalne verovatnoće i verovatnoće bitova se izračunavaju po jednom za svaki čvor trelisa, što znači da vremenska kompleksnost ovog algoritma raste sa $n2^{n-k}$, što je za duže kodove značajno manja složenost nego kod prvog algoritma.

4. KOMPLEKSNOST KONSTRUKCIJE TRELISA

Problem BCJR algoritma nije samo vremenska kompleksnost izračunavanja verovatnoća sekvenci koda, već i kompleksnost konstruisanja samog trelisa. U slučaju konvolucionih kodova ovo je drastično manji problem jer su njihovi trelisi pravilni. Drugim rečima, ako zanemarimo prvih i poslednjih l veza u trelisu, ova struktura postaje periodična. Kod blok kodova toga nema. Trelisi su nepravilni i neophodno je izvršiti sva izračunavanja jer nema simetrije koja bi skratila taj posao.

Trelis se za bilo koji kod izračunava samo jednom da bi se ista struktura koristila za dekodovanje svih primljenih signala. Problem je, očigledno, praktične prirode - Da li je uopšte moguće čuvati ceo trelis u memoriji računara? Dakle, postavlja se pitanje koliko će vremena i memorije uzeti konstruisanje trelisa.

Definicioni algoritam za konstrukciju trelisa (2.6) za linearne blok kodove zahteva n sabiranja vektora za svaku kodnu reč. Ovo valja ponoviti 2^k puta jer toliko ima kodnih reči. Vremenska kompleksnost je stoga proporcionalna sa $n2^k$. Za kod (7,4) to je stotinak operacija. Za nešto složeniji kod (15,11) to je već trideset hiljada operacija.

Kompleksnost u memorijskom smislu podrazumeva količinu memorije neophodnu za skladištenje trelisa, odnosno svih njegovih čvorova. Oni se mogu jednostavno prebrojati. Postoji po jedan čvor sa vremenskim indeksima 0 i n, po dva sa indeksima 1 i n-1, po četiri sa 2 i n-2 itd. Broj čvorova raste eksponencijalno sa vremenskim indeksom sve do maksimalne vrednosti 2^{n-k}. Posmatrajući niz od n-tog člana unazad imamo isti takav rast. Odavde sledi da je ukupan broj čvorova

$$N = 2(2^{n-k} - 1) + 2^{n-k}(2k - n + 1) = (2k - n + 3)2^{n-k} - 2.$$

Ovaj broj čvorova je isti bez obzira na način konstrukcije trelisa. Konačno za $n \to \infty$ i brzinu prenosa informacija koja teži 1 imamo da su vremenska (tc) i memorijska kompleksnost (mc) proporcionalne sa

$$tc \sim n2^k \qquad mc \sim n2^{n-k}.$$

Iz ovog rezulata vidimo da vreme izvršavanja algoritma postaje kritično za iole duže kodove.

5. KOMPLEKSNOST BCJR ALGORITMA

Do sada smo razmatrali koliko "košta" konstruisanje trelisa koda i dobijanje verovatnoća pojedinačnih bitova iz verovatnoća stanja u trelisu. Sam BCJR algoritam je ono što logički dolazi između ova dva segmenta.

Za računarsku implementaciju ovog algoritma korisno je trelis predstaviti kao listu, jer na taj način izbegavamo nepotrebno računanje veličine $\gamma_t(m)$ u formuli (3). Pošto u listama znamo redosled elemenata, tj. imamo već definisane veze između čvorova trelisa, formula (4) se praktično svodi samo na nalaženje $R(Y_t, X)$.

Drugi bitan element konkretnog računarskog programa mora biti *backtracking*[6] računanje verovatnoća α_t i β_t, jer se na taj način izbegava sumiranje po svim kodnim rečima i broj izračunavanja postaje proporcionalan broju čvorova trelisa. Izračunavanje α_t i β_t se praktično svodi na rekurziju

$$\alpha_t(S_t) = R(Y_t|X)\cdot\alpha_{t-1}(S_{t-1}) + R(Y_t|\overline{X})\cdot\alpha_{t-1}(\overline{S}_{t-1}), \qquad \alpha_0(S_0) = 1$$

$$\beta_t(S_t) = R(Y_{t+1}|X)\cdot\beta_{t+1}(S_{t+1}) + (1 - R(Y_{t+1}|\overline{X}))\cdot\beta_{t+1}(\overline{S}_{t+1}), \qquad \beta_n(S_n) = 1$$

gde S_{t-1} i $\overline{S}_{t-1}$, odnosno S_{t+1} i $\overline{S}_{t+1}$ predstavljaju alternativna stanja direktno povezana sa S_t, dok X i $\overline{X}$ predstavljaju bitove određene prelazima između ovih stanja.

Kako se za svako izračunavanje vrednosti α_t i β_t svaki čvor posećuje tačno jednom, "cena" izračunavanja verovatnoće λ_t je proporcionalna broju čvorova trelisa, što znači da imamo još jedan segment algoritma čija kompleksnost raste sa $n2^{n-k}$ i da dekodovanje BCJR algoritmom za poznati trelis generalno ima pomenutu složenost.

---

[6] *Backtracking* je programerska tehnika koja omogućava prolaz kroz sve čvorove grafa tako da se između bilo koja dva direktno povezana čvora u istom smeru prolazi samo jednom.

Primena varijacione minimizacije slobodne energije na probleme dekodovanja i kriptoanalize

Za probleme dekodovanja i kriptoanalize je jako važno da postoje polinomijalni algoritmi koji vode do rešenja, jer su samo takvi algoritmi praktično upotrebljivi. Oni, nažalost, nisu optimalni[1] i predstavljaju njihove manje ili više dobre aproksimacije. Kod ekstremno dugih sekvenci više je nego poželjno da složenost algoritma bude najmanja moguća, tj. da vreme računanja bude skoro linearno zavisno od dužine signala koji obrađujemo. MekKejev algoritam (1993) varijacione minimizacije slobodne energije (FEM algoritam) ima traženu skoro linearnu složenost i daje naročito dobre rezultate sa dugim sekvencama i odgovarajućim retkim matricama.

MekKejev algoritam pretpostavlja da je problem eksponencijalne složenosti moguće rešavati tako što se sa diskretne pređe na kontinualnu reprezentaciju koja dozvoljava "fino" variranje parametara, pri tom definiše parametar koji predstavlja kvalitet rešenja i primeni neki od optimizacionih algoritma u potrazi za "najkvalitetnijim". FEM algoritam predstavlja verziju algoritma koji je po prvi put upotrebljen u statističkoj mehanici sedamdesetih godina (Feynman 1972) i čija je osnovna ideja da se komplikovan izraz za *a posteriori* raspodelu verovatnoće rešenja zameni jednostavnijim aproksimativnim separabilnim modelom koji dozvoljava kontinualno variranje parametara. Varijacione metode, generalno, spadaju u veoma bitne tehnike za aproksimiranje komplikovanih raspodela verovatnoće. Osim u teoriji informacija i statističkoj fizici, metoda se intenzivno koristi i u teoriji neuronskih mreža.

Strogo govoreći, FEM algoritam se zasniva na dobro poznatoj metodi za aproksimiranje kompleksnih raspodela, "teoriji srednjeg polja" koja je samo specijalni slučaj varijacije slobodne energije, metode koju su razvili Fejnman i Bogoljubov. Ključni matematički element za razumevanje ove metode je Gibsova nejednakost.

1. FEM ALGORITAM

U slučaju linearnih blok kodova problem dekodovanja se svodi na rešavanje matrične jednačine u polju GF(2):

$$\mathbf{s} \cdot \mathbf{A} + \mathbf{n} = \mathbf{r} \qquad (3.1)$$

po vektoru **s**, gde **s** predstavlja samu poruku, tj. izvorni kod (*source code*), **n** vektor šuma i **r** dobijeni signal. U ovako zapisanoj jednačini svi vektori su vrste. **A** je binarna matrica koja može biti ili generatorska ili transponovana matrica provere parnosti

[1] Optimalni dekoderi su oni koji na izlazu imaju sekvencu **s** koja maksimizuje *a posteriori* verovatnoću za dati kod, komunikacioni kanal i primljenu sekvencu.

(*parity check*). Ako je **A** generatorska matrica onda ima dimenzije $N \times K$ i označava se sa **G**, gde N i K predstavljaju ukupan broj bitova kodne reči i broj informacionih bitova, respektivno. Matricu provere parnosti matricu obeležavamo sa **H** i ta ona ima dimenzije $N \times N\text{-}K$. U oba slučaja, osnovni zadatak dekodovanja je naći najverovatniji vektor **s** ako znamo **r** i **A** pretpostavljajući statističke osobine vektora **s** i **n**.

1. 1. Pretpostavke za metod minimizacije slobodne energije

Pretpostavimo da su raspodele verovatnoće za **s** i **n** separabilne, tj. da važi

$$\Pr\{\mathbf{s},\mathbf{n}\} = \Pr\{\mathbf{s}\}\Pr\{\mathbf{n}\} = \prod_k \Pr\{s_k\} \prod_n \Pr\{n_n\}. \tag{3.2}$$

Ako definišemo transmisioni vektor $\mathbf{t(s)} = \mathbf{As}$, verovatnoća posmatranog vektora **r** se može zapisati kao funkcija vektora **s** (funkcija verodostojnosti):

$$\Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\} = \Pr\{\mathbf{r}|\mathbf{t}\} = \prod_n \Pr\{r_n|t_n\} = \prod_n \begin{cases} \Pr\{n_n = 1\}, & t_n \neq r_n \\ \Pr\{n_n = 0\}, & t_n = r_n \end{cases}$$

$$= \prod_n \begin{cases} \Pr\{n_n = 1\}, & r_n = 0, \quad t_n = 1 \\ \Pr\{n_n = 1\}, & r_n = 1, \quad t_n = 0 \\ \Pr\{n_n = 0\}, & r_n = 0, \quad t_n = 0 \\ \Pr\{n_n = 0\}, & r_n = 1, \quad t_n = 1 \end{cases}.$$

Ako definišemo verovatnoću

$$e_n = \begin{cases} \Pr\{n_n = 1\}, & r_n = 0 \\ \Pr\{n_n = 0\}, & r_n = 1 \end{cases},$$

uslovna verovatnoća vektora **r** se jednostavnije da zapisati kao

$$\Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\} = \prod_n e_n^{t_n}(1-e_n)^{1-t_n}.$$

Ovaj izraz koristimo u Bajesovoj (*Bayes*) formuli za *a posteriori* verovatnoću (APP) vektora **s**:

$$\Pr\{\mathbf{s}|\mathbf{r},\mathbf{A}\} = \frac{\Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\}\Pr\{\mathbf{s}\}}{\Pr\{\mathbf{r}|\mathbf{A}\}}$$

Kako najverovatnije **s** ima najveću verovatnoću $\Pr\{\mathbf{s}|\mathbf{r},\mathbf{A}\}$, naš cilj je pre svega da maksimizujemo prethodni izraz po ansamblu vektora **s**. Jedan od načina da se

napadne ovaj kombinatorni problem je da definišemo problem u kojem **s** postaje kontinualna promenljiva i gde se rešenje može tražiti nekom od kontinualnih optimizacija.

1. 2. Varijaciona minimizacija slobodne energije

Metod varijacione minimizacije slobodne energije (Feynman 1972) se primenjuje na "nezgrapne" raspodele, kao što je ova naša, i pokušava da je aproksimira jednostavnijom raspodelom $Q(\mathbf{s};\boldsymbol{\theta})$, parametrizovanom vektorom parametara $\boldsymbol{\theta}$. Mera kvaliteta aproksimacije je relativna entropija, odnosno Kulbak-Lajblerova (*Kullback-Leibler*) divergencija između dve raspodele verovatnoća $Q(x)$ i $P(x)$ definisanih na istom ansamblu,

$$D_{\mathrm{KL}}(Q\|P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)}.$$

Relativna entropija zadovoljava Gibsovu (*Gibbs*) nejednakost $D_{\mathrm{KL}}(Q\|P) \geq 0$ i jednaka je nuli samo kada je $Q=P$. Shodno tome, za meru kvaliteta aproksimirane raspodele možemo uzeti varijacionu slobodnu energiju,

$$F(\boldsymbol{\theta}) = \sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log \frac{Q(\mathbf{s};\boldsymbol{\theta})}{P(\mathbf{s})}.$$

Logično, tražimo $\boldsymbol{\theta}$ za koje je $F(\boldsymbol{\theta})$ minimalno, jer tada je aproksimativna raspodela Q najsličnija pravoj raspodeli P. U slučaju da raspodela $P(\mathbf{s})$ nije normalizovana, donja granica za P nije 0 već $-\log Z$, gde je $Z = \sum_{\mathbf{s}} P(\mathbf{s})$. Ovakav varijacioni metod se tradicionalno koristi u statističkoj fizici za procenjivanje $\log Z$, dok je u našem slučaju $\log Z$ samo aditivna konstanta koju možemo zanemariti.

Pretpostavimo da je Q separabilna raspodela, tj. da raspodelu verovatnoće Q možemo zapisati kao proizvod verovatnoća pojedinačnih bitova

$$Q(\mathbf{s};\boldsymbol{\theta}) \equiv \prod_k q_k(s_k;\theta_k), \tag{3.3}$$

gde θ_k definiše verovatnoće q_k na sledeći način:

$$q_k(s_k = 1;\theta_k) = \frac{1}{1+e^{-\theta_k}} \equiv q_k^1;\ q_k(s_k = 0;\theta_k) = \frac{1}{1+e^{\theta_k}} \equiv q_k^0. \tag{3.4}$$

Ova parametrizacija je pogodna jer je logaritam odnosa ovih verovatnoća upravo θ_k. Drugim rečima, $\theta_k = \log(q_k^1 / q_k^0)$. Za varijacionu slobodnu energiju uzimamo

$$F(\boldsymbol{\theta}) = \sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log \frac{Q(\mathbf{s};\boldsymbol{\theta})}{\Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\}\Pr\{\mathbf{s}\}}, \tag{3.5}$$

te je možemo prikazati kao zbir tri nezavisna izraza

$$F(\theta) = E_{\mathrm{L}}(\boldsymbol{\theta}) + E_{\mathrm{P}}(\boldsymbol{\theta}) - S(\boldsymbol{\theta}).$$

1) 'Entropija'[2]

$$S(\boldsymbol{\theta}) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log Q(\mathbf{s};\boldsymbol{\theta}) = -\sum_{k} [q_k^0 \log q_k^0 + q_k^1 \log q_k^1],$$

$$\frac{\partial}{\partial \theta_k} S(\boldsymbol{\theta}) = -q_k^0 q_k^1 \theta_k .$$

2) 'Energija izvornog koda'

$$E_{\mathrm{P}}(\boldsymbol{\theta}) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log \Pr\{\mathbf{s}\} = -\sum_{k} b_k q_k^1 ,$$

gde su

$$b_k = \log(\Pr\{s_k=1\}/\Pr\{s_k=0\}) \text{ i}$$

$$\frac{\partial}{\partial \theta_k} E_{\mathrm{P}}(\boldsymbol{\theta}) = -q_k^1 q_k^0 b_k .$$

---

[2] Iz MekKejevog rada nije očigledna druga jednakost u izrazu za entropiju, odnosno energiju izvornog koda. Ovde je dat dokaz jednakosti za entropiju. Dokaz za energiju izvornog koda je potpuno analogan. Po definiciji imamo:

$$S = -\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log Q(\mathbf{s};\boldsymbol{\theta}) = -\sum_{\mathbf{s}_K} Q(\mathbf{s};\boldsymbol{\theta}) \log Q(\mathbf{s};\boldsymbol{\theta}) = -\sum_{\mathbf{s}_K} \prod_{k} q_k(s_k;\theta_k) \sum_{k} \log q_k(s_k;\theta_k) \cdot$$

Ovde sumiranje po $\mathbf{s}$, odnosno $\mathbf{s}_K$ znači sumiranje po svim kodnim rečima kojih ima 2^K, pa ceo izraz $\sum_{\mathbf{s}} Q \log Q$ možemo zameniti sumom:

$$\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log Q(\mathbf{s};\boldsymbol{\theta}) = q_1^0 q_2^0 \ldots q_k^0 \log(q_1^0 q_2^0 \ldots q_k^0) + q_1^0 q_2^0 \ldots q_k^1 \log(q_1^0 q_2^0 \ldots q_k^1) + \ldots + q_1^1 q_2^1 \ldots q_k^1 \log(q_1^1 q_2^1 \ldots q_k^1) =$$

$$= q_1^0 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k (\log q_1^0 + \sum_{k=2}^{K} \log q_k) + q_1^1 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k (\log q_1^1 + \sum_{k=2}^{K} \log q_k) =$$

$$= q_1^0 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \log q_1^0 + q_1^0 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \cdot \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} \log q_k + q_1^1 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \log q_1^1 + q_1^1 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \cdot \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} \log q_k =$$

$$= q_1^0 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \log q_1^0 + q_1^1 \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \log q_1^1 + \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \sum_{\mathbf{s}_{K-1}} \sum_{k=2}^{K} \log q_k = q_1^0 \log q_1^0 + q_1^1 \log q_1^1 + \sum_{\mathbf{s}_{K-1}} \prod_{k=2}^{K} q_k \sum_{\mathbf{s}_{K-1}} \sum_{k=2}^{K} \log q_k \cdot$$

Odavde se već indukcijom može zaključiti je $S = -\sum_{k} q_k^0 \log q_k^0 + q_k^1 \log q_k^1$.

