CAPITOLUL 4

COMPRESIA FĂRĂ PIERDERI UTILIZÂND CODAREA ARITMETICĂ

4.1. Introducere

Codarea aritmetică este folosită în special când alfabetul sursei este mic, iar probabilitățile mesajelor sunt dezechilibrate. Cuvintele de cod obținute în urma acestei codări au lungime variabilă.

Aşa cum s-a arătat în Capitolul 3, prin codare binară Huffman se obține un număr mediu de biți/mesaj (rata) cuprins între $H(S)$ şi $H(S)+1$. În [25] limita superioară pentru lungimea medie a cuvintelor în cazul codului Huffman a fost stabilită mai precis la valoarea $H(S)+p_{\max}+0,086$, unde $p_{\max}$ este probabilitatea cea mai mare din distribuția sursei ce urmează a fi codată.

În aplicații în care alfabetul sursei este mare şi $p_{\max}$ este relativ mic, diferența dintre limita superioară a lungimii medii a cuvintelor de cod şi entropia sursei codate este relativ mică.

Când alfabetul sursei este mic şi probabilitățile de apariție a diferitelor litere sunt foarte diferite, $p_{\max}$ poate fi mare şi codul Huffman poate deveni ineficient, privind raportul dintre entropia sursei şi lungimea medie a cuvintelor. O modalitate de a evita acest lucru este

de a efectua codarea Huffman pentru o extensie a sursei. Acest lucru, însă, nu este întotdeauna eficient.

Exemplul 4.1.

Fie o sursă care furnizează mesajele din alfabetul $S = \{s_1, s_2, s_3\}$ cu probabilitățile $p(s_1) = 0,95;\ p(s_2) = 0,02;\ p(s_3) = 0,03$. Entropia sursei este $H(S)$ =0,335 biți/mesaj.

Un cod Huffman pentru această sursă este

$s_1 \rightarrow 0$

$s_2 \rightarrow 11$

$s_3 \rightarrow 10$

pentru care lungimea medie este $\bar{l}$ =1,05 biți/mesaj. Redundanța relativă a codului este atunci $\rho = \frac{\bar{l} - H(S)}{H(S)} \approx 2,13$, ceea ce înseamnă că pentru codarea acestei surse, este nevoie de mai mult de dublul numărului de biți specificați de entropie.

Dacă se efectuează codarea extensiei de ordinul 2 a sursei, se obțin datele din Tabelul 4.1.

Tabelul 4.1

Mesaj compus	Probabilitate	Cuvânt de cod
s_1s_2	0,9025	0
s_1s_2	0,0190	111
s_1s_3	0,0285	100
s_2s_1	0,0190	1101
s_2s_2	0,0004	110011
s_2s_3	0,0006	110001

s_3s_1	0,0285	101
s_3s_2	0,0006	110010
s_3s_3	0,0009	110000

În acest caz, lungimea medie este $\overline{l}_2 = 1,222$ biți/mesaj compus, adică 0,611 bit/mesaj. În acest caz redundanța relativă devine $\rho = 0,82$. Continuând codarea pe extensii de ordin superior, redundanța relativă atinge valori acceptabile când extensia este de ordinul 8, adică un alfabet de mărime 3^8=6561, ceea ce conduce la o complexitate exagerată a codorului şi, deci, la un preț de cost inacceptabil din punct de vedere practic. Pentru eliminarea acestui inconvenient se foloseşte codarea aritmetică.

Codarea aritmetică permite atribuirea cuvintelor de cod unor secvențe particulare, fără a trebui să se genereze cuvintele de cod pentru toate secvențele de acea lungime.

Pentru înțelegerea codării aritmetice aceasta se împarte în două etape:

- în prima etapă se generează o etichetă pentru o secvență dată de mesaje;
- în a doua etapă i se atribuie etichetei un cod binar.

4.2. Codarea unei secvențe

Pentru a se putea face distincție între diferite secvențe de mesaje, fiecare este etichetată cu un identificator unic. O modalitate de reprezentare a secvenței este de a-i atribui un număr în intervalul [0, 1). O funcție care ia valori în acest interval este *funcția de repartiție* a variabilei aleatoare asociate sursei care furnizează secvența de mesaje.

Se reaminteşte că o variabilă aleatoare realizează corespondența între evenimente probabilistice şi valori pe axa reală. De exemplu, în experimentul de aruncare cu zarul, variabila aleatoare realizează corespondența dintre fețele zarului şi numerele 1,2,...,6, care sunt puncte pe axa reală. Pentru a folosi această tehnică este nevoie de a transforma mesajele sursei în numere.

Pentru uşurință, în cele ce urmează se va folosi transformarea: $x(s_i) = i; s_i \in S,$ unde $S = \{s_1, s_2, ..., s_m\}$ este alfabetul sursei. Variabila aleatoare, pe care o notăm în continuare cu *x,* poate lua valorile 1,2,...,*i*,...,*m*, cu probabilitățile cu care sunt furnizate mesajele corespunzătoare.

Aceasta înseamnă că pentru o distribuție dată a sursei, se poate asocia o funcție de repartiție pentru variabila aleatoare *x*, de forma

$$F_x(i) = \sum_{k=1}^{i} p(x = k) \qquad (4.1)$$

4.2.1. Generarea unei etichete

Procesul de generare a unei etichete se începe cu intervalul [0,1]. Funcția de repartiție a variabilei aleatoare asociate sursei este folosită la împărțirea intervalului de lungime 1 în subintervale de forma$[F_x(x_i - 1), F_x(x_i)]$, proporționale cu valorile funcției de repartiție. Cum funcția de repartiție ia valori între 0 şi 1, această abordare partiționează exact intervalul unitate.

Apariția primului mesaj din secvență restricționează intervalul care conține eticheta la unul din subintervale.

Se presupune că primul mesaj este s_k şi $x(s_k) = x_k$; atunci intervalul care conține valoarea etichetei va fi subintervalul $[F_x(x_k - 1), F_x(x_k)]$. Acesta este partiționat în subintervale proporționale

cu valorile funcției de repartiție, la fel ca intervalul original. Fiecare mesaj care urmează impune ca eticheta să fie restricționată la un subinterval care este partiționat în continuare în aceleaşi proporții.

Exemplul 4.2.

Fie sursa cu alfabetul $S = \{s_1, s_2, s_3\}$ cu $p(s_1) = 0,7$; $p(s_2) = 0,1$; $p(s_3) = 0,2$. Folosind corespondența dată de relația (4.1) rezultă că $F_x(1) = 0,7$, $F_x(2) = 0,8$, $F_x(3) = 1$. Împărțirea intervalului [0,1] este dată în Fig. 4.1. Pe prima verticală din figură s-au marcat punctele 0,7 şi 0,8. adică intervalul [0,1] a fost împărțit proporțional cu numerele 0,7 şi 0,8.

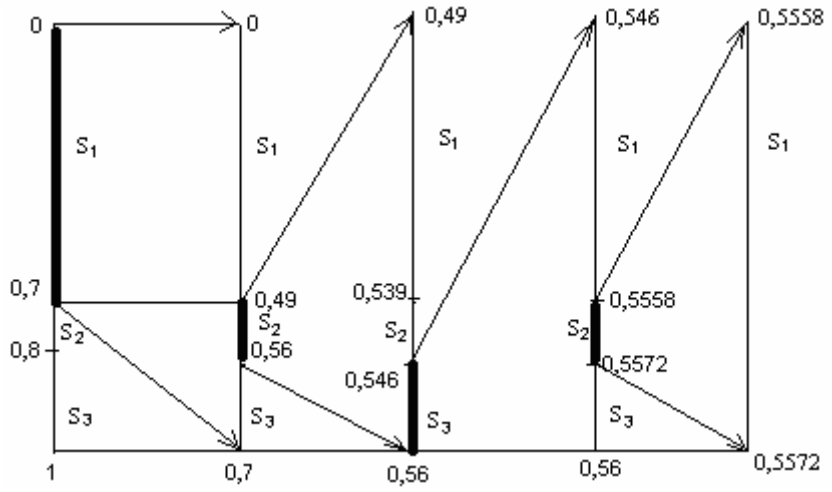

Fig. 4.1. Restricționarea intervalului care conține eticheta pentru secvența $s_1s_2s_3s_2$

Subintervalul în care se află eticheta depinde de primul mesaj din secvența ce urmează a fi codată. Dacă acesta este s_1, eticheta este în intervalul [0;0,7), dacă este s_2, eticheta se află în intervalul [0,7;0,8),

dacă este s_3, aceasta se află în intervalul [0,8;1). Odată ce a fost determinat intervalul în care se află eticheta, restul intervalului unitate este îndepărtat şi intervalul rămas se împarte în aceleaşi proporţii ca intervalul original. În cazul exemplului de față, intervalul [0; 0,7], corespunzător primului mesaj se împarte în aceleaşi proporţii ca intervalul [0,1]. Pe cea de-a doua verticală din Fig. 4.1, intervalul [0;0,7] se împarte proporțional cu numerele 0,7, respectiv 0,8, marcându-se astfel punctele 0,7 x 0,8=0,56 şi 0,7 x 0,7 =0,49. Lungimile celor trei intervale sunt 0,49 – 0=0,49; 0,56-0,49=0,07, şi, respectiv, 0,7-0,56=0,14. Deoarece cel de-al doilea mesaj este s_2, se reține intervalul de lungime 0,07, cu limitele 0,49 şi 0,56. Pentru a treia verticală se reține numai intervalul [0,49; 0,56] a cărui lungime de 0,07 se împarte proporțional cu numerele 0,7 şi 0,8, obținându-se astfel punctele: 0,07 x 0,7 + 0,49 = 0,539 şi 0,07 x 0,8 + 0,49 = 0,546. Deoarece cel de-al treilea mesaj este s_3, se reține intervalul cu limitele 0,546 şi 0,56, cu care se construieşte cea de a patra verticală, pe care se marchează punctele: (0,56 – 0,546) x 0,7 + 0,546 = 0,5558, respectiv, (0,56 – 0,546) x 0,8 + 0,546 = 0,5572. Deoarece cel de al patrulea mesaj este s_2, se alege intervalul cu limitele 0,5558 şi 0,5572. Eticheta pentru această succesiune de mesaje se alege ca un punct din interiorul acestui interval.

Se observă că apariția oricărui nou mesaj restricționează eticheta la un interval separat de orice alt interval care poate fi generat folosind acest procedeu. Chiar dacă două secvențe sunt identice până la un punct, intervalele etichetelor sunt întotdeauna disjuncte. Aceasta înseamnă că orice punct (valoare) din acel interval poate fi folosit ca etichetă. De obicei, eticheta se alege limita inferioară sau mijlocul intervalului.

Fie cazul în care eticheta are valoarea punctului din mijlocul intervalului. Se presupune că sursa generează mesaje din alfabetul $S = \{s_1, s_2, \ldots, s_m\}$. Mesajele $\{s_i\}$ sunt transformate în numerele reale $\{i\}$, care definesc o variabilă aleatoare discretă. Eticheta mesajului s_i se defineşte cu relația

$$\overline{T}_x(s_i) = \sum_{k=1}^{i-1} p(x=k) + \frac{1}{2} p(x=i) = F_x(i-1) + \frac{1}{2} p(x=i) \qquad (4.2)$$

Pentru fiecare mesaj s_i, eticheta $\overline{T}_x(s_i)$ va avea o valoare unică.

Exemplul 4.3.

Fie experimentul de aruncare cu un zar corect. Rezultatele unei aruncări cu zarul pot fi transformate în numerele {1,2,…,6}. Pentru un zar corect $p(x=k) = \frac{1}{6}$ pentru k=$\overline{1,6}$. Cu relația (4.2), rezultă etichetele pentru realizările experimentului de aruncare cu zarul date în Tabelul 4.2.

Tabelul 4.2

Realizare	$\overline{T}_x(s_i)$
1	0,08(3)
2	0,25
3	0,41(6)
4	0,58(3)
5	0,75
6	0,9166

Acest mod de atribuire a unei etichete unei secvențe de lungime unitară se poate extinde la secvențe de lungime *n*, impunând o ordine în

secvență. În acest caz eticheta corespunzătoare secvenței de lungime *n* se calculează cu relația

$$\overline{T}_x^{(n)}(x_i) = \sum_{y<x_i} p(y) + \frac{1}{2} p(x_i) \tag{4.3}$$

unde $y<x_i$ înseamnă că y precede pe x_i în ordonare şi (n) semnifică lungimea secvenței.

O ordonare simplă este cea lexicografică, care este echivalentă cu ordinea dicționarului.

Exemplul 4.4.

Exemplul 4.3 se poate extinde, astfel încât secvența constă în 2 aruncări cu zarul. Realizările ordonate sunt 11, 12, 13,..., 66. Etichetele pot fi generate cu relația (4.3). De exemplu, pentru secvența 13, eticheta este

$$\begin{aligned} \overline{T}_x(13) &= p(x=11) + p(x=12) + \frac{1}{2} p(x=13) = \\ &= \frac{1}{36} + \frac{1}{36} + \frac{1}{2} \cdot \frac{1}{36} = \frac{5}{72} \end{aligned} \tag{4.4}$$

Se observă că pentru a genera o etichetă pentru o succesiune dată de mesaje trebuie cunoscute numai probabilitățile mesajelor individuale.

Prin modul de generare a etichetei şi a unicității intervalului căruia îi corespunde, orice valoare din interval poate fi un identificator unic pentru x_i. Prin urmare, pentru a îndeplini obiectivul inițial de a identifica în mod unic o secvență, este suficient a calcula limita superioară (u) şi inferioară (l) a intervalului ce conține eticheta şi a selecta orice valoare din interval.

Exemplul 4.5.

Se consideră sursa din Exemplul 4.3 şi se doreşte a găsi limitele intervalului care conține eticheta pentru secvența 322. Pentru această secvență se calculează limitele superioară şi inferioară ale intervalelor corespunzătoare lui 3, apoi 2 şi iar 2.

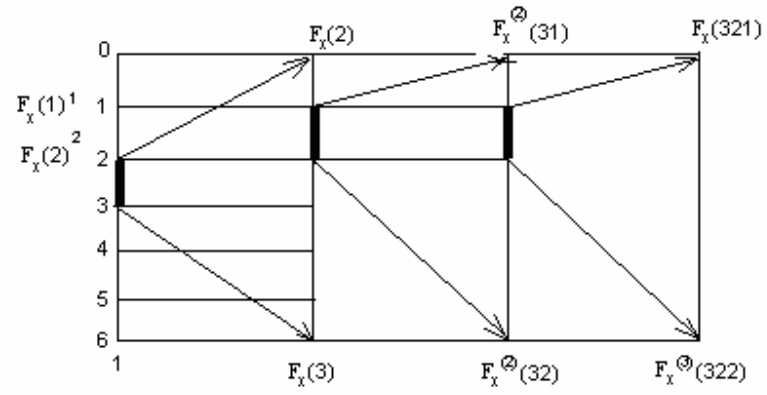

Fig. 4.2.

Pentru primul mesaj $u^{(1)} = F_x(3) \quad l^{(1)} = F_x(2)$

Al doilea mesaj este 2 şi secvența devine x=32, pentru care $u^{(2)} = F_x^{(2)}(32)\,; \;\; l^{(2)} = F_x^{(2)}(31)$

$$F_x^{(2)}(32) = p(x=11) + p(x=12) + \ldots + p(x=16) + p(x=21) + \\ + p(x=22) + \ldots + p(x=26) + p(x=31) + p(x=32) \tag{4.5}$$

Dar

$$\sum_{i=1}^{6} p(x = x_1 x_2 = ki) = \sum_{i=1}^{6} p(x_1 = k)\, p(x_2 = i) = \\ = p(x_1 = k)[p(x_2 = 1) + \ldots p(x_2 = 6)] = p(x_1 = k) \tag{4.6}$$

şi relația (4.5) devine

$$F_x^{(2)}(32) = p(x_1 = 1) + p(x_2 = 2) + p(x = 31) + p(x = 32) = \\ F_x(2) + p(x = 31) + p(x = 32) \tag{4.7}$$

$$p(x=31)=p(x_1=3)p(x_2=1) \tag{4.8}$$

$$p(x=32)=p(x_1=3)p(x_2=2) \tag{4.9}$$

$$p(x=31)+p(x=32)=p(x_1=3)[p(x_2=1)+p(x_2=2)]= \\ =p(x_1=3)F_x(2) \tag{4.10}$$

$$p(x_1=3)=F_x(3)-F_x(2) \tag{4.11}$$

Cu (4.11), relația (4.10) devine

$$p(x=31)+p(x=32)=(F_x(3)-F_x(2))F_x(2) \tag{4.12}$$

şi

$$F_x^{(2)}(32)=F_x(2)+(F_x(3)-F_x(2))F_x(2) \tag{4.13}$$

adică

$$u^{(2)}=l^{(1)}+(u^{(1)}-l^{(1)})F_x(2) \tag{4.14}$$

Similar

$$F_x^{(2)}(31)=F_x(2)+p(x=31)=F_x(2)+p(x_1=3)p(x_2=1)= \\ F_x(2)+(F_x(3)-F_x(2))F_x(1) \tag{4.15}$$

sau

$$l^{(2)}=l^{(1)}+(u^{(1)}-l^{(1)})F_x(1) \tag{4.16}$$

Al treilea mesaj este 2 şi secvența este 322. Limitele superioară şi inferioară ale intervalului care conține eticheta sunt:

$$u^{(3)}=F_x^{(3)}(322) \quad l^{(3)}=F_x^{(3)}(321) \tag{4.17}$$

Urmând aceeaşi procedură în determinarea limitelor, se obține:

$$F_x^{(3)}(322)=$$

$$\underbrace{\underbrace{P(x=111)+P(x=112)+..P(x=116)}_{P(11)}+\underbrace{P(x=121)+...P(x=126)}_{P(12)}+...\underbrace{P(x=161)..+P(x=166)}_{P(16)}+}_{P(1)}$$

$$+\underbrace{\underbrace{P(x=211)+..P(x=216)}_{P(21)}+\underbrace{P(x=221)+..+P(x=226)}_{P(22)}+...\underbrace{P(x=261)+..+P(x=266)}_{P(26)}+}_{P(2)}$$

$$+\underbrace{P(x=311)+P(x=312)+\ldots+P(x=316)}_{P(31)}+P(x=321)+P(x=322)=$$

$$=\underbrace{(F_x(2)+p(31))}_{F_x^{(2)}(31)}+p(32)\underbrace{(p(1)+p(2))}_{F_x(2)}=F_x^{(2)}(31)+(F_x^{(2)}(32)-F_x^{(2)}(31))F_x(2) \quad (4.18)$$

unde

$$p(32)=F_x^{(2)}(32)-F_x^{(2)}(31),$$

$$F_x^{(2)}(31)=p(11)+p(12)+\ldots+p(16)+p(21)+\ldots p(26)+p(31)$$

şi

$$F_x^{(3)}(321)=F_x^{(2)}(31)+p(x=321)=F_x^{(2)}(31)+p(32)\cdot p(1)= \quad (4.19)$$
$$F_x^{(2)}(31)+(F_x^{(2)}(32)-F_x^{(2)}(31))F_x(1)$$

sau

$$\left.\begin{aligned} u^{(3)}&=l^{(2)}+(u^{(2)}-l^{(2)})F_x(2)\\ l^{(3)}&=l^{(2)}+(u^{(2)}-l^{(2)})F_x(1)\end{aligned}\right\} \quad (4.20)$$