3) 'Energija verodostojnosti'

$$E_{\mathrm{L}}(\boldsymbol{\theta}) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) \log \Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\} = -\sum_{n} g_n \sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) t_n(\mathbf{s}) + const\,^{3},$$

gde je $g_n(e_n) \equiv \log(e_n/(1-e_n))$.

Da bismo izračunali vrednost E_{L}, potrebno je naći srednju vrednost $t_n(\mathbf{s})$ pod separabilnom raspodelom $Q(\mathbf{s};\boldsymbol{\theta})$, odnosno verovatnoću da je $\sum_k A_{nk} s_k = 1$. Aditivna konstanta je nezavisna od $\boldsymbol{\theta}$, pa je nadalje nećemo uzimati u obzir.

1. 3. Algoritam "unapred"

Prethodno pomenutu verovatnoću možemo izračunati rekurzivno za svako m. Definišimo vrednosti $p^1_{n,\nu}$ i $p^0_{n,\nu}$ za svako ν=1, 2, ... K kao verovatnoće da su parcijalne sume $t_n^{1\nu} = \sum_{k=1}^{\nu} A_{nk} s_k$ jednake jedinici i nuli respektivno. Ove verovatnoće zadovoljavaju sledeće jednakosti:

$$\left.\begin{aligned} p^1_{n,\nu} &= q^0_\nu p^1_{n,\nu-1} + q^1_\nu p^0_{n,\nu-1} \\ p^0_{n,\nu} &= q^0_\nu p^0_{n,\nu-1} + q^1_\nu p^1_{n,\nu-1} \end{aligned}\right\} \quad A_{n\nu} = 1$$

$$\left.\begin{aligned} p^1_{n,\nu} &= p^1_{n,\nu-1} \\ p^0_{n,\nu} &= p^0_{n,\nu-1} \end{aligned}\right\} \quad A_{n\nu} = 0.$$

Početni uslovi za rekurziju su $p^1_{n,0} = 0$ i $p^0_{n,0} = 1$. Stoga imamo da je

$$\sum_{\mathbf{s}} Q(\mathbf{s};\boldsymbol{\theta}) t_n(\mathbf{s}) = p^1_{n,K}\,.$$

Vrednost $p^1_{n,K}$ je uopštenje prozvoda t_n=$\mathbf{A}_n\mathbf{s}$ na kontinualni prostor, sa osobinom da ukoliko je i jedan sabirak jednak 0.5, onda je i rezultujuća vrednost $p^1_{n,K} = 0.5$. Tražena vrednost energije verodostojnosti je onda

$$E_{\mathrm{L}}(\theta) = -\sum_{n} g_n p^1_{n,K}\,.$$

1. 4. Algoritam "unazad"

Da bismo dobili parcijalni izvod E_{L} po θ_k neophodno je da nađemo i verovatnoće da je t_n=1 unazad, tj. počevši od $p^1_{n,K+1}$. Veličine $r^1_{n,\nu}$ i $r^0_{n,\nu}$ definisane za

---

[3] Ako uvedemo novu promenljivu $g_n(e_n) \equiv \log(e_n/(1-e_n))$, onda imamo

$$\log \Pr\{\mathbf{r}|\mathbf{s},\mathbf{A}\} = \sum_n [t_n(\mathbf{s}) \log \frac{e_n}{1-e_n} - \log(1-e_n)] = \sum_n t_n(\mathbf{s}) g_n(e_n) + const\,.$$

v=K, K-1, ..., 1 predstavljaju verovatnoće da je parcijalna suma $t_n^{\nu} = \sum_{k=\nu}^{K} A_{nk} s_k$ jednaka jedinici, odnosno nuli. Vrednost $p_{n,K}^1$ je moguće zapisati u obliku

$$p_{n,K}^1 = q_k^0 (p_{n,k-1}^1 r_{n,k+1}^0 + p_{n,k-1}^0 r_{n,k+1}^1) + q_k^1 (p_{n,k-1}^1 r_{n,k+1}^1 + p_{n,k-1}^0 r_{n,k+1}^0),$$

tako da važi za svako k. Kako samo q_k^1 i q_k^0 zavise od θ_k, izvod E_L ima oblik

$$\frac{\partial}{\partial \theta_k} E_L(\theta) = -q_k^0 q_k^1 \sum_n g_n d_{nk}, \text{ gde je}$$

$$d_{nk} = (p_{n,k-1}^1 r_{n,k+1}^1 - p_{n,k-1}^0 r_{n,k+1}^0) - (p_{n,k-1}^1 r_{n,k+1}^0 - p_{n,k-1}^0 r_{n,k+1}^1).$$

Odavde imamo da je totalni diferencijal ukupne slobodne energije:

$$\frac{\partial F}{\partial \theta_k} = q_k^0 q_k^1 \left[\theta_k - b_k - \sum_n g_n d_{nk} \right]. \qquad (3.6)$$

1. 5. Optimizacija

Iz prethodne jednakosti je očigledno da bi trebalo izjednačiti sa nulom samo izraz u uglastim zagradama i da tako dobijamo neki od minimuma funkcije $F(\theta)$. Ovo izjednačavanje možemo uraditi pomoću raznih algoritama optimizacije. MekKej je u svom radu poredio algoritme konjugovanog gradijenta i re-estimacije, odnosno sukcesivnih aproksimacija. Ovaj drugi se pokazao kao bolji, iako se algoritam konjugovanog gradijenta može poboljšati upotrebom tehnike poznate pod nazivom simulirano kaljenje (*simulated annealing*). Na ovaj način se sprečava "zarobljavanje" u lokalnim minimumima, ali se bitno produžava vreme izvršavanja algoritma.

Re-estimacija podrazumeva samo postavljanje početnih uslova i iteriranje procedure dok ne budu ispunjeni traženi uslovi. MekKej uzima na početku vrednosti θ_k=0, što odgovara verovatnoći 0.5 da je odgovarajući bit jednak jedinici. Dalje, računamo b_k i d_{nk} i to pridružujemo novim vrednostima θ_k:

$$b_k + \sum_n g_n d_{nk} \rightarrow \theta_k.$$

Algoritam re-estimacije može biti implementiran ili sa sinhronom ili sa sekvencijalnom dinamikom, odnosno možemo računati svih K vrednosti odjednom ili jednu po jednu. Sekvencijalna re-estimacija sigurno smanjuje slobodnu energiju u svakom narednom koraku, dok za sinhroni optimizator nije sigurno da je stabilan[4] iako ponekad daje bolje rezultate.

[4] Za sinhroni optimizator se može pokazati da nema pridruženu Ljapunovljevu funkciju.

Nijedan od ovih algoritama ne daje traženo rešenje sa sigurnošću jer **a)** globalni optimum i minimum slobodne energije ne moraju nužno biti povezani; **b)** može postojati više od jednog rešenja i na kraju **c)** da li je dobijeno rešenje globalni minimum ili samo neki od brojnih lokalnih minimuma u K-dimenzionalnom prostoru, to ne možemo znati bez iscrpnog pretraživanja prostora stanja, tj. svih 2^K kodnih reči.

Postoji mogućnost da se problem rešava i u nekoj drugoj reprezentaciji $\mathbf{s}'=\mathbf{s}\cdot\mathbf{U}$ sa novom matricom $\mathbf{A}'=\mathbf{U}^{-1}\cdot\mathbf{A}$, gde je $\mathbf{U}$ unitarna matrica transformacije. Ovo postaje više veoma značajno jer se pokazuje da uspeh algoritma na određenom problemu veoma zavisi od izbora reprezentacije, odnosno bazisa[5]. Očekuje se da algoritam daje najbolje rezultate ako je $\mathbf{A}$ retka matrica[6] i ako je prava *a posteriori* distribucija po $\mathbf{s}$ zaista bliska separabilnoj.

2. VARIJANTA ALGORITMA SA MATRICOM PROVERE PARNOSTI

MekKejev algoritam se može primeniti na sve probleme koji se daju predstaviti u formi $\mathbf{s}\cdot\mathbf{A}+\mathbf{n}=\mathbf{r}$, gde je $\mathbf{s}$ binarni vektor dužine K, dok su $\mathbf{n}$ i $\mathbf{r}$ binarni vektor dužine N. $\mathbf{A}$ je u tom slučaju binarna matrica dimenzija $N\times K$. Problem je naći $\mathbf{s}$ za poznat vektor $\mathbf{r}$ i matricu $\mathbf{A}$.

Kada je u pitanju dekodovanje, matrica $\mathbf{A}$ može biti ili generatorska ($\mathbf{G}$) ili transponovana matrica provere parnosti ($\mathbf{H}^T$). Obe matrice, i $\mathbf{G}$ i $\mathbf{H}$ moraju imati linearno nezavisne vrste. U prvom slučaju $\mathbf{s}$, $\mathbf{n}$ i $\mathbf{r}$ predstavljaju izvorni kod, vektor šuma i primljeni signal, respektivno i jednačina ima oblik $\mathbf{s}\cdot\mathbf{A}+\mathbf{n}=\mathbf{r}$. Sa matricom provere parnosti jednačina dobija oblik $\mathbf{s}\cdot\mathbf{A}=\mathbf{r}$, gde su $\mathbf{s}$ i $\mathbf{r}$, ovoga puta, primljeni signal i sindrom. Pošto ima 2^K vrednosti $\mathbf{s}$ kojima odgovara isti sindrom, iz praktičnih razloga se za $\mathbf{s}$ uzima samo vektor realnog šuma. Sve kodne reči, na koje delujemo istim vektorom šuma, imaju isti sindrom. On je jednak sindromu koji dobijamo za sam vektor realnog šuma, pa se naša matrična jednačina, kao i sam problem dekodovanja, svodi na nalaženje vektora šuma koji je narušio transmitovanu kodnu reč.

Razlika između ove dve varijante FEM algoritma je, očigledno, u tome što u drugom slučaju nema vektora koji karakteriše dejstvo šuma. Ukoliko je $\mathbf{s}$ kodna reč, ili nulti vektor šuma, sindrom $\mathbf{r}$ postaje jednak nuli i obrnuto, nulti sindrom znači da je $\mathbf{s}$ kodna reč, odnosno da nema šuma. Ovo, drugim rečima, znači da ne postoji "disperzija" stanja koja bi omogućila da imamo više različitih rešenja za $\mathbf{s}$ sa različitim verovatnoćama, pa samim tim nemamo ni način da definišemo vektor verovatnoća da su pojedinačni bitovi u transmitovanom signalu jedinice.

MekKej je problem varijacije rešio tako što je uveo virtuelni šum $\mathbf{v}$. Na ovaj način je definisao jednu širu klasu problema u kojoj bi rešenje koje mi tražimo trebalo da bude specijalni slučaj kada nivo virtuelnog šuma teži nuli. Uvođenjem virtuelnog

---

[5] Bazis je jednoznačno određen generatorskom, odnosno odgovarajućom matricom provere parnosti. Ove matrice nisu jedinstvene za određeni kod, pa je dozvoljeno praviti linearne kombinacije kolona, tj. vrsta u ovim matricama čime se prelazi u neki drugi bazis istog koda.

[6] Težina binarne matrice predstavlja broj jedinica koje ta matrica sadrži. Gustina je odgovarajući odnos težine i ukupnog broja bitova. Ako je gustina manja od 0.5, matrica je retka. Vrlo retka matrica je ona kod koje gustina teži nuli kada barem jedna dimenzija matrice teži beskonačnosti. Analogno se ovi atributi definišu i za vektore.

šuma dobijamo matričnu jednačinu (1) u formi $\mathbf{s}\cdot\mathbf{A}+\mathbf{v}=\mathbf{r}$. Bitno je imati u vidu da ovde **v** ne predstavlja realni šum u komunikacionom kanalu već virtuelni koji smo uveli samo zato da bi kontinualno variranje energije bilo moguće. Slično kao što smo realni šum **n** posmatrali kao BSC šum sa verovatnoćom greške na svakom pojedninačnom bitu p_c, ovde vituelni šum **v** zadajemo preko virtuelne verovatnoće greške f.

MekKej je virtuelni šum zadao preko vektora $\mathbf{g}=(g_1, g_2, \ldots g_n)$ koji upotrebljavamo za izračunavanje energije verodostojnosti. g_n uzima nenulte vrednosti, čime omogućava variranje slobodne energije (MacKay 1995):

$$g_n = \begin{cases} +1 & r_n = 1 \\ -1 & r_n = 0 \end{cases}, \qquad n = 1,2,\cdots,N,$$

što odgovara nivou virtuelnog šuma $f=1/(e+1)\approx 0.27$.

FEM algoritam sa generatorskom matricom	FEM algoritam sa matricom provere parnosti
$\mathbf{s}\cdot\mathbf{A}+\mathbf{n}=\mathbf{r}$	$\mathbf{s}\cdot\mathbf{A}+\mathbf{v}=\mathbf{r}$
A, generatorska matrica **G**	**A**, transponovana matrica provere parnosti $\mathbf{H}^T$
s, izvorni kod	**s**, vektor realnog šuma
n, šum u komunikacionom kanalu nivoa p_c	**v**, vektor virtuelnog šuma nivoa f
r, primljeni signal	**r**, sindrom

Tabela 3.1.