În general, pentru o secvență $x=x_1x_2\ldots x_n$, se poate arăta că limita superioară şi cea inferioară a intervalului etichetei pot fi calculate recursiv, cu relațiile [86]

$$l^{(n)}=l^{(n-1)}+(u^{(n-1)}-l^{(n-1)})F_x(x_n-1) \quad (4.21)$$

$$u^{(n)}=l^{(n-1)}+(u^{(n-1)}-l^{(n-1)})F_x(x_n) \quad (4.22)$$

unde prin $l^{(n)}$ şi $u^{(n)}$ s-au notat limitele inferioară, respectiv, superioară a intervalului ce conține eticheta corespunzătoare succesiunii de *n* mesaje. În relațiile (4.21) şi (4.22) se va considera $l^{(0)}=0$ şi $u^{(0)}=1$ deoarece înainte de a furniza mesajele intervalul inițial se consideră [0,1].

Dacă se foloseşte mijlocul intervalului pentru etichetă, rezultă:

$$\overline{T}_x(x)=\frac{u^{(n)}+l^{(n)}}{2} \quad (4.23)$$

În concluzie, eticheta oricărei secvențe se poate calcula secvențial, singura informație necesară fiind funcția de repartiție a sursei, care poate fi obținută din modelul de probabilitate.

Exemplul 4.6.

Fie sursa cu alfabetul $S = \{s_1, s_2, s_3\}$, cu $p(s_1)=0,8$, $p(s_2)=0,02$, $p(s_3)=0,18$. Se defineşte variabila aleatoare $x(s_i)=i$ şi se doreşte a coda secvența 1321.

Din modelul de probabilitate se obține

$F_x(k) = 0, k \le 0, F_x(1) = 0,8; F_x(2) = 0,82; F_x(3) = 1; F_x(k) = 1,$ pentru k>3.

Se folosesc relațiile (4.21) şi (4.22). Se inițializează cu $u^{(0)} = 1, l^{(0)} = 0$.

Primul element al secvenței este 1, aşa că rezultă

$$l^{(1)} = 0 + (1-0)\cdot 0 = 0$$

$$u^{(1)} = 0 + (1-0)\cdot 0,8 = 0,8$$

adică eticheta este conținută în intervalul [0, 0,8). Următorul element este 3 şi rezultă:

$$l^{(2)} = 0 + (0,8-0)F_x(2) = 0,656$$

$$u^{(2)} = 0 + (0,8-0)F_x(3) = 0,8$$

aşa că intervalul care conține eticheta pentru secvența 13 este [0,656; 0,8). În continuare urmează 2.

$$l^{(3)} = 0,656 + (0,8-0,656)F_x(1) = 0,7712$$

$$u^{(3)} = 0,656 + (0,8-0,656)F_x(2) = 0,77408$$

Intervalul pentru etichetă este [0,7712; 0,77408).

Urmează 1 şi limitele intervalului devin:

$$l^{(4)} = 0,7712 + (0,77408 - 0,7712)F_x(0) = 0,7712$$
$$u^{(4)} = 0,7712 + (0,77408 - 0,7712)F_x(1) = 0,773504$$
$$\overline{T}_x(1321) = \frac{0,7712 + 0,773504}{2} = 0,772352$$

Se observă că fiecare interval care urmează este conținut în precedentul. Această proprietate se foloseşte la descifrarea etichetei. O consecință nedorită a acestui proces este că intervalele devin din ce în ce mai mici şi este necesară o precizie cu atât mai ridicată, cu cât secvența devine mai lungă. Această problemă este depăşită printr-o strategie de rescalare, care va fi descrisă ulterior.

4.2.2. Descifrarea unei etichete

Exemplul 4.7.

Fie eticheta $\overline{T}_x(1321) = 0,772352$, obținută în Exemplul 4.6.

Pentru a efectua decodarea, se va simula codorul. Intervalul care conține eticheta este o submulțime a fiecărui interval obținut la codare. Se urmăreşte a se decoda elementele din secvență astfel încât valoarea etichetei să fie cuprinsă între limitele $l^{(k)}$ şi $u^{(k)}$ pentru fiecare *k*.

Se începe cu $l^{(0)} = 0$ şi $u^{(0)} = 1$. După decodarea primului element din secvență , x_1, limitele devin:

$$l^{(1)} = 0 + (1-0)F_x(x_1 - 1) = F_x(x_1 - 1);$$
$$u^{(1)} = 0 + (1-0)F_x(x_1) = F_x(x_1)$$

Aceasta înseamnă că intervalul care conține eticheta este $[F_x(x_1 - 1); F_x(x_1))$. Este necesară aflarea valorii lui x_1 pentru care valoarea etichetei 0,772352 cade în intervalul obținut.

Alegând

x_1=1, rezultă intervalul [0, 0,8),

x_1=2 rezultă intervalul [0,8, 0,82),

x_1=3 rezultă intervalul [0,82; 1).

Cum eticheta este conținută în primul interval, se decide $x_1 = 1$.

Se repetă procedura pentru al doilea element.

$$l^{(2)} = 0 + (0,8-0)F_x(x_2 - 1) = 0,8F_x(x_2 - 1);$$

$$u^{(2)} = 0 + (0,8-0)F_x(x_2) = F_x(x_2);$$

Dacă se alege

$x_2 = 1$, rezultă intervalul $[0; 0,64)$,

$x_2 = 2$, rezultă intervalul $[0,64; 0,656)$,

$x_2 = 3$, rezultă intervalul $[0,656; 0,8)$.

Eticheta este conținută în ultimul interval, aşa că se decide $x_2 = 3$.

Continuând, se obține

$$l^{(3)} = 0,656 + (0,8-0,656)F_x(x_3 - 1) = 0,656 + 0,144F_x(x_3 - 1);$$

$$u^{(3)} = 0,656 + (0,8-0,656)F_x(x_3) = 0,656 + 0,144F_x(x_3);$$

Se observă că odată cu creşterea lungimii secvenței, calculele devin laborioase. Pentru a calcula mai puțin, se observă următoarele: Din relațiile (4.21) şi (4.22) rezultă că

$$F_x(x_n - 1) = \frac{l^{(n)} - u^{(n)}}{l^{(n-1)} - u^{(n-1)}}, \tag{4.24}$$

respectiv,

$$F_x(x_n) = \frac{u^{(n)} - l^{(n-1)}}{u^{(n-1)} - l^{(n-1)}} \tag{4.25}$$

Ținând cont de (4.24) şi (4.25), rezultă atunci că mijlocul intervalului $[F_x(x_n - 1), F_x(x_n)]$ se poate calcula după cum urmează:

$$\frac{1}{2}\left[F_x(x_n)+F_x(x_n-1)\right]=\frac{l^{(n)}-l^{(n-1)}+u^{(n)}-l^{(n-1)}}{2\left[u^{(n-1)}-l^{(n-1)}\right]}=\frac{\dfrac{l^{(n)}+u^{(n)}}{2}-l^{(n-1)}}{u^{(n-1)}-l^{(n-1)}} \quad (4.26)$$

sau, ținând cont de (4.23), rezultă

$$\frac{1}{2}\left[F_x(x_n)+F_x(x_n-1)\right]=\frac{\overline{T}_x(x)-l^{(n-1)}}{u^{(n-1)}-l^{(n-1)}} \quad (4.27)$$

Astfel, pentru n=4, în cazul Exemplului 4.7 rezultă

$$l^{(4)}=0,7712+(0,77408-0,7712)F_x(x_4-1);$$

$$u^{(4)}=0,7712+(0,77408-0,7712)F_x(x_4);$$

Scăzând $l^{(3)}$ din ambele relații şi din etichetă şi împărțindu-le apoi la $u^{(3)}-l^{(3)}=0,001152$ se găseşte valoarea lui x_4 pentru care $\dfrac{\overline{T}_x(1321)-l^{(3)}}{u^{(3)}-l^{(3)}}$ cade în intervalul $[F_x(x_4-1);F_x(x_4))$. Aceasta este $x_4=1$. Cunoscând lungimea secvenței se ştie când se opreşte procedura.

În general, algoritmul de descifrare este următorul:

1. Se inițializează $l^{(0)}=0$ şi $u^{(0)}=1$;

2. Pentru fiecare *k* se determină $t^*=\dfrac{\overline{T}_x-l^{(k-1)}}{u^{(k-1)}-l^{(k-1)}}$;

3. Se determină valoarea lui x_k pentru care $F_x(x_k\text{-}1)\le t^*<F_x(x_k)$;

5. Se actualizează $u^{(k)}$ şi $l^{(k)}$;

6. Se continuă până la descifrarea întregii secvențe. Pentru a şti când s-a decodat întreaga secvență, fie decodorul cunoaşte lungimea secvenței şi procesul de decodare se opreşte la obținerea numărului de mesaje, fie se decodează un mesaj particular care indică sfârşitul transmisiei.