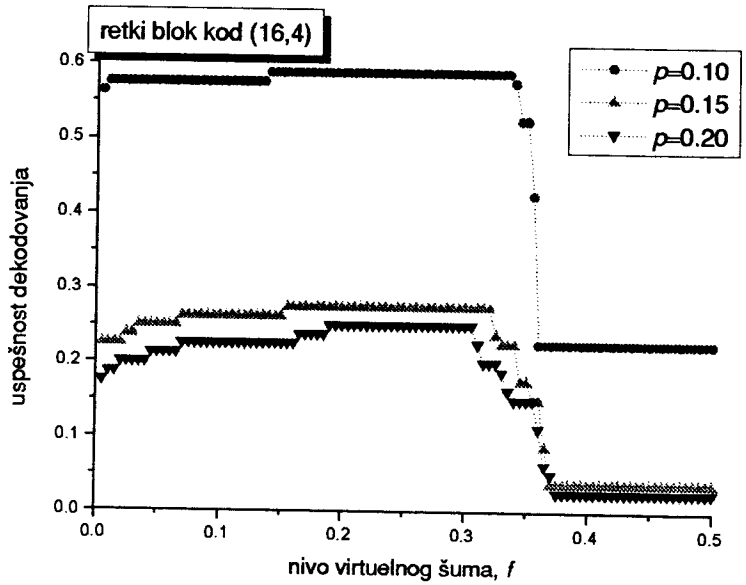
2. 1. Virtuelni šum

MekKejev algoritam pretpostavlja kontinualnu minimizaciju slobodne energije i zbog toga je u verziji algoritma sa matricom provere parnosti implicitno uveden šum koji nema poseban fizički smisao, šum koji nam samo definiše početnu "zapreminu" K-dimenzionalnog prostora u kojoj počinjemo pretragu za minimumom slobodne energije. Ovaj šum za razliku od realnog koji postoji u komunikacionom kanalu nazivamo virtuelnim[7]. Uspešnost algoritma u mnogome zavisi od nivoa virtuelnog šuma i, što je još interesantnije, optimalna vrednost tog nivoa zavisi od vrste koda i karakteristika matrice provere parnosti.

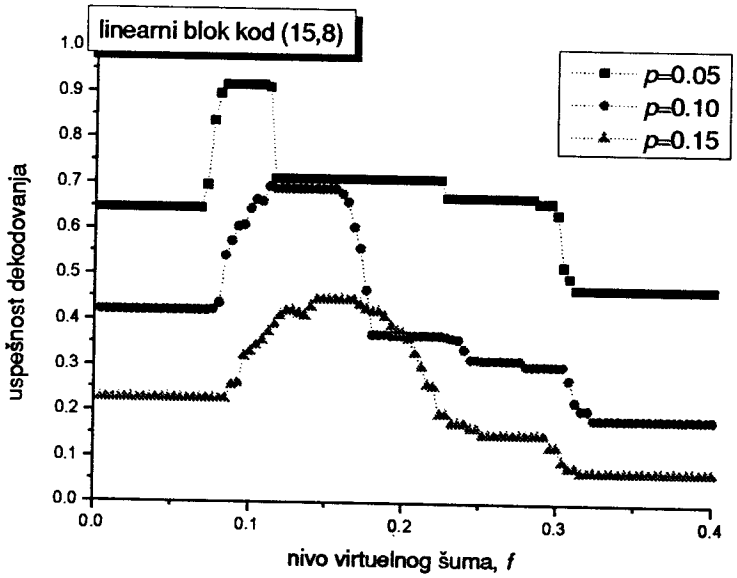
U radu u kom se porede performanse različitih korelacionih napada, odnosno algoritama za određivanje maksimalne *a posteriori* raspodele bitova u transmitovanoj sekvenci (Clark, Golić, Dawson 1996) MekKejev algoritam koristi predefinisani virtuelni šum nivoa $f=1/(e+1)$. Za neke druge primene taj nivo šuma je daleko od optimalnog. U slučaju blok kodova sa relativno malim brojem bitova u kodnoj reči moguće je i na personalnom računaru detaljno proučiti zavisnost uspešnosti algoritma od nivoa virtuelnog šuma. Ispostavlja se da su binarne sekvence sa srazmerno malim brojem

[7] Termin "virtuelni šum" se ne pominje u literaturi, ali dobro opisuje prirodu veličine koju uvodimo u FEM algoritam.

jedinica poseban slučaj gde je pomenuta zavisnost, praktično, step funkcija. Primeri uspešnosti dekodovanja u zavisnosti od nivoa virtuelnog šuma za različite nivoe šuma u komunikacionom kanalu dati su slikama 3.1. i 3.2.



Slika 3.1. - Uspešnost dekodovanja u zavisnosti od nivoa virtuelnog šuma za retki blok kod (16,4)



Slika 3.2. - Uspešnost dekodovanja u zavisnosti od nivoa virtuelnog šuma za linearni blok kod (15,8)

3. PRIMENA FEM ALGORITMA NA DEKODOVANJE

Zbog pretpostavljene separabilnosti raspodela, FEM algoritam najbolje radi sa retkim matricama i retkim vektorima. Kod kodovanja to nije čest slučaj. Većina kodova koji se danas koriste imaju za pretpostavku da originalna poruka ima maksimalnu moguću entropiju, tj. da su jedinice i nule podjednako zastupljene u sekvenci. Ovakvi vektori nisu retki jer je gustina u idelanom slučaju jednaka 0.5. Odgovarajuće matrice takođe imaju ovu gustinu jer se i od kodovanog signala očekuje da nosi maksimum informacija po simbolu.

Prava je primena FEM algoritma moguća tek kod kodova koji su konstruisani tako da budu retki. Galagerovi kodovi (Gallager 1962) i njihova modifikovana verzija, MN kodovi (MacKay, Neal 1995) imaju osobinu da su im originalne sekvence retke. Ovo znači da se redundancija ne uvodi kodovanjem već se koristi signal koji je sam po sebi redundantan.

Zašto je zgodna upotreba ovakve vrste linearnih blok kodova najbolje se vidi na primeru. Kod konvencionalnih blok kodova kodna reč je duža od originalne poruke ($N>K$) jer je neophodno uvesti kontrolne bitove. Pretpostavimo da je $N=2K$, što znači da su brzina prenosa simbola i informacija jednake 0.5. Istu poruku možemo kodovati i retkim kodovima tako da je $N=K$, tj. da brzina prenosa simbola bude jednaka 1. Neophodnu redundantnost unosimo "razređivanjem" izvornog koda tako da je njegova gustina 0.1. Odavde sledi da je brzina prenosa informacija u ovom slučaju jednaka 0.47, što je približno isto kao u slučaju konvencionalnih kodova. Drugim rečima, potpuno je svejedno da li ćemo originalnoj poruci dodati još toliko kontrolnih bitova ili ćemo poruku razrediti tako da bude samo 10% jedinica, ukupna količina informacija je približno ista.

Da bi FEM algoritam dao najbolje rezultate neophodno je ne samo da kodovi budu retki već i matrice. Algoritam konstrukcije takvih kodova su dali MekKej i Nil (1995). Za ovakve matrice je neophodno da imaju srazmerno mali broj jedinica po vrstama i kolonama. Takođe, poželjno je da broj jedinica u svakoj vrsti, tj. koloni bude jednak. MekKej i Nil navode da su ovi kodovi "dobri"[8] za tri ili više jedinica po koloni ($t\geq 3$) u generatorskoj matrici, a da mogu biti i "veoma dobri" ukoliko možemo da biramo t i odgovarajuće N.

4. PRIMENA FEM ALGORITMA U KRIPTOANALIZI

Korelacioni napad, tj. traženje stanja pomeračkog registra na osnovu posmatranja izlazne sekvence je zadatak koji se pojavljuje u nekim problemima kriptoanalize (Meier, Stafelbach 1989; Anderson 1995).

Jednostavna se kriptografska, ali teorijski sigurna metoda, dobija kombinovanjem transmitovanog signala i sekvence slučajnih brojeva poznate i pošiljaocu i primaocu. Kombinovanje se vrši korišćenjem logičke operacije ekskluzivno ILI (XOR),

---

[8] "Dobri" kodovi predstavljaju familiju kodova koji mogu postići proizvoljno malu verovatnoću greške ε za nenultu brzinu prenosa informacija (R) ispod R_{max} koji je manji ili jednak kapacitetu transmisionog kanala, C. U slučaju "veoma dobrih" kodova $R_{max}=C$.

odnosno sabiranjem po modulu 2. U praksi je problem transmitovanja sekvence random brojeva rešen korišćenjem generatora pseudoslučajnih brojeva poznatog obema stranama i inicijalizovanog na isti način. Zbog jednostavne hardverske i softverske implementacije često se kao generator pseudoslučajnih brojeva koristi linearni pomerački registar sa povratnom spregom (LFSR). LFSR dužine K bita produkuje veoma dugu sekvencu (2^K-1) pseudoslučajnih bitova.

Šifra može biti razbijena ako znamo delove otvorenog teksta i odgovarajuću kriptovanu sekvencu. Stanje pomeračkog registra se može dobiti iz K bitova otvorenog teksta, a odatle i cela kriptovana poruka. Moguće je čak i bez dela otvorenog teksta razbiti ovaj kod ako otvoreni tekst ima visoku redundantnost.

U složenijoj varijanti LFSR kriptovanja koristi se nekoliko različitih linearnih pomeračkih registara koji se kombinuju nekom nelinearnom funkcijom. Kod korelacionih napada je poželjno da ta funkcija bude što jednostavnija, pa se sama nelinearnost funkcije modeluje šumom.

Postoje dva načina na koje se izloženi problem kriptoanalize da svesti na rešavanje matrične jednačine tipa $\mathbf{s}\cdot\mathbf{A}+\mathbf{n}=\mathbf{r}$. Prvi je da definišimo **s** kao početno stanje pomeračkog registra, a **r** kao primljenu zašumljenu sekvencu, **n** kao vektor šuma i **A** kao odgovarajuću matricu transformacije. U ovoj reprezentaciji, međutim, matrica **A** postaje sve gušća, tj. ima sve veći broj jedinica, polazeći od prvog ka poslednjem redu matrice, što FEM algoritam čini neprimenljivim (MacKay 1994).

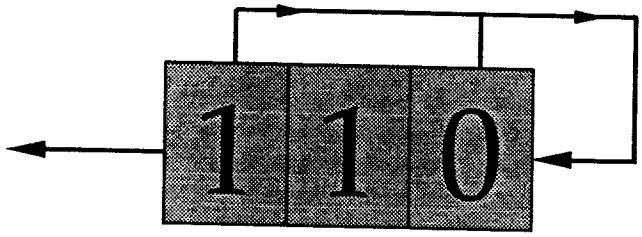
Druga reprezentacija, inspirisana metodom Majera i Štafelbaha (Meier i Staffelbach 1989), daje bolje rezultate za velike količine podataka. Ovde, **s** definišemo kao vektor šuma. Uzmimo da je originalna LFSR sekvenca jednaka nuli i neka je dobijena sekvenca $\mathbf{a1}=(\mathbf{a0}+\mathbf{s})$. Algoritmi Majera i Štafelbaha ispituju pre svega provere parnosti male težine koje zadovoljava **a0**. Ove relacije se mogu zapisati i kao $\mathbf{a0}\cdot\mathbf{A}=0$ i

$$\mathbf{s}\cdot\mathbf{A}=\mathbf{r}, \tag{3.7}$$

gde je $\mathbf{r}=\mathbf{a1}\cdot\mathbf{A}$, odnosno sindrom, vektor koji sadrži vrednosti svih provera parnosti. Relacija (3.7) definiše naš problem. Ovde bi trebalo izračunati vektor **s**, ali za razliku od jednačine (3.1) ovde nema šuma dodatog na $\mathbf{s}\cdot\mathbf{A}$. Međutim, mi možemo generalizovati problem tako da je rešenje našeg problema $\mathbf{s}\cdot\mathbf{A}+\mathbf{n}=0$ upravo granični slučaj kada nivo šuma teži nuli. Tada je moguće primeniti minimizaciju slobodne energije na ovaj problem.

Postoje dve bitne razlike u odnosu na algoritam primenjen na jednačinu (3.1) sa generatorskom matricom. Prvo, postoji neuniformna raspodela *a priori* verovatnoća po **s** koja favorizuje vektore **s** male težine. Drugo, postoji 2^K vrednosti za **s** koje zadovoljavaju jednačinu (3.7), po jedna za svako moguće inicijalno stanje pomeračkog registra plus nulto stanje. Naš je zadatak da nađemo rešenje koje ima maksimalnu verovatnoću, tj. u slučaju uniformnog i razumno malog nivoa šuma, **s** sa najmanjim brojem jedinica.

4. 1. Linearni pomerački registar sa povratnom spregom (LFSR)



Slika 3.3. - Primer LFSR-a sa tri elementa

Na slici 3.3. je dat blok dijagram linearnog pomeračkog registra sa povratnom spregom (*linear feedback shift register*) sačinjenog od tri elementa za kašnjenje. U ovom primeru imamo povratnu spregu koja je suma po modulu 2 vrednosti prvog i trećeg elementa za kašnjenje. Izlaz pomeračkog registra je vrednost elementa za kašnjenje sa leve strane, tj. jedinica.

Uopšte, pomerački registri su vremenski diskretni i sekvenca bitova se menja sa svakim novim korakom. Dužina sekvence LFSR-a i izbor elemenata za kašnjenje koji sačinjavaju povratnu spregu određuju period posle kog se sekvenca bitova ponavlja. Ukoliko je polinom koji karakteriše povratnu spregu primitivan, period će biti 2^{K-1} ciklusa, gde je K dužina registra. U polinomu povratne sprege $P(x)=1+a_1x+a_2x^2+...+a_Kx^K$ koeficijenti a_K su jednaki jedinici ako K-ti element za kašnjenje učestvuje u povratnoj sprezi, u protivnom imaju vrednost nula. Pri tome elemente za kašnjenje numerišemo sa desna na levo, što znači da je vrednost K-tog elementa za kašnjenje izlaz celog LFSR-a. Ako posmatramo izlaz LFSR-a u vremenu dobijamo pseudoslučajnu sekvencu bitova, dok niz sekvenci sadržanih u elementima za kašnjenje predstavlja pseudoslučajni niz binarnih vektora. Ovakav način dobijanja pseudoslučajnih brojeva se jako često koristi u kriptografiji zbog svoje jednostavne hardverske i softverske implementacije.

Slici 3.3. odgovara primitivni polinom $P(x)=1+x+x^3$, pa je period pomeračkog registra maksimalno moguć za dužinu $K=3$, 2^3-1. Početno stanje registra je proizvoljno izabran nenulti vektor dužine 3, u konkretnom slučaju **(110)**. U sledećem koraku sekvenca bitova će biti (101). Ceo ciklus stanja registra izgleda ovako:

1 1 0
1 0 1
0 1 0
1 0 0.
0 0 1
0 1 1
1 1 1

Prva kolona gornje matrice predstavlja vremenski niz izlaza pomeračkog registra.