4.3. Generarea unui cod binar

Prin modul de obținere a etichetei, rezultă că aceasta este o reprezentare unică a secvenței. Înseamnă atunci că reprezentarea binară a etichetei determină un cuvânt de cod binar unic pentru secvență. Reprezentarea binară a unor valori din intervalul etichetei poate fi infinit de lungă, caz în care, deşi unic, codul nu este eficient. Apar următoarele întrebări:

1. Dacă se trunchiază reprezentarea binară, codul rezultat este încă unic?
2. Codul astfel realizat este eficient?
3. Cât de aproape este rata de entropie prin această reprezentare?

4.3.1. Unicitatea şi eficiența codului aritmetic

Valoarea etichetei $\overline{T}_x(x)$ a unui mesaj x este un număr în intervalul [0,1). Un cod binar pentru $\overline{T}_x(x)$ se poate obține considerând reprezentarea binară a numărului şi trunchiindu-l la $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$ biți, unde $\lceil x \rceil$ reprezintă cel mai mic număr mai mare sau egal cu x [86]. Baza logaritmului este 2.

Exemplul 4.8.

Fie sursa cu alfabetul $S = \{s_1, s_2, s_3, s_4\}$, cu probabilitățile $p(s_1) = \frac{1}{2}$, $p(s_2) = \frac{1}{4}$, $p(s_3) = p(s_4) = \frac{1}{8}$.

$\overline{T}_x(x)$ se obține cu relația (4.2). Reprezentarea binară a lui $\overline{T}_x(x)$ trunchiată la $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$ reprezintă codul etichetei şi se va nota cu $\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)}$.

Tabelul 4.3

Mesaj	F_x	$\overline{T}_x(s_i)$	Reprezentare binară	$l(s_i) = \left\lceil \log \frac{1}{p(s_i)} \right\rceil + 1$	Codul etichetei
s_1	0,5	0,25	01	2	01
s_2	0,75	0,625	101	3	101
s_3	0,875	0,8125	1101	4	1101
s_4	1	0,9375	1111	4	1111

În Tabelul 4.3 se prezintă, pentru fiecare mesaj al sursei, s_i, funcția de repartiție, F_x, calculată cu relația (4.1), valoarea zecimală a etichetei $\overline{T}_x(s_i)$, calculată cu relația (4.2), reprezentarea acesteia sub forma unei fracții binare, lungimea cuvântului de cod corespunzător şi cuvântul de cod ataşat etichetei. De exemplu, pentru primul mesaj s_1, furnizat cu probabilitatea $p(s_1) = \frac{1}{2}$, $F_x(s_1) = F_x(1) = 0,5$, eticheta acestuia este $\overline{T}_x(s_1) = F_x(0) + \frac{1}{2} p(x=1) = 0 + \frac{1}{2} \cdot 0,5 = 0,25$.

Reprezentarea acestei etichete sub forma unei fracții binare este 0 1, biți care reprezintă coeficienții părții zecimale a numărului $0,25 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$. Cum valoarea zecimală a etichetei a putut fi

reprezentată exact pe 2 biți, care reprezintă lungimea cuvântului de cod, aceştia constituie chiar cuvântul de cod al etichetei.

Se va arăta că un cod obținut în acest mod este unic decodabil. Întâi se arată că reprezentarea pe $l(x)$ biți a etichetei $\lfloor \overline{T}_x(x) \rfloor_{l(x)}$ este o reprezentare unică pentru $\overline{T}_x(x)$. Se reaminteşte că orice număr din intervalul $[F_x(x-1); F_x(x))$ poate fi un identificator unic. Prin urmare, dacă cuvântul de cod corespunzător mesajului x este conținut în intervalul $[F_x(x-1); F_x(x))$, atunci codul $\lfloor \overline{T}_x(x) \rfloor_{l(x)}$ este nesingular. Deoarece $\lfloor \overline{T}_x(x) \rfloor_{l(x)}$ se obține prin trunchierea reprezentării binare a lui $\overline{T}_x(x)$, rezultă că

$$0 \le \overline{T}_x(x) - \lfloor \overline{T}_x(x) \rfloor_{l(x)} < \frac{1}{2^{l(x)}} \tag{4.24}$$

sau, echivalent,

$$\overline{T}_x(x) < \lfloor \overline{T}_x(x) \rfloor_{l(x)} + \frac{1}{2^{l(x)}} \tag{4.24'}$$

Cum $\overline{T}_x(x) < F_x(x)$, rezultă

$$\lfloor \overline{T}_x(x) \rfloor_{l(x)} < F_x(x) \tag{4.25}$$

Pentru a arăta că $\lfloor \overline{T}_x(x) \rfloor_{l(x)} \ge F_x(x-1)$, se observă că

$$\frac{1}{2^{l(x)}} = \frac{1}{2^{\left\lceil \log \frac{1}{p(x)} \right\rceil + 1}} < \frac{1}{2^{\log \frac{1}{p(x)} + 1}} < \frac{1}{2 \frac{1}{p(x)}} < \frac{p(x)}{2} \tag{4.26}$$

Din relația (4.2) rezultă

$$\frac{p(x)}{2} = \overline{T}_x(x) - F_x(x-1) \tag{4.27}$$

Prin urmare,

$$\overline{T}_x(x) - F_x(x-1) > \frac{1}{2^{l(x)}} \tag{4.28}$$

sau, echivalent,

$$\overline{T}_x(x) > F_x(x-1) + \frac{1}{2^{l(x)}} \tag{4.28'}$$

Din (4.24') şi (4.28') rezultă

$$\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} + \frac{1}{2^{l(x)}} > \overline{T}_x > F_x(x-1) + \frac{1}{2^{l(x)}} \tag{4.29}$$

sau

$$\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} > F_x(x-1) \tag{4.29'}$$

Prin urmare, codul $\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)}$ este o reprezentare unică pentru $\overline{T}_x(x)$.

Pentru a arăta că se obține un cod unic decodabil, se va arăta că acesta este instantaneu, ştiut fiind faptul că un cod instantaneu este şi unic decodabil.

Dacă un număr *a* din intervalul [0,1) are reprezentarea pe *n* biți $[b_1 b_2 \ldots b_n]$, pentru ca orice alt număr *b* să aibă reprezentarea binară cu $[b_1 b_2 \ldots b_n]$ prefix, *b* trebuie să fie în intervalul $\left[a, a + \frac{1}{2^n}\right)$.

Dacă *x* şi *y* sunt două mesaje distincte, atunci $\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)}$ şi $\left\lfloor \overline{T}_x(y) \right\rfloor_{l(x)}$ aparţin la două intervale disjuncte $[F_x(x-1); F_x(x))$, respectiv, $[F_x(y-1); F_x(y))$. Prin urmare, dacă se va arăta că pentru orice mesaj *x*, intervalul $\left[\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)}, \left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} + \frac{1}{2^{l(x)}}\right)$ este conținut în întregime în intervalul $[F_x(x-1); F_x(x))$, atunci cuvântul de cod

pentru un mesaj nu poate fi prefix pentru cuvântul de cod al celuilalt mesaj.

Anterior s-a arătat că $\left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} > F_x(x-1)$. Se va arăta în continuare că

$$\left\lfloor \overline{T}_x(x) \right\rfloor + \frac{1}{2^{l(x)}} < F_x(x),$$

sau

$$F_x(x) - \left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} > \frac{1}{2^{l(x)}}$$

Folosind (4.24), (4.27) şi (4.26), rezultă

$$\begin{aligned} &F_x(x) - \left\lfloor \overline{T}_x(x) \right\rfloor_{l(x)} > F_x(x) - \overline{T}_x(x) = \\ &F_x(x) - F_x(x-1) - \frac{p(x)}{2} = \frac{p(x)}{2} > \frac{1}{2^{l(x)}} \end{aligned} \tag{4.30}$$

În concluzie, considerând reprezentarea binară a lui $\overline{T}_x(x)$ şi trunchiind-o la $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$ biți, se obține un cod unic decodabil.

În ceea ce priveşte eficiența unui cod astfel întocmit, se calculează lungimea medie a cuvintelor de cod pentru un cod aritmetic pentru o secvență de lungime *m*, adică

$$\begin{aligned} &\overline{l}_{A^{(m)}} = \sum p(x) l(x) = \sum p(x) \left(\left\lceil \log \frac{1}{p(x)} \right\rceil + 1 \right) \leq \\ &\sum p(x) \left(\log \frac{1}{p(x)} + 1 + 1 \right) = -\sum p(x) \log p(x) + 2 \sum p(x) = H(X^m) + 2 \end{aligned} \tag{4.31}$$

Deoarece $\overline{l}_{A^{(m)}}$ este întotdeauna mai mare decât entropia $H(X^m)$, rezultă

$$H(X^{(m)}) \le \overline{l}_{A^{(m)}} \le H(X^{(m)}) + 2 \tag{4.32}$$

Lungimea medie pe mesaj, $\overline{l}_A$, sau rata codului aritmetic este $\frac{\overline{l}_{A^{(m)}}}{m}$. Divizând cu *m* relația (4.32), rezultă

$$\frac{H(X^m)}{m} \le \overline{l}_A \le \frac{H(X^m)}{m} + \frac{2}{m} \tag{4.33}$$

Deoarece $H(X^m) = mH(X)$, rezultă

$$H(X) \le \overline{l}_A \le H(X) + \frac{2}{m} \tag{4.34}$$

Se observă că rata codului va fi cu atât mai apropiată de entropie, cu cât lungimea secvenței este mai mare.

4.3.2. Implementarea algoritmului

Algoritmii de codare şi decodare descrişi nu sunt implementabili în maniera descrisă. Procedura descrisă s-a raportat la intervalul [0,1), în care există o infinitate de numere. În practică, numărul numerelor ce pot fi reprezentate unic pe un calculator este limitat de numărul de biți folosiți. Fie $l^{(n)}$ şi $u^{(n)}$ din Exemplul 4.6. Cu cât *n* creşte, cu atât aceste valori devin mai apropiate, ceea ce înseamnă că, pentru a reprezenta unic toate subintervalele, trebuie crescută precizia reprezentării. Într-un sistem cu precizie finită, cele două valori sunt constrânse să conveargă. Pentru evitarea acestei situații, este necesar a rescala intervalul etichetei $[l^{(n)}, u^{(n)}]$. Acest lucru trebuie făcut astfel încât să se păstreze informația ce se transmite. În acest scop se realizează o codare progresivă sau incrementală, adică se transmit porțiuni din cuvintele de cod, pe măsură

ce se inspectează secvența, în loc să se aştepte până se observă întreaga secvență înaintea transmiterii primului bit.

Algoritmul ce urmează a fi descris ține seama de *rescalarea sincronizată şi codarea incrementală*.

Cu cât intervalul devine mai îngust, este foarte posibil ca intervalul ce conține eticheta să se găsească fie în jumătatea inferioară, fie în cea superioară a intervalului [0,1), fapt ce determină ca cel mai semnificativ bit (MSB) al etichetei să fie complet determinat. Dacă eticheta este mai mare decât 0,5, primul bit al etichetei este 1, iar dacă este mai mică decât 0,5, acesta este 0.

În această situație se poate indica decodorului cărei jumătăți a intervalului aparține eticheta, prin trimiterea unui 1 pentru jumătatea superioară şi 0 pentru cea inferioară. Transmisia lui 1 sau 0 reprezintă primul bit al etichetei.

Odată ce codorul şi decodorul ştiu care jumătate a intervalului unitate conține eticheta, se poate ignora cealaltă jumătate.

Deoarece se foloseşte aritmetica de precizie finită, se transformă jumătatea de interval care conține eticheta în întregul interval [0,1). Aceste transformări sunt de forma

$$E_1 : [0; 0,5) \to [0,1), \quad E_1 = 2x; \tag{4.35}$$

pentru jumătatea inferioară a intervalului şi

$$E_2 : [0,5;1) \to [0,1), \; E_1 = 2(x - 0,5); \tag{4.36}$$

pentru jumătatea superioară a intervalului.

Odată efectuată una din aceste transformări, se pierde toată informația despre MSB. Aceasta nu mai contează, câtă vreme acest bit a fost trimis deja spre decodor.

Procesul poate fi continuat generând un alt bit al etichetei de câte ori intervalul etichetei este restricționat fie la jumătatea superioară, fie la cea inferioară a intervalului. Procesul de generare a biților etichetei

fără a se aştepta observarea întregii secvenţe se numeşte codare incrementală.

Exemplul 4.9.

Generarea unei etichete cu scalare.

Se reia Exemplul 4.6, în care se codează secvenţa 1321, cu modelul de probabilitate $p(s_1) = 0,8,\ p(s_2) = 0,02,\ p(s_3) = 0,18$.

Iniţializând $u^{(0)} = 1$ şi $l^{(0)} = 0$, primul element al secvenţei, 1, are ca rezultat actualizarea limitelor

$$l^{(1)} = 0 + (1-0)\cdot 0 = 0$$

$$u^{(1)} = 0 + (1-0)\cdot 0,8 = 0,8$$

Intervalul [0, 0,8) nu este inclus în jumătatea superioară sau inferioară a intervalului unitate, aşa că se continuă.

Următorul element este 3. Acesta are ca rezultat actualizarea:

$$l^{(2)} = 0 + (0,8-0)F_x(2) = 0,8\cdot 0,82 = 0,656$$

$$u^{(2)} = 0 + (0,8-0)F_x(3) = 0,8\cdot 1 = 0,8$$

Intervalul [0,656, 0,8) este în întregime cuprins în jumătatea superioară a intervalului unitate, astfel încât se transmite 1 şi se rescalează.

$$l^{(2)} = 2(0,656-0,5) = 0,312$$

$$u^{(2)} = 2(0,8-0,5) = 0,6$$

Al treilea element, 2, are ca rezultat reactualizarea

$$l^{(3)} = 0,312 + (0,6-0,312)\cdot 0,8 = 0,5424$$

$$u^{(3)} = 0,312 + (0,6-0,312)\cdot 0,82 = 0,54816$$

Intervalul etichetei [0,5424; 0,54816) este conţinut în întregime în jumătatea superioară a intervalului unitate. Se transmite 1 şi se rescalează cu transformarea E_2, rezultând

$$l^{(3)} = 2(0,5424 - 0,5) = 0,0848$$

$$u^{(3)} = 2(0,54816 - 0,5) = 0,09632$$

Acest interval aparţine în întregime jumătăţii inferioare a intervalului unitate. Se transmite 0 şi se rescalează cu transformarea E_1, rezultând

$$l^{(3)} = 2 \cdot 0,0848 = 0,1696$$

$$u^{(3)} = 2 \cdot 0,09632 = 0,19624$$

Intervalul este în întregime cuprins în jumătatea inferioară a intervalului unitate, aşa că se transmite 0 şi iar se rescalează.

$$l^{(3)} = 2 \cdot 0,1696 = 0,3392$$

$$u^{(3)} = 2 \cdot 0,19624 = 0,38528$$

Intervalul se află iar în jumătatea inferioară, aşa că se transmite 0 şi iar se rescalează.

$$l^{(3)} = 2 \cdot 0,3392 = 0,6784$$

$$u^{(3)} = 2 \cdot 0,38528 = 0,77056$$

Rezultând acum un interval în jumătatea superioară a intervalului unitate, se transmite 1 şi se rescalează cu E_2.

$$l^{(3)} = 2(0,6784 - 0,5) = 0,3568$$

$$u^{(3)} = 2(0,77056 - 0,5) = 0,54112$$

La fiecare etapă se transmite cel mai semnificativ bit (MSB) când este acelaşi atât pentru limita superioară şi inferioară a intervalului etichetei. Prin urmare, prin transmiterea MSB a limitelor intervalului etichetei, se transmite de fapt reprezentarea binară a etichetei. Operaţia de rescalare este văzută ca o deplasare spre stânga. Astfel, bitul următor celui mai semnificativ bit devine cel mai semnificativ bit.

Continuând cu ultimul element, limitele inferioară şi superioară ale intervalului ce conține eticheta sunt

$$l^{(4)} = 0,3568 + (0,54112 - 0,3568)F_x(0) = 0,3568$$

$$u^{(4)} = 0,3568 + (0,54112 - 0,3568)F_x(1) = 0,504256$$

În acest moment, dacă se doreşte oprirea codării, trebuie informat receptorul despre starea finală a etichetei. Aceasta se face transmițând valoarea binară a oricărui număr din intervalul final al etichetei. În general, aceasta se consideră a fi $l^{(n)}$. În acest exemplu este convenabil să se folosească valoare de 0,5, cu reprezentarea binară 100…, deci se va transmite 1 urmat de atâția de 0, câți sunt necesari pentru lungimea cuvântului.

Se observă că mărimea intervalului etichetei este de 64 de ori mai mare decât în cazul utilizării algoritmului nemodificat. După cum se va vedea, biții care au fost transmişi la fiecare transformare reprezintă eticheta însăşi, care satisface cerința codării incrementale (progresive).

Secvența binară obținută în timpul procesului de codare din Exemplul 4.6 este 1100011. Numărul binar .1100011 corespunde numărului zecimal 0,7734375 care aparține intervalului final al etichetei din Exemplul 4.6. Acest lucru se poate folosi pentru decodarea secvenței. Ca şi codarea, decodarea va fi efectuată tot incremental. Aceasta ridică următoarele probleme: cum se începe decodarea, cum se continuă şi cum se opreşte.

Pentru a începe decodarea, este nevoie de informație suficientă pentru a decoda primul mesaj fără ambiguitate. Pentru aceasta, numărul de biți recepționați ar trebui să reprezinte un interval mai mic decât cel mai mic interval în care se poate găsi eticheta. Pe baza intervalului minim al etichetei, se poate determina numărul de biți necesari începerii

decodării. Odată începută decodarea, pentru a o continua, trebuie simulat algoritmul de codare.

Exemplul 4.10.

Fie secvența din Exemplul 4.9, adică 110001100….0.

Cel mai mic interval al etichetei este [0,8; 0,82), de lungime 0,2. Prin urmare, pentru o decodare fără ambiguități sunt necesari k biți, astfel încât $2^{-k} < 0,02$, adică k=6.

Ca şi la codor, se pleacă cu inițializarea $u^{(0)} = 1$ şi $l^{(0)} = 0$.