Može se pokazati da za svaku konkretnu realizaciju LFSR-a postoji matrica **G** koja se može dobiti iz polinoma povratne sprege i za koju važi:

$$\mathbf{G}^T \cdot \mathbf{s} = \mathbf{u},$$

gde su **s** početna sekvenca i vremenski niz izlaza pomeračkog registra. U našem primeru, **G**, **s** i **u** imaju vrednosti:

$$\mathbf{G}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Očigledno ovde imamo linearni blok kod (7,3) sa generatorskom matricom **G**. Ovaj kod se naziva LFSR kod. Činjenica da LFSR možemo razmatrati kao kod nam omogućava da kriptoanalitički problem nalaženja početne sekvence LFSR-a (**s**) poistovetimo sa problemom dekodovanja gde tražimo izvorni kod (**s**). Naravno i u jednom i u drugom slučaju postoji šum koji nam taj posao otežava, pa matrična jednačina ima oblik:

$$\mathbf{G}^T \cdot \mathbf{s} + \mathbf{n} = \mathbf{u}$$

Kod dekodovanja **n** je šum u komunikacionom kanalu, dok kod kriptoanalitičkih problema on najčešće predstavlja samu poruku, odnosno otvoreni tekst.

Interesantno je da generatorske matrice LFSR kodova odgovaraju matricama provera parnosti odgovarajućih Hemingovih kodova i obrnuto. Drugim rečima, LFSR kod $(2^K\text{-}1, K)$ i Hemingov $(2^K\text{-}1, 2^K\text{-}1\text{-}K)$ su uzajamno dualni.

5. KOMPLEKSNOST FEM ALGORITMA

Gradijent slobodne energije se može izračunati u vremenu koje linearno zavisi od težine vrsta transformacione matrice, odnosno broja jedinica u vrstama matrica $\mathbf{G}^T$ ili **H**. Da bi složenost izračunavanja bila proporcionalna broju jedinica u matricama neophodno je u konkretnoj programerskoj realizaciji algoritma sve matrice pamtiti kao liste. Naravno, algoritam je iterativni i potrebno je proceduru ponoviti više puta da bi rezultat imao zadovoljavajuću verodostojnost. MekKej tvrdi da je N iteracija sasvim dovoljno da algoritam konvergira (MacKay 1993), iako je često potrebno mnogo manje. U svakom slučaju, kompleksnost FEM algoritma uvek raste sporije od N^2, dok je kod retkih matrica to praktično linearna zavisnost.

Poređenje BCJR i FEM algoritma

Oba algoritma, i BCJR i minimizacija slobodne energije, spadaju u klasu *sum-product* algoritama, odnosno njihov specijalni slučaj *forward/backward* (Frey *et al.* 1998; Kschischang *et al.* 1998). *Sum-product* klasa podrazumeva sve algoritme zasnovane na faktor grafovima; Perlovi "*belief propagation*" i "*belief revision*" algoritmi, FFT algoritam, Viterbi, iterativni algoritmi za dekodovanje "turbo" kodova, itd. Njihova zajednička karakteristika je postojanje procedure koja prebrojava proizvode lokalnih funkcija duž putanja faktor grafa, pri čemu se pretpostavlja da je graf stablo, odnosno da ne sadrži petlje. Opis algoritama se može uprostiti ako pretpostavimo da je svaki čvor faktor grafa sposoban da prima i šalje poruke duž veza samog grafa. Kod BCJR i FEM algoritama pomenuta procedura prenosi informacije o verovatnoćama koje karakterišu delove primljenog signala i u ovom je radu označena kao propagacija verovatnoće.

Složenost izračunavanja u ovim algoritmima, bez obzira na sličnosti, nije ista. Oba algoritma daju *a posteriori* verovatnoće bitova kodne reči, s tim što je BCJR algoritam optimalan, tj. daje "prave" *a posteriori* verovatnoće, dok je FEM algoritam baziran na aproksimacijama koje drastično pojednostavljuju izračunavanja i daju samo približne vrednosti. Optimalno dekodovanje je u slučaju linearnih blok kodova NP kompletan problem, što znači da nije verovatno da postoji polinomijalni algoritam za njegovo rešavanje[1] (Berlekamp, McEliece, van Tilborg 1978). Konkretno, BCJR algoritam ima eksponencijalnu složenost. Sa duge strane, minimizacija slobodne energije radi mnogo brže, ali ne postoje garancije da će rešenje koje da taj algoritam biti dovoljno dobro.

Prevashodni zadatak poređenja BCJR i FEM algoritma je nalaženje uslova pod kojim verovatnoće pojedinačnih bitova u transmitovanoj kodnoj reči za jedan i drugi algoritam postaju jednake, odnosno kada π_t postaje jednako q_t. Veličina π_t je definisana za BCJR algoritam u (2.6), dok je q_t verovatnoća bitova transmitovane kodne reči data formulom (3.4). Da bi nalaženje ovih uslova bilo moguće, potrebno je, pre svega, uzeti u obzir varijantu FEM algoritma sa matricom provere parnosti jer samo u tom slučaju kao izlaz oba algoritma imamo uređene n-torke koje predstavljaju verovatnoće da su pojedinačni bitovi transmitovane kodne reči jednaki jedinici. Drugi zadatak poređenja ovih algoritama je identifikovanje svih onih promenljivih koje predstavljaju iste veličine i u BCJR i u FEM algoritmu.

[1] Ne postoji dokaz da za NP kompletne probleme ne postoji polinomijalni algoritam, ali za to postoje jake indicije, te se smatra da ti problemi imaju eksponencijalnu složenost.

1. BCJR ALGORITAM NA REDUKOVANOM TRELISU

BCJR algoritam se zasniva na propagaciji verovatnoća kroz trelis dužine $n+1$ i visine 2^{n-k}. Da bi algoritam bio porediv sa minimizacijom slobodne energije, složenost, i u vremenskom, i u memorijskom smislu, mora biti polinomijalna. Ovo, bez sumnje, upućuje na nužnost uprošćavanja trelisa, a time i samog BCJR algoritma. Naravno, to uprošćenje ide na štetu tačnosti sa kojom dobijamo verovatnoće bitova kodne reči. Kod iole dužih kodnih reči, ovo je više nego opravdano.

Jedna od mogućih redukcija trelisa pretpostavlja nezavisnost između doprinosa različitih vrsta **H** matrice verovatnoćama pojedinačnih bitova kodne reči. Ovo, naravno, ne može biti u potpunosti zadovoljeno, ali daje dobre rezultate za mali broj jedinica po koloni ove matrice. Konkretno, umesto cele matrice **H** posmatramo samo pojedinačne njene vrste $\mathbf{h}^{\mu}$, gde je $\mu=1,2,...,n-k$, i za svaku od njih kreiramo odgovarajući "mini" trelis. Takvi trelisi su dužine $n+1$ i visine 2. Na ovaj način trelis visine 2^{n-k} zamenjujemo strukturom koju čini $n-k$ "mini" trelisa, redukovanim trelisom. U slučaju Hemingovog koda (7,4) umesto cele matrice **H** imamo tri vrste $\mathbf{h}^{\mu}$.

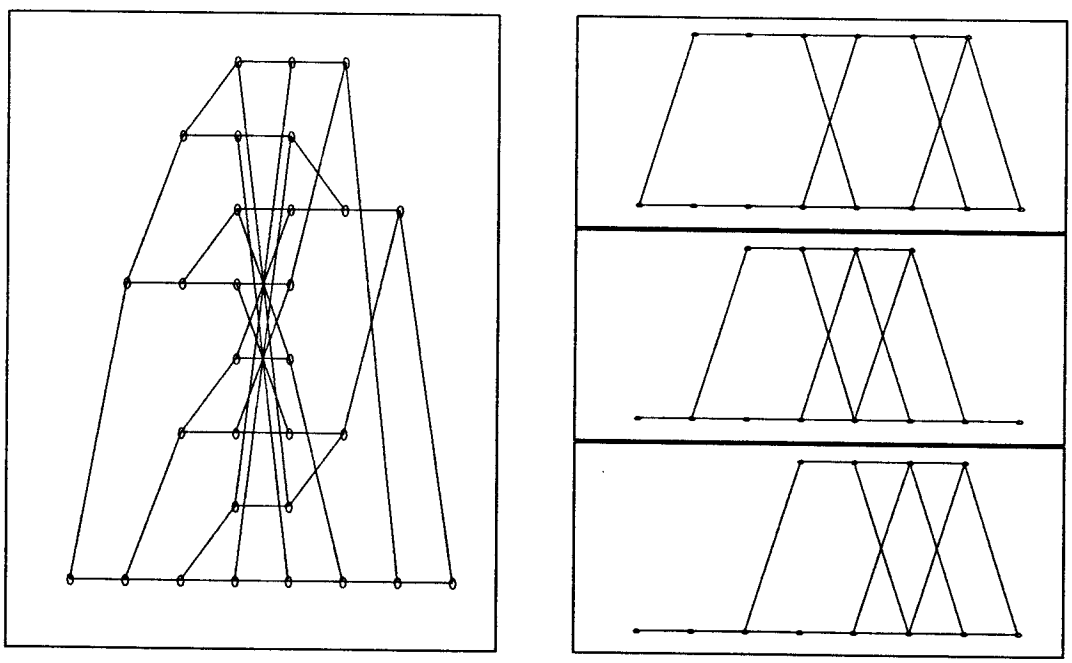
$$\mathbf{H}=\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix}$$

$$\mathbf{h}^1=[1\ 0\ 0\ 1\ 0\ 1\ 1]$$

$$\mathbf{h}^2=[0\ 1\ 0\ 1\ 1\ 1\ 0]$$

$$\mathbf{h}^3=[0\ 0\ 1\ 0\ 1\ 1\ 1]$$

Na slici 4.1. je dat primer trelisa i njegove redukovane varijante za kod zadat na ovaj način.



Slika 4.1. - Redukcija trelisa

Razlika između ova dva trelisa je, pre svega, u tome što za prvi znamo kakav je fizički smisao njegovih stanja. Za redukovani trelis imamo ***n-k***, u opštem slučaju, različitih skupova stanja $S_{t,m}^{\mu}$. Parametar μ prebrojava te skupove stanja, odnosno vrste **H** matrice, t je vremenski indeks, dok m predstavlja indeks stanja i može imati vrednosti 0 ili 1.

Naravno, čak i kada su ista stanja, tj. stanja sa istim t i m, definisana za različite vrste matrice **H**, ona mogu imati različite vrednosti. Faktički, mi ovde imamo ***n-k*** različitih kodova (n,n-1) sa samo jednim bitom provere parnosti. U tom slučaju greška na kodnoj reči može biti detektovana samo ako se nalazi na onom mestu u primljenoj sekvenci na kojoj se nalazi jedinica u vrsti provere parnosti, tj. kada je $\mathbf{h}_t^{\mu}=1$.

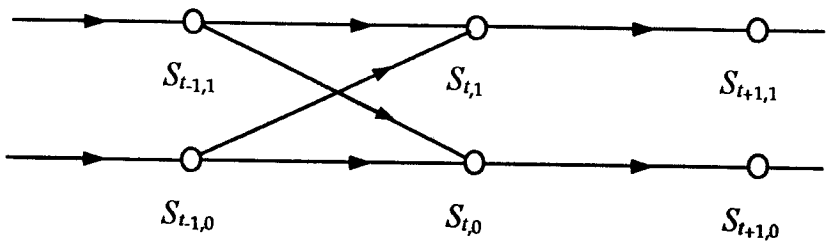
Tako umesto parametra $\alpha_{t,m}$, tj. zajedničke verovatnoće određenog stanja izvora m u trenutku t i dela izvorne sekvence $\mathbf{Y}_1^{\,t}$, definisanog za ceo trelis, formula (2.1), imamo skup od ***n-k*** različitih parametara $\alpha_{t,m}^{\mu}$ za svaku vrstu **H** matrice posebno. Isto važi i za parametar $\beta_{t,m}$.

$$\alpha_{t,m} \longrightarrow \alpha_{t,0}^{\mu} \quad \alpha_{t,1}^{\mu}$$
$$m = 0,1,\ldots 2^{n-k}-1 \qquad \mu = 0,1,\ldots n-k-1$$

$$\beta_{t,m} \longrightarrow \beta_{t,0}^{\mu} \quad \beta_{t,1}^{\mu}$$
$$m = 0,1,\ldots 2^{n-k}-1 \qquad \mu = 0,1,\ldots n-k-1$$

Rekurzivno izračunavanje parametara $\alpha_{t,m}^{\mu}$ i $\beta_{t,m}^{\mu}$ u slučaju redukovamog trelisa postaje još jednostavnije jer umesto sume po m' imamo samo dva sabirka.

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\,\gamma_t(m',m) \longrightarrow \begin{matrix} \alpha_{t,0}^{\mu} = \alpha_{t-1,0}^{\mu}\cdot\gamma_t^{\mu}(0,0) + \alpha_{t-1,1}^{\mu}\cdot\gamma_t^{\mu}(1,0) \\ \alpha_{t,1}^{\mu} = \alpha_{t-1,0}^{\mu}\cdot\gamma_t^{\mu}(0,1) + \alpha_{t-1,1}^{\mu}\cdot\gamma_t^{\mu}(1,1) \end{matrix} \qquad (4.1)$$



Slika 4.2.

Na slici 4.2. vidimo tipičnu sekvencu jednog "mini" trelisa gde t-tom bitu vrste $\mathbf{h}^{\mu}$ odgovara jedinica, a bitu t+1 nula. Ukoliko je $\mathbf{h}^{\mu}_t$=1, onda je iz stanja m' moguće preći i u stanje m=0 i u stanje m=1. Tu su mogući prelazi i horizontalno i ukoso. Horizontalni prelaz odgovara nuli kao primljenom bitu, dok kosi prelaz predstavlja jedinicu. Interesantan je slučaj kada je moguć samo horizontalan prelaz iz stanja $S^{\mu}_{t,m}$ u $S^{\mu}_{t+1,m}$. Kod celog trelisa se tako nešto ne događa jer je trelis "osetljiv" na sve bitove u kodnoj reči. Kod "mini" trelisa algoritam ne reaguje na bitove koji se nalaze na mestima gde su u μ-toj vrsti $\mathbf{H}$ matrice nule.

Da bismo upoređivali BCJR i FEM algoritam, moramo se ograničiti na BSC šum u komunikacionom kanalu kao što je to učinjeno za sam FEM. Za ovaj šum važi da su i transmitovani i primljeni signal u digitalnom obliku i da je verovatnoća greške jednaka za sve bitove. U slučaju BSC šuma izrazi za verovatnoće $\alpha^{\mu}_{t,m}$ i $\beta^{\mu}_{t,m}$ se dalje pojednostavljuju. Parametar $\gamma_t(m',m)$ uzima vrednosti 1-p_c ili p_c u zavisnosti od toga da li prelaz između stanja m' i m primljenog bita Y_t odgovara posmatranoj putanji X_t ili ne. U slučaju redukovanog trelisa imamo n-k različitih skupova parametara $\gamma^{\mu}_t(m',m)$, za svaku vrstu $\mathbf{H}$ matrice po jedan.

$$\gamma_t(m',m)=\begin{cases} p_c, & Y_t \neq X_t \\ 1-p_c, & Y_t = X_t \end{cases} \longrightarrow \gamma^{\mu}_t(m',m)=\begin{cases} p_c, & Y_t \neq X^{\mu}_t \\ 1-p_c, & Y_t = X^{\mu}_t \end{cases}$$

Ovo se drugačije da zapisati u obliku:

$$\gamma^{\mu}_t(m',m)=\begin{cases} \pi_{t,1}, & m' \neq m \\ \pi_{t,0}, & m' = m \end{cases} \text{ za } \mathbf{h}^{\mu}(t)=1 \text{ i}$$

$$\gamma^{\mu}_t(m',m)=\begin{cases} 0, & m' \neq m \\ 1, & m' = m \end{cases} \text{ za } \mathbf{h}^{\mu}(t)=0, \tag{4.2}$$

pri čemu su $\pi_{t,1}$ i $\pi_{t,0}$ definisani na sledeći način:

$$\pi_{t,1}=\begin{cases} 1-p_c, & Y_t = 1 \\ p_c, & Y_t = 0 \end{cases}, \qquad \pi_{t,0}=1-\pi_{t,1}. \tag{4.3}$$

U izrazu za $\pi_{t,1}$ prepoznajemo primljenu sekvencu gde su jedinice zamenjene sa 1-p_c, a nule sa p_c, što upravo odgovara *a priori* verovatnoći da su bitovi transmitovane kodne reči jedinice. Slično važi i za veličinu $\pi_{t,0}$. Pri tome su sve vrednosti parametara jednake bez obzira na vrednost indeksa μ. Ako je šum u komunikacionom kanalu 10%, tj. ako je p_c=0.1, i ako je primljeni bit jedinica, sa verovatnoćom od 90% možemo tvditi da je jedinica i transmitovana. Prethodno zapažanje je prvi bitan rezultat poređenja BCJR i FEM algoritma.

Sa ovako definisanim veličinama $\alpha_{t,m}^{\mu}$ i $\beta_{t,m}^{\mu}$ jednostavno se mogu izračunati verovatnoće stanja u "mini" trelisima koji odgovaraju vrstama $\mathbf{h}^{\mu}$, $P_{t,m}^{\mu}$. Koristeći analogiju sa algoritmom za ceo trelis nalazimo:

$$\lambda_{t,m}^{\mu} = \alpha_{t,m}^{\mu} \cdot \beta_{t,m}^{\mu} \quad \text{i} \quad P_{t,m}^{\mu} = \lambda_{t,m}^{\mu} / \lambda_{n,0}^{\mu}.$$

Ovde je bitno primetiti da verovatnoće stanja u "mini" trelisima, $P_{t,m}^{\mu}$, nisu jednake za različite indekse μ, odnosno različite vrste **H** matrice. Šta više, za različito μ imamo različite trelise koji se, u opštem slučaju, sastoje iz različitih skupova stanja. Jasno je da za ovako izračunate verovatnoće stanja možemo dobiti n-k međusobno različitih vektora $\pi_{t,1}^{\mu}$ za razliku od BCJR algoritma na celom trelisu koji daje samo jedan niz brojeva $\pi_{t,1}$ koji predstavlja "pravu" *a posteriori* verovatnoću da su bitovi kodne reči jedinice. Kako iz ovog "viška" podataka dobiti vrednosti verovatnoća koje što manje odstupaju od optimalnog rešenja, predstavlja problem algoritma, a ne samog redukovanog trelisa. Ovaj problem će dalje u tekstu biti detaljnije opisan.

2. POREĐENJE BCJR I FEM ALGORITMA

Da bismo našli pod kojim uslovima imamo jednoznačnu vezu između veličina $\pi_{t,1}$ i q_t^1 neophodno je uvesti sledeća ograničenja:

- FEM algoritam se primenjuje u varijanti sa matricom provere parnosti.
- BCJR algoritam se primenjuje na redukovanom trelisu.
- Razmatra se samo realan šum sa separabilnom raspodelom verovatnoće.
- BCJR se poredi samo sa prvom iteracijom FEM algoritma.

FEM algoritam se pojavljuje u dve varijante. Prva je sa generatorskom matricom, a druga sa matricom provere parnosti. U prvom slučaju, kao izlaz algoritma imamo verovatnoće pojedinačnih informacionih bitova izvornog koda kojih ima k. U drugom slučaju, izlaz je n-dimenzionalni vektor $\mathbf{q}^1 = \{q_1^1, q_2^1, \ldots q_n^1\}$ koji predstavlja ili verovatnoće bitova kodne reči ili vektora šuma. Šta će biti n-dimenzionalni vektor na izlazu FEM algoritma zavisi od početnih vrednosti za sam taj vektor. Ako za početne vrednosti uzmemo sam primljeni vektor, $\mathbf{q}^1$ će "konvergirati"[2] ka vektoru *a posteriori* verovatnoća za svaki bit transmitovane kodne reči. U tom slučaju imamo isti oblik izlaza kao i kod BCJR algoritma.

[2] Konvergenciju vektora $\mathbf{q}^1$ treba shvatiti uslovno. Prvo, nema garancija da će izlaz FEM algoritma konvergirati istoj vrednosti za različite vrednosti virtuelnog šuma. Drugo, FEM algoritam teži stanju minimuma slobodne energije, što se, u opštem slučaju, ne poklapa sa optimalnim rešenjem BCJR algoritma.

FEM algoritam se zasniva na nezavisnom računanju vektora $\mathbf{q}^1$ za svaku vrstu **H** matrice. Svako računanje verovatnoća podrazumeva horizonatalni prolaz, odnosno računanje vrednosti parametra $p^m_{\mu,t}$, i sleva udesno i zdesna ulevo. On predstavlja verovatnoću da je parcijalna suma

$$\sum_{\tau=1}^{t} \mathrm{H}_{\mu\tau} s_\tau ,$$

jednaka m, pri čemu je **s** vektor realnog šuma ili primljeni signal, μ je redni broj vrste **H** matrice, τ redni broj kolone, a m vrednost bita koja može biti ili 0 ili 1.

Od pojednostavljene verzije BCJR algoritma se očekuje da ima iste ovakve prolaze i da postoji parametar uporediv sa $p^m_{\mu,t}$. To je moguće jedino u slučaju BCJR algoritma na redukovanom trelisu.

FEM algoritam pretpostavlja separabilnost raspodele za vektor realnog šuma. To znači da verovatnoća greške na bilo kom bitu ne zavisi od toga da li se greška dogodila i na nekom drugom. U opštem slučaju, verovatnoće greške za pojedinačne bitove ne moraju biti iste, ali se najčešće u konkretnim implementacijama uzima BSC šum kada je ta verovatnoća jednaka p_c za svaki bit. U ovom radu je korišćen BSC šum, ali je moguće poređenje i za nešto opštiji slučaj.

BCJR, za razliku od FEM algoritma, nije iterativni bez obzira da li radili na celom ili redukovanom trelisu. U FEM algoritmu postoji procedura koja menja vrednosti vektora $\mathbf{q}^1$ koji onda postaje početni vektor za sledeću iteraciju. Ta procedura podrazumeva računanje slobodne energije, koja nije definisana BCJR algoritmom. Zbog ovog ograničenja, poređenje dva algoritma je moguće samo u prvoj iteraciji.

2.1. Prvi prolaz BCJR algoritma na redukovanom trelisu

Ako iskoristimo sva gore navedena ograničenja, tj. ako zamenimo vrednosti iz (4.2) u formulu (4.1) dobijamo sledeći izraz za parametar $\alpha^\mu_{t,m}$:

$$\left.\begin{aligned} \alpha^\mu_{t,1} &= \alpha^\mu_{t-1,1}\pi_{t,0} + \alpha^\mu_{t-1,0}\pi_{t,1} \\ \alpha^\mu_{t,0} &= \alpha^\mu_{t-1,0}\pi_{t,0} + \alpha^\mu_{t-1,1}\pi_{t,1} \end{aligned}\right\}, \quad \mathbf{h}^\mu(t) = 1 .$$

U slučaju nule na t-tom mestu u vrsti, tj. $\mathbf{h}^\mu_t = 0$, taj izraz postaje

$$\left.\begin{aligned} \alpha^\mu_{t,1} &= \alpha^\mu_{t-1,1} \\ \alpha^\mu_{t,0} &= \alpha^\mu_{t-1,0} \end{aligned}\right\}, \quad \mathbf{h}^\mu_t = 0 .$$

Početne vrednosti za ovaj iterativni postupak su $\alpha^\mu_{0,0} = 1$ i $\alpha^\mu_{0,1} = 0$.

Za prolaz zdesna-ulevo, odnosno za računanje parametra $\beta^{\mu}_{t,m}$, odgovarajući izrazi su potpuno analogni. Jedina razlika je što računanje počinje od $\beta^{\mu}_{n+1,m}$ i što se menja smer vremenskog indeksa.

Iz ovih izraza postaje jasno da za BCJR algoritam na redukovanom trelisu dobijamo izraze koji su potpuno analogni onima koji se koriste u FEM algoritmu.

BCJR na redukovanom trelisu	FEM algoritam
$\left.\begin{aligned}\alpha^{\mu}_{t,1} &= \alpha^{\mu}_{t-1,1}\pi_{t,0} + \alpha^{\mu}_{t-1,0}\pi_{t,1}\\ \alpha^{\mu}_{t,0} &= \alpha^{\mu}_{t-1,0}\pi_{t,0} + \alpha^{\mu}_{t-1,1}\pi_{t,1}\end{aligned}\right\}, \quad \mathbf{H}_{\mu t}=1$	$\left.\begin{aligned}p^{1}_{\mu,t} &= p^{1}_{\mu,t-1}q^{0}_{t} + p^{0}_{\mu,t-1}q^{1}_{t}\\ p^{0}_{\mu,t} &= p^{0}_{\mu,t-1}q^{0}_{t} + p^{1}_{\mu,t-1}q^{1}_{t}\end{aligned}\right\}, \quad \mathbf{H}_{\mu t}=1$
$\left.\begin{aligned}\alpha^{\mu}_{t,1} &= \alpha^{\mu}_{t-1,1}\\ \alpha^{\mu}_{t,0} &= \alpha^{\mu}_{t-1,0}\end{aligned}\right\}, \quad \mathbf{H}_{\mu t}=0$	$\left.\begin{aligned}p^{1}_{\mu,t} &= p^{1}_{\mu,t-1}\\ p^{0}_{\mu,t} &= p^{0}_{\mu,t-1}\end{aligned}\right\}, \quad \mathbf{H}_{\mu t}=0$
$\alpha^{\mu}_{0,0}=1$ i $\alpha^{\mu}_{0,1}=0$	$p^{0}_{\mu,0}=1$ i $p^{1}_{\mu,0}=0$

Tabela 4.1.

Ako za početnu vrednost vektora $\mathbf{q}^1$ uzmemo[3]

$$\mathbf{q}^{1}=\begin{cases}1-p_c, & Y_t=1\\ p_c, & Y_t=0\end{cases}, \quad \forall t,$$

onda izlaz FEM algoritma daje verovatnoće da su bitovi transmitovane kodne reči jedinice. Takva početna vrednost $\mathbf{q}^1$ u potpunosti odgovara vektoru $\boldsymbol{\pi}_1=\{\pi_{1,1},\pi_{2,1},..\pi_{n,1}\}$ koji predstavlja *a priori* verovatnoće da su bitovi kodne reči jednaki jedinici. Kako su i početne vrednosti $\alpha^{\mu}_{t,m}$ i $p^{m}_{\mu,t}$ potpuno jednake za iste indekse, imamo da su sve vrednosti ovih parametara jednake, tako da je

[3] U svim radovima posvećenim minimizaciji slobodne energije MekKej je za početni vektor uzimao q_t=0.5, $\forall t$, što znači da se pretpostavlja ista verovatnoća i za jedinice i za nule. Kako postoji k rešenja jednačine (3.1) najverovatnije je da će iterativni postupak konvergirati ka rešenju najmanje težine, što odgovara realnom vektoru šuma. Nama je, međutim, zbog analogije sa BCJR algoritmom zgodnije da za početne vrednosti uzmemo *a priori* verovatnoće da su bitovi transmitovane kodne reči jedinice.

$\alpha^{\mu}_{t,1} = p^{1}_{\mu,t}$, $\forall t,\mu$. Potpuno isto tvrđenje važi i za m=0, $\alpha^{\mu}_{t,0} = p^{0}_{\mu,t}$ $\forall t,\mu$. Jasno, $\alpha^{\mu}_{t,m}$ i $p^{m}_{\mu,t}$ imaju potpuno različite interpretacije, ali iste brojne vrednosti.

Bitno je napomenuti da indeks t kod BCJR označava vremenski indeks stanja, odnosno čvora u trelisu, dok kod FEM algoritma predstavlja redni broj bita u signalu, odnosno redni broj veze u trelisu. Logično, ako posmatramo proceduru zdesna-ulevo t-tom čvoru odgovara prelaz t+1, pa imamo $\beta^{\mu}_{t-1,1} = r^{1}_{m,t}$.

Osim gore navedene, čisto formalne, analogije između veličina $\alpha^{\mu}_{t,m}$ i $p^{m}_{\mu,t}$ moguće je naći vezu između njih i na drugi način - posmatrajući njihove reprezentacije na "mini" trelisima. Zaista,

$$p^{1}_{\mu,t} = \Pr\{\sum_{\tau=1}^{t} \mathrm{H}_{\mu\tau} s_{\tau} = 1\},$$

verovatnoća da je suma prvih t bitova proizvoda μ-te vrste **H** matrice i izvornog koda **s** jednaka 1 po modulu 2, odgovara verovatnoći stanja $\mathrm{S}^{\mu}_{t,1}$. Svaka jedinica u pomenutom proizvodu podrazumeva prelaz ukoso, što znači prelaz iz stanja 0 u 1 ili iz 1 u 0. Kako je početna vrednost 0, ukoliko je suma sekvence 0 i stanje kojim se ona završava mora imati indeks m=0. Za sumu jedan važi obrnuto.

Konačno, postoji jednostavna veza između vrednosti definisanih za BCJR algoritam na redukovanom trelisu i FEM algoritma sa matricom provere parnosti:

$\pi_{t,m} = q^{m}_{t}$	$\alpha^{\mu}_{t,m} = p^{m}_{\mu,t}$	$\beta^{\mu}_{t-1,m} = r^{m}_{\mu,t}$

Tabela 4.2.

2.2. Agregiranje, računanje slobodne energije i re-estimacija

Kao ulaz BCJR algoritma na redukovanom trelisu imamo vektor $\boldsymbol{\pi}_1$, dok na izlazu imamo n-k različitih vektora $\pi^{\mu}_{t,1}$. Kod iterativnih algoritama mora da postoji procedura koja na osnovu nekog heurističkog kriterijuma informacije iz ovih n-k vektora koristi za kreiranje novog vektora $\boldsymbol{\pi}_1$, odnosno njegove re-estimacije. Ta se procedura naziva agregiranjem. Kod FEM algoritma se agregiranje zasniva na računanju slobodne energije i traženju njenog globalnog minimuma.