Primii 6 biți corespund unei valori a etichetei de 0,765625, care înseamnă că primul element este 1, care are ca rezultat actualizarea

$$l^{(1)} = 0 + (1-0)\cdot 0 = 0$$

$$u^{(1)} = 0 + (1-0)\cdot 0,8 = 0,8$$

Intervalul [0; 0,8) nu aparține vreunei jumătăți a intervalului unitate, aşa că se continuă. Eticheta 0,765625 cade în partea superioară a 18% din intervalul [0; 0,8), prin urmare al 2-lea element din secventa este 3. Se actualizează intervalul

$$l^{(2)} = 0 + (0,8-0)F_x(2) = 0,656$$

$$u^{(2)} = 0 + (0,8-0)F_x(3) = 0,8$$

Intervalul este conținut în jumătatea superioară a intervalului unitate. Aceasta înseamnă că la codor s-a transmis 1 şi s-a rescalat. La decodor se va deplasa un 1 afară din bufferul de la recepție şi se introduce următorul bit în buffer pentru a forma cei 6 biți ai etichetei, adică 100011. Reactualizând intervalul etichetei, se obține

$$l^{(2)} = 2(0,656-0,5) = 0,312$$

$$u^{(2)} = 2(0,8-0,5) = 0,6$$

Deplasarea bitului spre stânga furnizează eticheta 0,546875. Comparând această valoare cu intervalul etichetei se observă că acesta cade în domeniul 80-82% din interval, aşa că se decodează următorul element din secvență ca fiind 2. Se pot actualiza ecuațiile pentru intervalul etichetei de forma

$$l^{(3)} = 0,312 + (0,6 - 0,312)F_x(1) = 0,5424$$

$$u^{(3)} = 0,312 + (0,6 - 0,312)F_x(2) = 0,54816$$

Cum intervalul este conținut în întregime în jumătatea superioară a intervalului unitate, se rescalează cu E_2, rezultând:

$$l^{(3)} = 2(0,5424 - 0,5) = 0,0848$$

$$u^{(3)} = 2(0,54816 - 0,5) = 0,09635$$

Se deplasează din nou afară spre stânga un bit din etichetă şi se introduce din dreapta un bit. Acum eticheta este 000110. Intervalul este conţinut în jumătatea inferioară a intervalului unitate. Se scalează cu E_1 şi deplasând încă un bit, limitele intervalului devenind

$$l^{(3)} = 2 \cdot 0,0848 = 0,1696$$

$$u^{(3)} = 2 \cdot 0,09635 = 0,19264$$

Eticheta devine 001100. Intervalul este încă conținut în jumătatea inferioară, aşa că se mai deplasează un zero, obținând eticheta 011000 şi se efectuează o altă scalare

$$l^{(3)} = 2 \cdot 0,1696 = 0,3392$$

$$u^{(3)} = 2 \cdot 0,19264 = 0,38528$$

Intervalul etichetei este iar în jumătatea inferioară a intervalului unitate, aşa că se mai deplasează un zero, obținând eticheta 110000 şi se mai scalează o dată.

$$l^{(3)} = 2 \cdot 0,3392 = 0,6784$$

$$u^{(3)} = 2 \cdot 0,38528 = 0,77056$$

Intervalul fiind conținut în întregime în jumătatea superioară a intervalului unitate, se deplasează un 1 din etichetă, rămânând 10000 şi se scalează cu E_2.

$$l^{(3)} = 2 \cdot (0,6784 - 0,5) = 0,3568$$

$$u^{(3)} = 2 \cdot (0,77056 - 0,5) = 0,54112$$

Se compară valoarea etichetei 100000 (adică 0,5) cu intervalul etichetei şi se observă că aceasta cade în primele 80 de procente ale intervalului (0,8(0,54112-0,3568)), aşa încât se decodează 1.

În exemplul precedent intervalul etichetei a fost cuprins fie în jumătatea superioară, fie în cea inferioară a intervalului unitate. Se observă că rescalarea se poate aplica în mod repetat, până ce $l^{(n)} < 0,5 \le u^{(n)}$. Scopul rescalării este acela de a evita situația în care, cu fiecare mesaj al sursei, limitele intervalului devin prea apropiate. În cazul aritmeticii de precizie finită folosită în tehnica de calcul este posibil ca intervalele să fie reprezentate de aceeaşi valoare.

În continuare se va analiza cazul în care intervalul etichetei conține mijlocul intervalului unitate. În acest scop se verifică dacă intervalul etichetei este conținut în intervalul [0,25; 0,75), adică, dacă $0,25 \le l^{(n)} < 0,5 \le u^{(n)} < 0,75$. În acest caz reprezentarea binară a oricărui număr din intervalul considerat începe cu 0,01... sau 0,10... , al doilea bit al reprezentării diferind de primul.

În acest caz, intervalul etichetei se rescalează cu ajutorul transformării E_3 definită astfel:

$$E_3 : [0,25; 0,75) \to [0,1); \quad E_3(x) = 2(x - 0,25) \tag{4.37}$$

Prin înmulțirea cu 2, transformarea E_3 deplasează spre stânga biții reprezentării binare a numărului respectiv. Operația de scalare dată de transformarea E_3 înlătură cel de-al doilea bit al reprezentării binare a

numărului x, astfel încât, codul aritmetic corespunzător numărului după scalare se obține din cel al numărului nescalat, prin înlăturarea celui de-al doilea bit. Acesta este compensat ulterior prin generarea unui bit adițional, imediat după următoarea ieşire. Acest bit adițional este complementar ieşirii căreia îi urmează.

Limitele intervalului etichetei se transformă în $2 \cdot l^{(n)} - 0,5$, respectiv, $2 \cdot u^{(n)} - 0,5$.

Se reaminteşte că în cazul când intervalul etichetei nu conține mijlocul intervalului unitate s-a folosit un 1 pentru a transmite informații despre transformarea E_2 şi un 0 pentru transformarea E_1. Când se efectuează o transformare E_3, nu se transmite nici o informație către decodor, în schimb se înregistrează acest lucru la codor, prin incrementarea unui contor.

După o transformare E_3 pot apărea următoarele situații:

1 - la actualizarea intervalului etichetei, acesta este conținut în jumătatea superioară a intervalului unitate;

2 - la actualizarea intervalului etichetei, acesta este conținut în jumătatea inferioară a intervalului unitate;

3 - urmează n-1 transformări E_3.

În primul caz, codorul va transmite un bit de 1, corespunzător transformării E_2, urmat de un bit de 0, care ar corespunde unei transformări E_1. Acest lucru se justifică prin faptul că efectul transformării E_3 urmate de E_2 este echivalent cu transformarea E_2 urmată de E_1, adică

$$E_2(E_3(x)) = E_1(E_2(x))$$

Într-adevăr,

$$E_2(E_3(x)) = E_2(2(x-0,25)) = 2(2(x-0,25)-0,5) = 4 \cdot x - 2$$

şi

$$E_1(E_2(x)) = E_1(2(x-0,5)) = 2\cdot 2(x-0,5) = 4\cdot x - 2$$

În cazul al doilea, codorul va transmite un bit de 0, corespunzător transformării E_1, urmat de un bit de 1, care ar corespunde unei transformări E_2. Acest lucru se justifică prin faptul că efectul transformării E_3 urmate de E_1 este echivalent cu transformarea E_1 urmată de E_2, adică

$$E_1(E_3(x)) = E_2(E_1(x))$$

Într-adevăr,

$$E_1(E_3(x)) = E_1(2(x-0,25)) = 2(2x-0,5) = 4\cdot x - 1$$

şi

$$E_2(E_1(x)) = E_2(2x) = 2(2x-0,5) = 4\cdot x - 1$$

În cazul al treilea, pot apărea două situații:

a) după cele *n* transformări E_3 urmează o transformare E_2;

b) după cele *n* transformări E_3 urmează o transformare E_1;

În cazul a), codorul va transmite un bit de 1 urmat de *n* biți de 0. Aceasta se justifică prin faptul că efectul a *n* transformări E_3 urmate de o transformare E_2 este echivalent cu transformarea E_2 urmată de *n* transformări E_1, adică

$$E_2\underbrace{(E_3(E_3...(E_3(x)))}_{n\ ori} = \underbrace{E_1(E_1(...E_1}_{n\ ori}(E_2(x)))$$

Într-adevăr,

$$E_2\underbrace{(E_3(E_3...(E_3(x)))}_{n\ ori} = E_2(2^n\cdot x - 0,5\sum_{k=0}^{n-1}2^k) = 2^{n+1}\cdot x - 2^n$$

şi

$$\underbrace{E_1(E_1(...E_1}_{n\ ori}(E_2(x))) = \underbrace{E_1(E_1(...E_1}_{n\ ori}(2\cdot x - 1))) = 2^{n+1}\cdot x - 2^n$$

În cazul b), codorul va transmite un bit de 0 urmat de *n* biți de 1. Aceasta se justifică prin faptul că efectul a *n* transformări E_3 urmate de

o transformare E_1 este echivalent cu transformarea E_1 urmată de n transformări E_2, adică

$$E_1(\underbrace{E_3(E_3...(E_3(x))}_{n\ ori}) = \underbrace{E_2(E_2(...(E_2}_{n\ ori}(E_1(x)))$$

Într-adevăr,

$$E_1(\underbrace{E_3(E_3...(E_3(x))}_{n\ ori}) = E_1(2^n \cdot x - 0,5\sum_{k=0}^{n-1} 2^k) = 2^{n+1} \cdot x - 2^n + 1$$

şi

$$\underbrace{E_2(E_2(...(E_2}_{n\ ori}(E_1(x))) = \underbrace{E_2(E_2(...(E_2}_{n\ ori}(2 \cdot x))) = 2^{n+1} \cdot x - 2^n + 1$$

Din cele prezentate rezultă că la codare trebuie să existe un contor care să înregistreze numărul de transformări E_3 . Contorul se decrementează când, după una sau mai multe transformări succesive E_3, la reactualizarea intervalului etichetei, apare o transformare E_1 sau E_2.

Transformările E_3 sunt aplicate şi la decodor, când intervalul etichetei este conținut în intervalul [0,25; 0,75), cu $l^{(n)} < 0,5 \le u^{(n)}$.

4.3.3. Implementarea cu numere întregi

În cazul implementării cu numere întregi, procedurile dezvoltate pentru intervalul [0,1) se adaptează la aritmetica cu numere întregi. Acestea vor fi reprezentate folosind coduri binare.