Za računanje minimuma slobodne energije FEM algoritam koristi izraz dat u (3.6):

$$\frac{\partial F}{\partial \theta_t} = q^{0}_{t} q^{1}_{t} \left[\theta_t - b_t - \sum_{\mu} g_{\mu} d_{\mu t} \right],$$

pri čemu su

$$d_{\mu t} = (p^1_{\mu,t-1} r^1_{\mu,t+1} - p^0_{\mu,t-1} r^0_{\mu,t+1}) - (p^1_{\mu,t-1} r^0_{\mu,t+1} - p^0_{\mu,t-1} r^1_{\mu,t+1}),$$
$$\theta_t = \log(q^1_t / q^0_t), \qquad b_t = \log[\Pr\{s_t=1\}/\Pr\{s_t=0\}],$$

dok g_μ uzima vrednosti iz vektora virtuelnog šuma, **g**.

Sve promenljive u izrazu za totalni diferencijal slobodne energije imaju svoje analogone kod BCJR algoritma sem vektora virtuelnog šuma koji se može dodati na isti način kao u FEM algoritmu. Očigledno se slobodna energija onda može definisati i za BCJR na redukovanom trelisu i to bi otvorilo mogućnost re-stimacije i iterativnog dekodovanja, ali to više ne bi bio isti algoritam. Agregiranje u slučaju iterativnog BCJR postižemo na mnogo intuitivniji način.

3. ITERATIVNI BCJR ALGORITAM

Prvi prolaz FEM algoritma, a samim tim i potpuno analognog BCJR na redukovanom trelisu, ne daje dovoljno dobre rezultate za *a posteriori* verovatnoće pojedinačnih bitova. Da bi rezultat bio bolji neophodno je izračunati novu početnu vrednost za vektor $\mathbf{q}^1$, odnosno $\boldsymbol{\pi}_1$, i celu proceduru ponoviti onoliko puta koliko je neophodno da bi se "neodređenost" rezultata spustila ispod za nas prihvatljive granice. Ova neodređenost je kod MekKejevog algoritma definisana ukupnom slobodnom energijom sistema. Što je slobodna energija bliža nuli to su verovatnoće bitova u transmitovanoj kodnoj reči bliže jedinicama ili nulama, tj. rezultat biva određeniji. Kod BCJR algoritma na redukovanom trelisu nema iterativnog postupka, pa samim tim ni potrebe za kriterijumom koji bi odredio njegov kraj. Kada bi algoritam modifikovali tako da omogućava iteriranje, onda bi poređenje sa minimizacijom slobodne energije imalo smisla i u praktičnom pogledu.

Računanje verovatnoća svih bitova kodne reči, naravno, nije moguće na osnovu samo jedne vrste **H** matrice. Verovatnoća se da proceniti samo za ona mesta u kodnoj reči koja odgovaraju jedinicama u samoj vrsti. Vrednosti $\alpha^\mu_{t,m}$ i $\beta^\mu_{t,m}$ ne zavise od *a priori* vrednosti $\pi^\mu_{t,1}$, ako je $\mathbf{h}^\mu_t = 0$. To znači da mi u BCJR algoritmu na redukovanom trelisu verovatnoće pojedinačnih bitova procenjujemo onoliko puta koliko ima jedinica u odgovarajućoj koloni **H** matrice. Pojedinačne procene se, u opštem slučaju, razlikuju, ali je neophodno da na osnovu njih dobijemo globalnu procenu. Najjednostavniji način za ovo je da za globalnu procenu uzmemo aritmetičku srednju vrednost pojedinačnih. Na ovaj način definišemo *a posteriori* vrednost vektora $\boldsymbol{\pi}_1$, odnosno njegove komponente:

$$\pi_{t,1} = \frac{\sum_\mu \pi^\mu_{t,1} \cdot \mathbf{h}^\mu(t)}{\sum_\mu \mathbf{h}^\mu(t)}. \qquad (4.4)$$

Ukoliko ovu vrednost vektora $\boldsymbol{\pi}_1$ uzmemo za početnu i sa njim ponovimo izračunavanje, BCJR algoritam na redukovanom trelisu postaje iterativni.

Već pominjani kriterijum neodređenosti rešenja sada možemo definisati i za iterativni BCJR na redukovanom trelisu (IBCJR). Jasno, neodređenost je manja što se komponente vektora $\boldsymbol{\pi}_1$ manje razlikuju od nule ako su u intervalu [0, 0.5), odnosno jedinice ako pripadaju (0.5, 1]. Najjednostavniji način da definišemo neodređenost je da za njenu vrednost uzmemo sumu razlika od nule, odnosno jedinice:

$$\varepsilon = \sum_t \left| \pi_{t,1} - \text{round}(\pi_{t,1}) \right|,$$

pri čemu funkcija **round()** predstavlja zaokruživanje na najbližu celobrojnu vrednost

Naravno, ni u slučaju FEM ni u slučaju IBCJR algoritma nema garancija da će proces konvergirati ka vrednosti sa najmanjom neodređenošću. Uvek postoji mogućnost upadanja u lokalne minimume. Ovo se najčešće rešava upotrebom simuliranog kaljenja, mada ni ova metoda ne garantuje uspeh u potpunosti. Testiranje IBCJR algoritma na istim onim kodovima na kojima je testiran i FEM, pokazuje da se ovaj prvi mnogo češće "zaglavljuje", odnosno upada u lokalne minimume. Ovo posebno dolazi do izražaja kod gustih matrica provere parnosti. Ta razlika između algoritama, nesumnjivo, potiče od načina na koji se izračunava ulazna vrednost algoritma za narednu iteraciju. Ako za kriterijum valjanosti algoritma uzmemo njegovu uspešnost pri dekodovanju, onda je dozvoljeno modifikovati ga tako da uspešnost bude što veća. U programu za poređenje FEM i IBCJR algoritma korišćena je sledeća korekcija:

$$\pi^{\text{u}}_{t,1} = \pi^{\text{i}}_{t,1} + (\pi^{\text{i}}_{t,1} - 0.5) \cdot (0.45 - \left| \pi^{\text{i}}_{t,1} - 0.5 \right|), \qquad (4.5)$$

Ideja je da se izlazna vrednost jedne iteracije, dobijena formulom (4.4) i ovde označena kao $\pi^{\text{i}}_{t,1}$, ne koristi direktno kao ulaz sledeće, tj. $\pi^{\text{u}}_{t,1}$, već uz korekciju koja omogućava bržu konvergenciju i manji broj "zaglavljivanja". Formula (4.5) je dobijena empirijski i IBCJR algoritam sa ovom korekcijom daje bitno bolje rezultate.

3.1. Uspešnost IBCJR algoritma

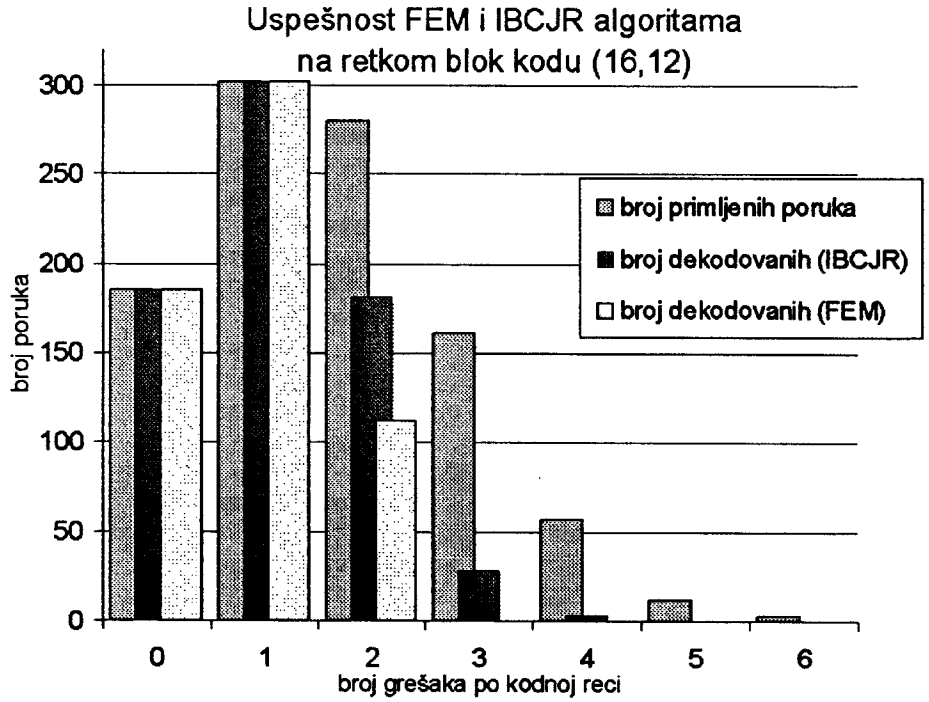
Ni FEM ni IBCJR algoritam, pre svega zbog pretpostavljene nezavisnosti doprinosa pojedinačnih vrsta **H** matrice, nisu primenljivi na blok kodove sa malom dužinom. Ukoliko matrica provere parnosti nije retka, separabilnost ima još manje opravdanja. Primera radi, algoritmi su testirani na Hemingovom kodu (7,4). Od 393 poruke sa po jednom greškom na transmitovanoj kodnoj reči FEM algoritam je uspešno dekodovao samo 343, dok je IBCJR bio još lošiji - 275. Hemingov kod, naravno, omogućava uspešno rekonstruisanje originalne poruke kad god je

u transmitovanoj poruci promenjen samo jedan bit. Sindrom dekodovanje bi, recimo, na pomenutom primeru imalo uspešnost 393 od 393.

Kvalitet algoritama koje poredimo najbolje se vidi na retkim matricama. Uostalom, MekKej je algoritam minimizacije slobodne energije prvi put upotrebio baš za MN kodove (MacKay, Neal 1995). Primer retke matrice koja zadovoljava uslove za MN, odnosno Galagerov kod, nalazimo u MekKejevoj knjizi "Information Theory, Inference and Learning Algorithms" (MacKay 1999). Matrica **H** ima težinu od 3 jedinice po koloni i dimenzija je 16×12.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Na slici 4.3. je dat grafik uspešnosti FEM i IBCJR algoritama u dekodovanju test primera retkog blok koda (16,12).



Slika 4.3. - Poređenje uspešnosti FEM i IBCJR aloritama

3.2. Mogućnosti poboljšanja IBCJR algoritma

Osnovni problem FEM i IBCJR algoritama je uslov separabilnosti koji praktično nikada nije dovoljno dobro ispunjen. Bitovi provere parnosti su uvek povezani na takav način da bez provere, koja nužno ima eksponencijalnu složenost, ne možemo znati koliko su dobra suboptimalna rešenja. Ipak, kod kodova baziranih na retkim matricama oba algoritma imaju zadovoljavajuće performanse. Bez obzira na dekoder, sam MN kod pripada klasi dobrih kodova tek za vrednosti n reda veličine 10000. Za tako velike, i pri tom retke matrice, pretpostavka o separabilnosti sigurno nije bez osnova.

Bitan rezultat ovog istraživanja je da možemo generisati niz različitih algoritama za dekodovanje koji će na potpuno isti način koristiti *forward-backward* računanje verovatnoća, a koji se međusobno razlikuju samo po načinu na koji uvode iterativni postupak, odnosno kriterijum "dovoljno dobrog" rešenja. Kod FEM algoritma imamo agregiranje bazirano na računanju slobodne energije. Kada bi slobodnu energiju definisali i za elemente IBCJR, odnosno verovatnoće dobijene na "mini" trelisima, oba algoritma bi davala iste rezultate. Očigledno, korak sa računanjem slobodne energije nije neophodan. Moguće je dobiti približno iste, pa čak i nešto bolje rezultate dekodovanja, a da pri tom ne uvodimo nove apstraktne veličine za koje ne nalazimo odgovarajući fizički smisao u sistemu koji posmatramo.

Postoji mogućnost da se pretpostavka separabilnosti "ublaži" i na taj način poveća efikasnost oba poređena algoritma na kodovima koji nisu isključivo retki. Jedna od tih mogućnosti je da verovatnoće pojedinačnih kodova ne računamo kao prostu sumu doprinosa koje daju pojedinačne vrste **H** matrice, već da uključimo i sledeći član razvoja koji bi zavisio od prozvoda doprinosa bilo koje dve vrste. Probe na nekim jednostavnijim primerima pokazuju da bi ovakav algoritam, verovatno, bio primenljiv i na kodove koji nisu retki kao MN kodovi. Drugo poboljšanje IBCJR algoritma bi trebalo tražiti u korišćenju "mini" trelisa konstruisanih za kombinacije od po dve ili više vrsta. Takav algoritam bi, nesumnjivo, bio složeniji, ali bi pomogao u situacijama gde je jednostavnija varijanta algoritma nemoćna, u traženju tačne pozicije greške u kodnoj reči koju detektuje više vrsta ali bez mogućnosti da je locira.

Zaključak

Polazeći od pretpostavke da FEM algoritam, tj.varijaciona minimizacija slobodne energije, spada u klasu algoritama za dekodovanje koji u specijalnim slučajevima predstavljaju aproksimacije optimalnog BCJR algoritma, izvedeni su uslovi pod kojima je to moguće. Oba algoritma spadaju u istu klasu *forward-backward* algoritama i pokazano je da rekurzivna izračunavanja verovatnoća pojedinačnih bitova kodne reči u oba slučaja daju identične vrednosti ukoliko posmatramo nezavisno sve vrste matrice provere parnosti. Razvijen je iterativni suboptimalni algoritam (IBCJR) koji umesto trelisa celog koda koristi redukovani trelis sačinjen od "mini" trelisa koji odgovaraju svakoj od vrsta matrice provere parnosti posebno. Kako za BCJR algoritam nije definisana slobodna energija, poređene su samo prve iteracije ova dva algoritma. Problem agregiranja podataka i postavljanja uslova za sledeću iteraciju je posebno razmatran. Zaključeno je da ova dva algoritma najbolje rezultate daju za retke blok kodove velikih dimenzija. Složenost IBCJR algoritma je linearno srazmerna dužini kodne reči i broju jedinica po koloni matrice provere parnosti.

Teorijski rezultati su provereni numeričkim eksperimentom. Značajan deo ovog rada je i paket računarskih programa za dekodovanje BCJR, FEM i IBCJR algoritmima, kao i program za upoređivanje njihovih performansi. Program za poređenje FEM i IBCJR algoritma je prikazan u dodatku ove magistarske teze.

Algoritam varijacione minimizacije slobodne energije u varijanti sa matricom provere parnosti je testiran i u slučaju linearnih blok kodova malih dimenzija. Posebno je analiziran uticaj nivoa virtuelnog šuma na uspešnost dekodovanja. Interesantan je rezultat da FEM algoritam za različite blok kodove najveću uspešnost pri dekodovanju daje za različite nivoe virtuelnog šuma. Za retke kodove se ispostavlja da ovaj parametar nije kritičan i da se najbolji rezultati dobijaju u širokom intervalu njegovih vrednosti. Vrednost za nivo virtuelnog šuma koju je MekKej implicitno predložio, praktično, predstavlja gornju granicu tog intervala.

Rezultati dobijeni za najjednostavniju redukciju trelisa upućuju na mogućnost da bi IBCJR algoritam na nešto složenijem trelisu bio još uspešniji jer bi tako bila "ublažena" pretpostavka separabilnosti realnog šuma u komunikacionom kanalu. Takvi bi algoritmi bili primenljivi i na kodove koji nisu isključivo retki. Sa druge strane, uspešnost IBCJR algoritma bi mogla biti poboljšana i ako bismo primenili drugačiji metod agregacije gde ne bismo vrste matrice provere parnosti posmatrali kao potpuno nezavisne. Složenost takvih algoritama bi bila srazmerna drugom ili trećem stepenu dužine kodne reči.

1. PROGRAM U MATLAB-U ZA POREĐENJE FEM I IBCJR ALGORITMA

```
%  Poredjenje Free Energy Minimization i
%        iterativnog BCJR algoritma
%
%        Srdjan Verbic  1998, 1999
%
%      (skript kod u MATLAB-u 5.2)

global H
global TRELLIS
global KOMS
global NOISE
global N
global K
global R

[H,G]=hammgen(3);

[K,N]=size(G);

% 2^K razlicitih poruka
s=fliplr(de2bi(0:2^K-1,K));
% slucajno izabrani kod
a=s(randint(1,1,2^K)+1,:);

% verovatnoca greske po bitu, tj. nivo suma
NOISE=0.1;

% prikaz H matrice i trelisa
colordef black;
set(figure,'Name','Dekodovanje linearnog
blok koda FEM i kombinovanim algoritmom
S. Verbic 1998','NumberTitle','off');
subplot(2,2,1);
imagesc(H);
axis equal;
axis tight;
title('matrica provere parnosti');
drawnow;

% transmitovana poruka t=a*A mod 2
t=mod(a*G,2);

% vektor suma (noise)
noise=floor(rand(1,N)/(1-NOISE));

% primljeni signal
r=mod(t+noise,2);

FEMH(noise,0.1);
flops(0);tic;
BCJR(t);
disp(['broj operacija=',num2str(flops)]);
disp(['proteklo vreme=',num2str(toc)]);
```

```
function x=FEMH(noise,vf);

%
% varijaciona minimizacija slobodne energije
%                (D. MacKay)
%
% noise – vektor suma
% vf – nivo virtuelnog suma
%

global H % matrica provere parnosti
global NOISE % nivo realnog suma

A=H'; % matrica A moze biti ili H' ili G

[N,M]=size(A);

% sindrom primljenog signala
sindrom=H*noise';

% preinicijalizacija promenljivih
y=zeros(1,N);

% bias vektor
b=log(NOISE/(1-NOISE));
bias=b*ones(1,N);
x=bias;

% F0=-w*b, konstantni clan slobodne
energije
F0=-N/2*b;

% vektor virtuelnog suma g
af=-log2(vf/(1-vf));
g=af*(sindrom*2-1);

% pocetni uslovi
```

```
q1=zeros(1,N);
q0=zeros(1,N);
p0=zeros(M,N+2);
p1=zeros(M,N+2);

for m=1:M
    p0(m,1)=1;
end
for m=1:M
    p0(m,N+2)=1;
end

d=100;
% uslov za kraj iterativnog postupka
while d>1e-8

    % algoritam unapred
    for n=1:N
        q1(n)=1/(1+exp(-x(n)));
        q0(n)=1/(1+exp( x(n)));
        for m=1:M
        if A(n,m)==0
        p0(m,n+1)=p0(m,n);
        p1(m,n+1)=p1(m,n);
        else

    p0(m,n+1)=q0(n)*p0(m,n)+q1(n)*p1(m
,n);

    p1(m,n+1)=q0(n)*p1(m,n)+q1(n)*p0(m
,n);
        end
    end
 end

 % algoritam unazad
 for n=N:-1:1
        gradient_m=0;
        for m=1:M
        if A(n,m)~=0
% akumuliranje gradijenta
        gradient_m=gradient_m-
g(m)*(p1(m,n)*p1(m,n)+p0(m,n)*p0(m,n+2
)-p0(m,n)*p1(m,n+2)-p1(m,n)*p0(m,n+2));
        end
    end
    gradient_m=gradient_m-bias(n);
    x(n)=-gradient_m;
    q1(n)=1/(1+exp(-x(n)));
    q0(n)=1/(1+exp( x(n)));
    for m=1:M
        if A(n,m)==0
            p0(m,n+1)=p0(m,n+2);
            p1(m,n+1)=p1(m,n+2);
        else

    p0(m,n+1)=q0(n)*p0(m,n+2)+q1(n)*p1
(m,n+2);

    p1(m,n+1)=q0(n)*p1(m,n+2)+q1(n)*p0
(m,n+2);
        end
        end
    end

 % slobodna energija
 % F=EL+EP+S
    EL=0;
    for m=1:M
    EL=EL+g(m)*p1(m,N+1);
    end
    EP=0;
    for n=1:N
    EP=EP+b*q1(n);
    end
    S=0;
    for n=1:N

    S=S+(q0(n)*log2(q0(n))+q1(n)*log2(q1
(n)));
    end;
    F=-(EL+EP-S);

 d=x-y;
 y=x;
 d=sqrt(sum(d.*d));

 subplot(2,2,3);
 bar(-0.1*noise,'m');
 hold on;
 bar(q1,'b');
 axis([0.5 N+0.5 -0.1 1]);
 title('vektor suma (FEM algoritam)');
 ylabel('verovatnoca greske');
 xlabel('promenjeni bitovi');
    drawnow;
    hold off;
end
x=q1; % procenjeni vektor suma
```

```
function BCJR(t);

%
% iterativna varijanta BCJR algoritma na
% redukovanom trelisu
```

```
%
% t – transmitovana poruka
%

global H
global TRELLIS
global GAMA
global ALFA
global PBITA
global PB
global KOMS
global NOISE
global N
global K
global P
global R
global LIMIT

LIMIT=99;

ppb=[];
PBITA=(R-NOISE*(r-0.5)*2)';
while sum(abs(PBITA-round(PBITA)))>0.5
spb=zeros(N+1,1);
for i=1:(N-K) % po svim vrstama H matrice
  h=H(i,:);
  brojcvorova=makeredtrellis(h);
% da ne bi racunali N*2^M puta, koristimo
% backtracking

% sad ide BCJR algoritam za blok kod
  ALFA=zeros(brojcvorova,2);
% druga kolona je, ustvari beta
  ALFA(1,1)=1; ALFA(TRELLIS(N+1,1),2)=1;
  GAMA=zeros(brojcvorova);

  indexbita=N;
% backtracking, trazimo verovatnoce gama i
% alfa
  brojcvora=TRELLIS(N+1,1);
  smer=1;
  hemrast=0; % Hemingovo rastojanje
  alfa5(brojcvora,indexbita,smer,hemrast);
  indexbita=1; % backtracking, trazimo
verovatnoce gama i beta
  brojcvora=1;
  smer=-1;
  hemrast=0;
  alfa5(brojcvora,indexbita,smer,hemrast);

  lambda=ALFA(:,1).*ALFA(:,2);
  P=lambda/lambda(1);
  PB=zeros(N+1,1);
  brojcvora=1;
  indexbita=1;
  part_prob=1;
  bit1(brojcvora,indexbita,part_prob);

  spb=spb+PB;
end

PBITA=spb./[sum(H)';1];
% empirijska formula za vektor PBITA u
sledecoj iteraciji
PBITA=PBITA+(PBITA-0.5).*(0.45-
abs(PBITA-0.5));

subplot(2,2,2);
ppb=[ppb; PBITA'];
plot(ppb);
title('konvergencija resenja');
drawnow;

subplot(2,2,4);
bar(-0.1*t,'y');
hold on;
bar(-0.02*mod(R-t,2),'m');
bar(PBITA,'g');
axis([0.5 N+0.5 -0.1 1]);
title('dekodovana kodna rec (iterativni
BCJR)');
ylabel('verovatnoca jedinicnog bita');
hold off;
drawnow;
end
```

```
function
a=alfa5(brojcvora,indexbita,smer,hemrast);

%
% izracunavanje parametra ALFA
% koriscenjem backtrackinga
%

global ALFA
global KOMS
global PBITA
global R
global LIMIT % maksimalno Hemingovo
rastojanje koje razmatramo


indexalfa=mod(smer,3); %indexalfa=1
prethodni indexalfa=2 sledeci u nizu
```

```
if hemrast<=LIMIT
  y_t=R(indexbita);
  pbc=KOMS(brojcvora,indexalfa,1);

  if ~isnan(pbc)
    if ALFA(pbc,indexalfa)==0
            ALFA(brojcvora,indexalfa)=(1-
PBITA(indexbita))*alfa5(pbc,indexbita-
smer,smer,hemrast+y_t);

    else

      ALFA(brojcvora,indexalfa)=(1-
PBITA(indexbita))*ALFA(pbc,indexalfa);
    end
  end

  pbc=KOMS(brojcvora,indexalfa,2);
  if ~isnan(pbc)
    if ALFA(pbc,indexalfa)==0

ALFA(brojcvora,indexalfa)=ALFA(brojcvora,in
dexalfa)+PBITA(indexbita)*alfa5(pbc,indexbi
ta-smer,smer,hemrast+1-y_t);
    else

ALFA(brojcvora,indexalfa)=ALFA(brojcvora,in
dexalfa)+PBITA(indexbita)*ALFA(pbc,indexal
fa);
    end
  end
end

a=ALFA(brojcvora,indexalfa);
```

```
function bit1(brojcvora,indexbita,part_prob);

%
% izracunavanje verovatnoca da su
pojedinacni bitovi jedinice
%

global KOMS
global P
global PB


sledeci=KOMS(brojcvora,2,:);

% p1 je verovatnoca da je bit(indexbita)
jednak jedinici ako kod
% prolazi kroz brojcvora
if isnan(sledeci(2)) p1=0;
else
  if ~isnan(sledeci(1))
    p12=P(sledeci(1))+P(sledeci(2));
    if p12~=0
      p1=P(sledeci(2))/p12;
    else
      p1=0;
    end
  else
    p1=1;
  end
end

PB(indexbita)=PB(indexbita)+part_prob*p1;

if ~isnan(sledeci(1))
  bit1(sledeci(1),indexbita+1,part_prob*(1-
p1));
end
if ~isnan(sledeci(2))
  bit1(sledeci(2),indexbita+1,part_prob*p1);
end
```

```
function brojcvorova=makeredtrellis(h);

%
% program za generisanje redukovanog
% trelisa
%

global TRELLIS
global KOMS
global N

% promenljiva KOMS ima 2 vrste i 2 kolone
% prvu vrstu cine indeksi prethodnih
% cvorova, a onu drugu indeksi potonjih

pola=floor((N+1)/2);
brc=2*(N+1); % brc je maksimalni broj
cvorova (malo preterana procena)
TRELLIS=NaN*ones(N+1,2); % ako cvor ne
postoji onda je element NaN
TRELLIS(1,1)=1;
KOMS=NaN*ones(brc,2,2);

% prva polovina trellisa
brojcvorova=1;
brm=1;
for j=2:pola
```

```
  if brm==1
    brojcvorova=brojcvorova+1;
    if h(j-1)==0
      TRELLIS(j,1)=brojcvorova;
      KOMS(TRELLIS(j,1),1,1)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,1)=TRELLIS(j,1);
    else
      TRELLIS(j,1)=brojcvorova;
      brojcvorova=brojcvorova+1;
      TRELLIS(j,2)=brojcvorova;
      KOMS(TRELLIS(j,1),1,1)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,1)=TRELLIS(j,1);
      KOMS(TRELLIS(j,2),1,2)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,2)=TRELLIS(j,2);
      brm=2;
    end
  else
    brojcvorova=brojcvorova+1;
    if h(j-1)==0
      TRELLIS(j,1)=brojcvorova;
      KOMS(TRELLIS(j,1),1,1)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,1)=TRELLIS(j,1);
    else
      TRELLIS(j,1)=brojcvorova;
      brojcvorova=brojcvorova+1;
      TRELLIS(j,2)=brojcvorova;
      KOMS(TRELLIS(j,1),1,1)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,1)=TRELLIS(j,1);
      KOMS(TRELLIS(j,2),1,2)=TRELLIS(j-
1,1);
      KOMS(TRELLIS(j-
1,1),2,2)=TRELLIS(j,2);
      brm=2;
    end
    if h(j-1)==0
      brojcvorova=brojcvorova+1;
      TRELLIS(j,2)=brojcvorova;
      KOMS(TRELLIS(j,2),1,1)=TRELLIS(j-
1,2);
      KOMS(TRELLIS(j-
1,2),2,1)=TRELLIS(j,2);
    else
      KOMS(TRELLIS(j,2),1,1)=TRELLIS(j-
1,2);
      KOMS(TRELLIS(j-
1,2),2,1)=TRELLIS(j,2);
      KOMS(TRELLIS(j,1),1,2)=TRELLIS(j-
1,2);
      KOMS(TRELLIS(j-
1,2),2,2)=TRELLIS(j,1);
    end
  end
end
polabrc=brojcvorova;

% druga polovina trellisa
brojcvorova=polabrc+1;
TRELLIS(N+1,1)=brojcvorova;
brm=1;
for j=N:-1:pola+1
  if brm==1
    brojcvorova=brojcvorova+1;
    if h(j)==0
      TRELLIS(j,1)=brojcvorova;

KOMS(TRELLIS(j,1),2,1)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,1)=TRELLIS(j,1);
    else
      TRELLIS(j,1)=brojcvorova;
      brojcvorova=brojcvorova+1;
      TRELLIS(j,2)=brojcvorova;

KOMS(TRELLIS(j,1),2,1)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,1)=TRELLIS(j,1);

KOMS(TRELLIS(j,2),2,2)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,2)=TRELLIS(j,2);
      brm=2;
    end
  else
    brojcvorova=brojcvorova+1;
    if h(j)==0
      TRELLIS(j,1)=brojcvorova;

KOMS(TRELLIS(j,1),2,1)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,1)=TRELLIS(j,1);
    else
      TRELLIS(j,1)=brojcvorova;
      brojcvorova=brojcvorova+1;
      TRELLIS(j,2)=brojcvorova;
```

```
KOMS(TRELLIS(j,1),2,1)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,1)=TRELLIS(j,1);

KOMS(TRELLIS(j,2),2,2)=TRELLIS(j+1,1);

KOMS(TRELLIS(j+1,1),1,2)=TRELLIS(j,2);
        brm=2;
      end
      if h(j)==0
        brojcvorova=brojcvorova+1;
        TRELLIS(j,2)=brojcvorova;

KOMS(TRELLIS(j,2),2,1)=TRELLIS(j+1,2);

KOMS(TRELLIS(j+1,2),1,1)=TRELLIS(j,2);
      else

KOMS(TRELLIS(j,2),2,1)=TRELLIS(j+1,2);

KOMS(TRELLIS(j+1,2),1,1)=TRELLIS(j,2);

KOMS(TRELLIS(j,1),2,2)=TRELLIS(j+1,2);

KOMS(TRELLIS(j+1,2),1,2)=TRELLIS(j,1);
      end
    end
end

% ove polovine valja "zasnirati"
% prvo vodoravno
if (~isnan(TRELLIS(pola+1,2)) &
~isnan(TRELLIS(pola,2)))

KOMS(TRELLIS(pola,2),2,1)=TRELLIS(pola+
1,2); end
KOMS(TRELLIS(pola,1),2,1)=TRELLIS(pola+
1,1);
if (~isnan(TRELLIS(pola+1,2)) &
~isnan(TRELLIS(pola,2)))

KOMS(TRELLIS(pola+1,2),1,1)=TRELLIS(pol
a,2); end
KOMS(TRELLIS(pola+1,1),1,1)=TRELLIS(pol
a,1);

% pa ukoso
if h(pola)==1
  if ~isnan(TRELLIS(pola+1,2))
KOMS(TRELLIS(pola,1),2,2)=TRELLIS(pola+
1,2); end
  if ~isnan(TRELLIS(pola,2))
KOMS(TRELLIS(pola,2),2,2)=TRELLIS(pola+
1,1); end
  if ~isnan(TRELLIS(pola,2))
KOMS(TRELLIS(pola+1,1),1,2)=TRELLIS(pol
a,2); end
  if ~isnan(TRELLIS(pola+1,2))
KOMS(TRELLIS(pola+1,2),1,2)=TRELLIS(pol
a,1); end
end
```

2. REČNIK ČESTO UPOTREBLJAVANIH TERMINA I SKRAĆENICA

BCJR je uobičajena oznaka za algoritam koji dekoduje zašumljene kodne reči minimizujući verovatnoću greške pojedinačnih bitova. Algoritam se zasniva na korišćenju skrivenih Markovljevih lanaca čiji su elementi definisani na trelisu samog koda. Složenost ovog algoritma je proporcionalna sa 2^{n-k}, pri čemu n i k predstavljaju ukupan broj bitova kodne reči i broj informacionih bitova, respektivno.