Implementarea codorului

Fie m numărul de biți folosiți în reprezentarea numerelor întregi. Valorile de interes din intervalul [0,1) (etichetele şi capetele intervalelor) se transformă în domeniul celor 2^m cuvinte binare. Capătul 0 al intervalului se transformă în $\underbrace{000\ldots00}_{m\ ori}$, iar 1 în $\underbrace{11\ldots11}_{m\ ori}$, valoarea

0,5 în $1\underbrace{0...0}_{m-1\ ori}$. Ecuațiile de actualizare rămân similare cu ecuațiile (4.21) şi (4.22), dar, deoarece se lucrează cu aritmetica numerelor întregi, sunt necesare anumite modificări, după cum urmează:

Se notează cu n_j numărul de apariții ale mesajului j într-o secvență de lungime $N = \sum_j n_j$. Funcția de repartiție $F_x(k)$ poate fi estimată cu relația

$$F_x(k) = \frac{\sum_{i=1}^{k} n_i}{N} \tag{4.38}$$

Se notează sumele parțiale

$$N(k) = \sum_{i=1}^{k} n_i \,. \tag{4.39}$$

Ecuațiile (4.21) şi (4.22) pot fi scrise sub forma:

$$l^{(n)} = l^{(n-1)} + \left\lfloor (u^{(n-1)} - l^{(n-1)} + 1)\frac{N(x_n - 1)}{N} \right\rfloor \tag{4.40}$$

$$u^{(n)} = l^{(n-1)} + \left\lfloor (u^{(n-1)} - l^{(n-1)} + 1)\frac{N(x_n)}{N} \right\rfloor - 1 \tag{4.41}$$

unde x_n este al n-lea mesaj ce urmează a fi codat, $\lfloor x \rfloor$ este cel mai mare întreg mai mic sau egal cu x, iar adunarea şi scăderea lui 1 sunt necesare pentru a adapta relațiile la aritmetica cu numere întregi.

Din cauza modului de transformare a capetelor intervalului, când atât $l^{(n)}$, cât şi $u^{(n)}$ sunt fie în jumătatea superioară, fie în cea inferioară a intervalului, bitul cel mai semnificativ (MSB) al lui $l^{(n)}$ şi $u^{(n)}$ va fi acelaşi. Dacă MSB este 1, atunci întreaga etichetă este cuprinsă în jumătatea superioară a intervalului [00…00, 11…..1]. Dacă MSB este 0, intervalul etichetei este în jumătatea inferioară a intervalului. Aplicarea

lui E_1 şi E_2 constă în a deplasa spre stânga secvența şi de a introduce un 0 pentru $l^{(n)}$ şi un 1 pentru $u^{(n)}$ în poziția celui mai puțin semnificativ bit (LSB).

Pentru fixarea ideilor, se presupune, că m=6, $u^{(n)} = 54$ şi $l^{(n)} = 33$. Reprezentările lui $l^{(n)}$ şi $u^{(n)}$ sunt 110110 şi, respectiv, 100001. Se observă că pentru ambele limite MSB este egal cu 1. Urmând procedura descrisă mai sus, se deplasează spre stânga secvența cu un bit şi se completează cu 1 pe LSB pentru $u^{(n)}$ şi 0 pentru $l^{(n)}$, obținând noile valori 101101 (=45) şi 000010(=2). Aceasta este echivalent cu efectuarea transformării E_2. Transformarea E_1 se efectuează folosind aceleaşi operații.

Transformarea E_3 se aplică atunci când intervalul etichetei cade în jumătatea de mijloc a intervalului [00…0, 11…1], cu $l^{(n)} < 10...0$ şi $u^{(n)} \geq 10...0$. Aceasta înseamnă că cei mai semnificativi biți pentru $l^{(n)}$ şi $u^{(n)}$ sunt 0 şi, respectiv, 1.

Pentru realizarea transformării E_3, se urmăreşte bitul următor celui mai semnificativ bit al lui $u^{(n)}$ şi $l^{(n)}$, care pentru $u^{(n)}$ este 0 şi pentru $l^{(n)}$ este 1. Aşa cum s-a precizat, operația de înmulțire cu 2 din transformarea E_3 înseamnă deplasarea spre stânga cu un bit a secvenței, iar operația de scădere a unui sfert din intervalul de valori echivalează cu complementarea celui de-al doilea bit. Prin urmare, pentru a implementa transformarea E_3, se complementează al doilea bit din $u^{(n)}$ şi $l^{(n)}$ şi se deplasează spre stânga, introducând un 1 în $u^{(n)}$ şi un 0 în $l^{(n)}$ pe poziția LSB. De asemenea, se urmăreşte numărul de transformări E_3 prin mărimea *Contor* 3.

Exemplul 4.11.

Se codează secvența 1321 cu parametrii din Tabelul 4.4.

Întâi se alege lungimea cuvântului, m . Se observă că $N(1)$ şi $N(2)$ diferă numai printr-o unitate. Se reaminteşte că valoarea lui N va fi translată la capetele subintervalelor.

Tabelul 4.4.

$n_1 = 40$	$N(0) = 0$	$Contor3 = 0$
$n_2 = 1$	$N(1) = 40$	
$n_3 = 9$	$N(2) = 41$	
$N = 50$	$N(3) = 50$	

Trebuie urmărit ca valoarea selectată pentru lungimea cuvântului să permită reprezentarea unui domeniu suficient pentru a reprezenta cea mai mică diferență dintre capetele intervalelor. Întotdeauna se recurge la rescalare când intervalul devine prea mic. Pentru a asigura disjuncția între capetele intervalului, trebuie urmărit ca toate valorile din domeniul $0 \div N$ să fie reprezentate în mod unic, chiar când nu se efectuează rescalarea intervalului.

Aşa cum s-a precizat, se foloseşte rescalarea, dacă limitele intervalului etichetei sunt cuprinse în jumătatea inferioară, superioară sau de mijloc a domeniului de valori. Există însă situații când fie limita inferioară a intervalului etichetei este mai mică decât un sfert din interval, iar cea superioară mai mare sau egală cu jumătate din interval, fie când limita inferioară a intervalului etichetei este mai mică decât jumătate din interval, iar cea superioară mai mare sau egală cu trei sferturi din interval, cazuri în care nu se poate aplica nici una din rescalări. În acest caz se consideră următorul mesaj, care micşorează limitele intervalului etichetei, fapt care, în aritmetica cu numere întregi ar putea conduce la aceeaşi valoare. Cel mai mic interval pentru care nu

se poate aplica rescalarea este de un sfert din întregul interval. În concluzie, cel mai mic interval $[l^{(n)}, u^{(n)}]$ posibil în care nu se poate folosi rescalarea este un sfert din domeniul total disponibil de 2^m valori. Deci m trebuie să fie suficient de mare pentru a reprezenta unic mulțimea valorilor dintre 0 şi $4 \cdot N$, adică $2^m = R > 4 \cdot N$.

Pentru acest exemplu, aceasta înseamnă că domeniul intervalului total trebuie să fie mai mare ca 200. m=8 satisface această cerință. Cu acest m se obține

$$l^{(0)} = 0 = (00000000)_2 \tag{4.42}$$

$$u^{(0)} = 255 = (11111111)_2 \tag{4.43}$$

Primul element al secvenței este 1. Cu relațiile (4.40) şi (4.41) rezultă

$$l^{(1)} = 0 + \left\lfloor \frac{256 \cdot N(0)}{50} \right\rfloor = 0 = (00000000)_2 \tag{4.44}$$

$$u^{(1)} = 0 + \left\lfloor \frac{256 \cdot N(1)}{50} \right\rfloor - 1 = 203 = (11001011)_2 \tag{4.45}$$

Următorul element este 3,

$$l^{(2)} = 0 + \left\lfloor \frac{204 \cdot N(2)}{50} \right\rfloor = 167 = (10100111)_2 \tag{4.46}$$

$$u^{(2)} = 0 + \left\lfloor \frac{204 \cdot N(3)}{50} \right\rfloor - 1 = 203 = (11001011)_2 \tag{4.47}$$

MSB pentru $l^{(2)}$ şi $u^{(2)}$ este 1. Prin urmare, se deplasează această valoare afară şi se transmite la decodor. Ceilalți biți sunt deplasați cu o unitate, obținând:

$$l^{(2)} = (01001110)_2 = 78 \tag{4.48}$$

$$u^{(2)} = (10010111)_2 = 151 \tag{4.49}$$

Se observă că MSB ai celor două limite sunt diferiți, al doilea bit al lui $u^{(2)}$ este 0, iar al lui $l^{(2)}$ este 1. Aceasta este condiția pentru transformarea E_3. Se complementează al doilea bit al ambelor limite şi se deplasează un bit spre stânga, introducând un 0 în LSB pentru $l^{(2)}$ şi un 1 în LSB pentru $u^{(2)}$, rezultând:

$$l^{(2)} = (00011100)_2 = 28 \tag{4.50}$$

$$u^{(2)} = (10101111)_2 = 175 \tag{4.51}$$

Se incrementează $Contor3$ cu 1.