BSC (*binary symmetric channel*) predstavlja najjednostavniji slučaj diskretnog komunikacionog kanala bez pamćenja. Nivo šuma u BSC kanalu je jednak verovatnoći da bit transmitovane poruke bude promenjen usled dejstva šuma. Simetrija se odnosi na jednaku verovatnoću da bit nula pređe u jedinicu i obrnuto, jedinica u nulu.

DMC (*discrete memoryless channel*) je komunikacioni kanal bez pamćenja kroz koji se prenosi diskretna informaciona sekvenca. Specijalni slučaj DMC-a je binarni kanal gde diskretni skup vrednosti čine nula i jedinica. Verovatnoća da će određeni bit biti promenjen usled dejstva šuma ne zavisi od stanja ostalih bitova i verovatnoće da oni budu promenjeni.

FEM (*free energy minimization*) je skraćenica za algoritam varijacione minimizacije slobodne energije. Osnovna ideja algoritma je da se komplikovan izraz za *a posteriori* raspodelu verovatnoće rešenja zameni jednostavnijim aproksimativnim separabilnim modelom koji dozvoljava kontinualno variranje parametara.

LFSR (*linear feedback shift register*) je vremenski diskretan registar sačinjen od elemenata za kašnjenje i povratne sprege. Elementi za kašnjenje su akumulatori koji sadrže po jedan bit. Linearni pomerački registar sa povratnom spregom se koristi kao generator pseudoslučajnih brojeva.

RETKI BLOK KODOVI su oni kod kojih je broj jedinica u generatorskoj matrici manji od broja nula. Sekvence koje se koduju retkim blok kodovima su i same retke. Za razliku od uobičajenih kodova gde odnos broja jedinica i broja nula u signalu teži jedinici, ovde razlikujemo brzinu prenosa simbola i brzinu prenosa informacija. Retki blok i turbo kodovi se ubrajaju u najbolje danas poznate kodove.

SKRIVENI MARKOVLJEV LANAC je niz simbola $X_1, X_2,..., X_n$ kod kog definisana verovatnoća prelaza, odnosno uslovna verovatnoća $\Pr\{X_k|X_{k-1}\}$. Elemente Markovljevog lanca nazivamo stanjima.

TRELIS je grafička interpretacija strukture određene stanjima Markovljevog lanca i prelazima između njih. Čvorovi ovog grafa su određeni indeksiom stanja m i vremenskim indeksom t. Trelis se može definisati za svaki konvolucioni i linearni blok kod.

VARIJACIONA SLOBODNA ENERGIJA je mera sličnosti između raspodele definisane na sistemu koji nas interesuje i aproksimativne raspodele definisane na prostoru parametara koji ne moraju imati realan fizički smisao. Mera sličnosti je zadata relativnom entropijom, odnosno Kulbak-Lajblerovom divergencijom ove dve raspodele. Minimizacijom slobodne energije raspodele postaju sve sličnije.

VIRTUELNI ŠUM je veličina koju uvodimo u FEM algoritam u varijanti sa matricom provere parnosti da omogućili varijacionu minimizaciju slobodne energije. Nivo virtuelnog šuma se određuje empirijski za svaki kod i nije ni u kakvoj vezi sa nivoom realnog šuma u komunikacionom kanalu.

LITERATURA:

(Anderson 1995)
Anderson R. J., "Searching for the optimum correlation attack", Proceedings of 1994 K. U. Leuven Workshop on Cryptographic Algorithms, Lecture Notes in Computer Science, Springer-Verlag, 1995

(Ash 1965)
Ash R., *Information Theory*, John Wiley & Sons, 1965

(Bahl *et al.* 1974)
Bahl L. R., Cocke J., Jelinek F., Raviv J., "Optimal decoding of linear codes for minimizing symbol error rate", IEEE Trans. Inform. Theory, vol. IT-20, March 1974, pp. 284-287

(Beker, Piper 1982)
Beker H., Piper F., *Cypher Systems: The Protection of Communications*, Wiley, New York, 1982

(Berger, Be'ery 1993)
Berger Y., Be'ery Y., "Bounds on the trellis size of linear block codes", IEEE Trans. Inform. Theory, vol. 39, 1993, pp. 764-773

(Berlekamp 1968)
Berlekamp E. R., *Algebraic Coding Theory*, McGraw-Hill, New York, 1968

(Berlekamp, McEliece, van Tilborg 1978)
Berlekamp E., McEliece R., van Tilborg H, "On the inherent intractability of certain coding problems", IEEE Trans. Inform. Theory, vol. IT-24, May 1978, pp. 384-386

(Berrou, Glavieux, Thitimajshima 1993)
Berrou C., Glavieux A., Thitimajshima P., "Near Shannon limit error-correcting coding and decoding: Turbo codes", Proc. IEEE Int. Conf. Communications, Geneva, 1993, pp. 1064-1070

(Clark, Cain 1992)
Clark G. C., Cain J. B., *Error-correcting Coding for Digital Communications*, Plenum, New York, 1982

(Clark, Golić, Dawson 1996)
Clark A., Golić J., Dawson E., "A comparison of fast correlation attacs", Proceedings of 1996 International Workshop on Fast Software Encryption, Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 145-157

(Feldman 2000)
Feldman D., *A Brief Introduction to: Information Theory, Excess Entropy and Computational Mechanics*, http://leopard.ucdavis.edu/dave/index.html

(Feynman 1972)
Feynman R. P., *Statistical Mechanics*, W. A. Benjamin Inc., 1972

(Frey *et al.* 1998)
Frey B., Kschischang F., Loeliger H., Wiberg N., "Factor graphs and algorithms", Proceedings of 1997 Allerton Conference on Communication, Control and Computing, 1998

(Forney 1973)
Forney G. D. Jr., "The Viterbi algorithm", Proc. IEEE, vol. 61, 1973, pp. 268-276

(Gallager 1962)
Gallager R., "Low density parity check codes", IRE Trans. Inform. Theory, vol. IT-8, Jan 1962, pp. 21-28

(Garey, Johnson 1978)
Computers and Intractability: A Guide to the Theory of NP-Completness, Freeman, San Francisco, 1978

(Golić 1996)
Golić J. Dj., "Computations of low-density parity-check codes", Electronic Letters, vol. 32, 1996, pp. 1981-1982
(Golić 1999)
Golić J. Dj., "Iterative probabilistic decoding and parity checks with memory", Electronic Letters, vol. 35, 1999, pp. 1721-1723
(Golić, Salmasizadeh, Dawson 2000)
Golić J. Dj., Salmasizadeh M., Dawson E., "Fast correlation attacks on the summation generator", Journal of Cryptology, vol. 13, 2000, 245-262
(Golić *et al.* 1997)
Golić J. Dj., Salmasizadeh M., Simpson L., Dawson E., "Fast correlation attacks on nonlinear filter generators", Information Processing Letters, vol. 64, 1997, pp. 37-42
(Häggström 2000)
Häggström O., *Finite Markov Chains and Algorithmic Applications*, Göteborgs universitet, 2000
(Hamming 1986)
Hamming R. W., *Coding and Information Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1986
(Hartmann, Rudolph 1976)
Hartmann C. R. P., Rudolph L. D., "An optimal symbol-by-symbol decoding rule for linear codes", IEEE Trans. Inform. Theory, vol. IT-22, 1976, pp. 514-517
(Hopfield, Tank 1985)
Hopfield J. J., Tank D. W., "Neural computation of decisions in optimization problems", Biological Cybernetics, vol. 52, pp. 1-25
(Ivković 1980)
Ivković Z., *Matematička statistika*, Naučna knjiga, Beograd, 1980
(Kschischang, Frey, Loeliger 1998)
Kschischang F., Frey B., Loeliger H.,"Factor graphs and the sum-product algorithm", http://www.comm.utoronto.ca/frank/factor/, 1998
(Lin, Costelo 1983)
Lin S., Costello D. J., *Error Control Coding*, Prentice-Hall, N. J., 1983
(MacKay 1994)
MacKay D., "A free energy minimization algorithm for decoding and cryptanalysis", ftp://wol.ra.phy.cam.ac.uk/pub/www/mackay/fem.ps.gz, 1994
(MacKay 1995)
MacKay D., "A free energy minimization framework for inference problems in modulo 2 arithmetic", Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms, Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 179-195
(MacKay 1999)
MacKay D., *Information Theory, Inference and Learning Algorithms*, http://wol.ra.phy.cam.ac.uk/mackay/itprnn/book.ps.gz, 1999
(MacKay, Neal 1995)
MacKay D., Neal R., "Good codes based on very sparse matrices", Proceedings of 1995 IMA Conference on Cryptography and Coding, Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 100-111
(MacWilliams, Sloane 1977)
MacWilliams M. J., Sloane N. J. A., *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977
(Massey 1963)
Massey. J. L., *Threshold Decoding*, MIT Press, Cambridge, 1963
(McEliece 1977)
McEliece R. J., *The Theory of Information and Coding: A Mathematical Framework for Communication*, Addison-Wesley, Reading, Mass., 1977

(McEliece 1996)
McEliece R., "On the BCJR trellis for linear block codes", IEEE Trans. Inform. Theory, vol. IT-42, July 1996, pp. 1072-1092
(Meier, Stafelbach 1989)
Maier W., Staffelbach O., "Fast correlation attacks on certain stream ciphers", J. Cryptology, vol. 1, 1989, pp. 159-176
(Meyer, Tuchman 1972)
Meyer C. H., Tuchman W. L., "Pseudo-random codes can be cracked", Electronic Design, 1972
(Mihaljević, Golić 1991)
Mihaljević M. J., Golić J. D., "A comparison of cryptanalytic principles based on iterative error-correction", Advances in Cryptology - EUROCRYPT 91, vol. 547, 1991, pp. 527-531
(Mihaljević, Golić 1992)
Mihaljević M. J., Golić J. D., "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence", Advances in Cryptology - AUSCRYPT 90, vol. 453, 1992, pp. 165-175
(Mihaljević, Golić 1993)
Mihaljević M. J., Golić J. D., "Convergence of a Bayesian iterative error-correction procedure on a noisy shift register sequence", Advances in Cryptology - EUROCRYPT 92, vol. 658, 1993, pp. 124-137
(Pearl 1988)
Pearl J., *Probabilistic reasoning in inteligent systems: networks of plausible inference*, Morgan Kaufmann, San Mateo, 1988
(Peterson, Sodenberg 1989)
Peterson C., Sodenberg B., "A new method for mapping optimization problems onto neural networks", Int. Journal Neural Systems, 1989
(Peterson, Weldon 1972)
Peterson W. W., Weldon E. J., *Error-Correcting Codes*, MIT Press, Cambridge, 1972
(Reichl 1980)
Reichl L., *A Modern Course in Statistical Physics*, Univ. of Texas Press, 1990
(Rueppel 1986)
Rueppel R. A., *Analysis and Design of Stream Cyphers*, Springer-Verlag, Berlin, 1986
(Shannon 1948)
Shannon C., "A mathematical theory of communication", The Bell System Technical Journal, vol. 27, October 1948, pp. 623-656
(Simpson *et al.* 1999)
Simpson L., Golić J. Dj., Salmasizadeh M., Dawson E., "A fast correlation attack on multiplexer generators", Information Processing Letters, vol. 70, 1999, pp. 89-93
(Schneier 1996)
Schneier B., *Applied Cryptography*, Wiley, New York 1996
(Van den Bout, Miller 1989)
Van den Bout D. E., Miller III T. K., "Improving the performance of the Hopfield-Tank neural network through normalization and annealing", Biological Cybernetics, vol. 62, 1989, pp. 129-139
(Zurek 1990)
Zurek W. H., ed., *Complexity, Entropy and Physics of Information*, Addison-Wesley, 1990

Kratka autobiografija
(Srđan Ž. Verbić)

Srđan je rođen 1970. godine u Gornjem Milanovcu. Osnovnu školu je završio u Beogradu. 1986. godine u časopisu "Vasiona" objavio je svoj prvi rad za koji je nekoliko meseci kasnije dobio i "Oktobarsku nagradu grada Beograda". Sledeće godine je upisao studije fizike i postao stipendista Srpske akademije nauka i umetnosti. Srednju školu nije završio. Diplomirao je teorijsku fiziku 1993. godine. Od jula 1994. radi u Istraživačkoj stanici Petnica kao rukovodilac Programa fizike i specijalnog programa Exploratorium.

Od 1984. godine redovni je učesnik obrazovnih programa u Petnici. Prvo kao polaznik, a kasnije kao saradnik i rukovodilac programa. Od 1994. godine je vodio više od 50 seminara za učenike srednjih škola, studente i srednjoškolske nastavnike. Od 1997. godine je na mestu rukovodioca Odeljenja za matematičko-fizičke nauke ISP. Na temu obrazovnog modela kojim se bavi objavio je nekoliko stručnih radova, uglavnom na međunarodnim konferencijama.

O radu učitelja najbolje svedoče rezultati njegovih učenika. Primera radi, do sada je petoro Srđanovih polaznika osvajalo medalje na Međunarodnim fizičarskim Olimpijadama. Petoro njih je osvajalo prvu nagradu na prestižnom međunarodnom takmičenju "First Step to Nobel Prize". Mnogi od njegovih bivših polaznika sada studiraju na najvećim svetskim univerzitetima: MIT-u, Kaltehu, Harvardu... Mnogi od njih veruju da je Srđan pozitivno uticao na njihovo bavljenje naukom.

Srđan Verbić je post-diplomske studije je upisao na Centru za multidisciplinarne studije Univerziteta u Beogradu 1996. godine. Sve programom predviđene predmete položio je ocenom 10. Izradu magistarskog rada je započeo 1998. kod profesora Jovana Golića. Rezultati tog rada su prezentovani u decembru 1999. na seminaru za kriptologiju Matematičkog istituta SANU.