Următorul element este 2. Actualizând limitele, rezultă:

$$l^{(3)} = 28 + \left\lfloor \frac{148 \cdot N(1)}{50} \right\rfloor = 146 = (10010010)_2 \tag{4.52}$$

$$u^{(3)} = 28 + \left\lfloor \frac{148 \cdot N(2)}{50} \right\rfloor - 1 = 148 = (10010100)_2 \tag{4.53}$$

Cei doi MSB sunt identici şi egali cu 1, ceea ce corespunde transformării E_2, aşa încât se deplasează afară un bit de 1 care se transmite la decodor. Cum $Contor3 = 1$, se transmite un bit care este opus celui corespunzător transformării E_2, adică un 0, pentru a simula transformarea E_3 la decodor şi se decrementează $Contor3$ la 0. Noile limite sunt

$$l^{(3)} = (00100100)_2 = 36 \tag{4.54}$$

$$u^{(3)} = (00101001)_2 = 41 \tag{4.55}$$

Cei doi MSB ai limitelor *u* şi *l* sunt ambii 0, astfel încât acesta se deplasează afară şi se transmite 0 la decodor.

$$l^{(3)} = (01001000)_2 = 72 \tag{4.56}$$

$$u^{(3)} = (01010011)_2 = 83 \tag{4.57}$$

Din nou ambii MSB sunt 0, astfel încât acesta se deplasează afară (spre stânga) şi se transmite 0 la decodor. Limitele devin:

$$l^{(3)} = (10010000)_2 = 144 \tag{4.58}$$

$$u^{(3)} = (10100111)_2 = 167 \tag{4.59}$$

Acum ambii MSB sunt 1, astfel încât îl deplasăm spre stânga şi îl transmitem decodorului. Limitele devin:

$$l^{(3)} = (00100000)_2 = 32 \tag{4.60}$$

$$u^{(3)} = (01001111)_2 = 79 \tag{4.61}$$

Din nou cei doi MSB sunt identici, astfel încât se deplasează afară 0 şi se transmite acesta la decodor.

$$l^{(3)} = (01000000)_2 = 64 \tag{4.62}$$

$$u^{(3)} = (10011111)_2 = 159 \tag{4.63}$$

Acum cei doi MSB sunt diferiți. Al doilea MSB pentru $l^{(3)}$ este 1, iar pentru $u^{(3)}$ este 0. Aceasta este condiția pentru transformarea E_3. Aplicând E_3 prin complementarea celui de-al doilea bit al ambelor limite şi deplasând un bit spre stânga, se obține:

$$l^{(3)} = (00000000)_2 = 0 \tag{4.64}$$

$$u^{(3)} = (10111111)_2 = 191 \tag{4.65}$$

Următorul mesaj din secvență, ce urmează a fi codat, este 1. Prin urmare:

$$l^{(4)} = 0 + \left\lfloor \frac{192 \cdot N(0)}{50} \right\rfloor = 0 = (00000000)_2 \tag{4.66}$$

$$u^{(4)} = 0 + \left\lfloor \frac{192 \cdot N(1)}{50} \right\rfloor - 1 = 152 = (10011000)_2 \tag{4.67}$$

Secvența binară generată este 1100010.

Implementarea decodorului

1. Se inițializează $l^{(0)} = 00\ldots0, u^{(0)} = 11\ldots1$.

2. Pentru fiecare *k* se găseşte

$t^* = (\overline{T}_x(x) - l^{(k-1)} + 1)\big/(u^{(k-1)} - l^{(k-1)} + 1)$, unde $\overline{T}_x(x)$ este valoarea etichetei.

3. Se găseşte valoarea lui $x^{(k)}$ pentru care

$$\frac{N(x^{(k)} - 1)}{N} \le t^* < \frac{N(x^{(k)})}{N}$$

4. Se actualizează $u^{(k)}$ şi $l^{(k)}$.

5. Se continuă până la decodarea întregii secvențe.

4.4. Comparație între codarea Huffman şi codarea aritmetică

În acest capitol a fost descrisă codarea aritmetică, care deşi mai complicată decât codarea Huffman, permite codarea secvențelor de mesaje. Pentru a face comparație între codarea aritmetică şi Huffman, se începe prin a folosi codarea aritmetică pentru surse pentru care se cunoaşte codul Huffman.

Se reia Exemplul 4.8, pentru care se calculează lungimea medie:

$$\bar{l} = 2 \cdot 0,5 + 3 \cdot 0,25 + 4 \cdot 0,125 + 4 \cdot 0,125 = 2,75 \text{ biți/mesaj} \qquad (4.68)$$

Entropia acestei surse este 1,75 biți/mesaj şi rata codului Huffman atinge această entropie, astfel încât codarea aritmetică nu este o alegere potrivită, dacă se intenționează a se coda câte un mesaj. Se repetă exemplul, cu extensia de ordinul doi.

Exemplul 4.12.

Dacă se codează extensia de ordinul doi rezultă codul prezentat în Tabelul 4.5.

Tabelul 4.5.

Mesaj	$p(x)$	$\overline{T}_x(x)$	$\overline{T}_x(x)$ în binar	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codul rezultat
11	0,25	0,125	0,001	3	001
12	0,125	0,3125	0,0101	4	0101
13	0,625	0,40625	0,01101	5	01101
14	0,625	0,46875	0,01111	5	01111
…	…	…	…	…	…
44	0,015625	0,984375	0,1111111	7	1111111

În acest caz lungimea medie pe mesaj este 2,25 biți/mesaj, care este mai mică decât 2,75, dar încă nu față de 1,75. Odată cu creşterea ordinului extensiei rezultatele devin mai bune.

Se pune problema cât trebuie să crească ordinul extensiei, astfel încât codarea aritmetică să fie superioară codării Huffman. Limitele ratei de codare pentru codarea aritmetică, l_A, şi Huffman, l_H, satisfac inegalitățile

$$H(X) \le l_A \le H(x) + \frac{2}{m}$$

$$H(x) \le l_H \le H(x) + \frac{1}{m}$$

Din compararea limitelor celor două rate rezultă că totdeauna prin codul Huffman se obține o limită superioară a ratei mai mică decât în cazul codului aritmetic. Pe de altă parte, se reaminteşte că pentru a

genera un cuvânt de cod pentru un mesaj compus, de lungime *m*, cu ajutorul codului Huffman trebuie construite toate cele k^m cuvinte de cod posibile (*k* reprezintă numărul de mesaje ce trebuie codate, iar *m*, ordinul memoriei). În cazul codării aritmetice nu este nevoie a construi toate cuvintele de cod, putându-se obține cuvântul de cod pentru o etichetă corespunzătoare unei secvențe de mesaje.

În practică, *m* poate fi crescut pentru codarea arimetică, dar nu şi pentru codarea Huffman. Excepție fac sursele ale căror probabilități sunt puteri negative ale lui 2, caz în care codarea Huffman pe mesaje individuale atinge entropia, care nu se va obține folosind codarea aritmetică, indiferent de lungimea secvenței de mesaje pe care s-ar efectua codarea.

În general, pentru codul Huffman realizat pe mesaje individuale, rata este mai mică decât $H(S)+0,086+p_{\max}$ [25]. Dacă probabilitățile de furnizare a mesajelor nu sunt prea dezechilibrate, $p_{\max}$ este în general mică pentru un alfabet de sursă relativ mare, caz în care avantajul codării aritmetice față de codarea Huffman este mic şi complexitatea suplimentară a calculelor nu justifică folosirea codării aritmetice față de codarea Huffman.

Dacă mărimea alfabetului este mică şi probabilitățile sunt puternic dezechilibrate, ca de exemplu în cazul facsimilului, folosirea codării aritmetice este justificată.

Un alt avantaj major al codării aritmetice este acela că este uşor de implementat un sistem cu coduri aritmetice multiple. Acest lucru pare contradictoriu, după ce anterior s-a precizat că, codarea aritmetică este mai complexă decât codarea Huffman pe mesaje individuale. Maşina de calcul determină creşterea complexității, dar odată ce se dispune de instalația de calcul pentru implementarea unei codări

aritmetice, pentru a implementa alte codări aritmetice nu trebuie decât să fie disponibile mai multe tabele de probabilități. Dacă alfabetul este mic (cazul surselor binare), complexitatea este mică. De asemenea, se poate adapta codarea aritmetică la schimbările statisticii de intrare. Tot ce trebuie făcut este de a estima probabilitățile alfabetului de intrare. Acest lucru poate fi făcut printr-o numărare a mesajelor, pe măsură ce acestea sunt codate. Mai mult, nu este nevoie a genera un arbore, ca în cazul codării Huffman adaptive, fapt care permite separarea procedurilor de modelare şi codare într-un mod care nu este realizabil cu codul Huffman. Această separare permite flexibilitatea mărită în proiectarea sistemelor de compresie.

4.5. Aplicații ale codării aritmetice

Codarea aritmetică este folosită într-o varietate de aplicații ale compresiei cu pierderi şi fără pierderi. În acest subcapitol se vor descrie două din cele mai cunoscute aplicații.

4.5.1. Compresia imaginii pe două nivele - Standardul JBIG

Codarea aritmetică este procedura de codare recomandată de Joint Bi-level Image Proccesing Group (JBIG) ca parte a standardului pentru codarea imaginilor binare.

În transmiterea progresivă a unei imagini, întâi se transmite o reprezentare a imaginii de rezoluție mică. Această reprezentare de rezoluție mică necesită foarte puțini biți pentru codare. Imaginea este apoi actualizată, sau îmbunătățită, la fidelitatea dorită, transmițând din ce în ce mai multă informație. Standardul JBIG combină transmisia progresivă şi compresia fără pierderi.

Compresia fără pierderi

Multe imagini pe două nivele au multe structuri locale, adică pixelii apropiați sunt puternic corelați. Se consideră o pagină de text în formă digitală. În multe locuri ale imaginii se vor întâlni pixeli albi cu o probabilitate apropiată de unu. În alte părți ale imaginii se vor întâlni cu o mare probabilitate pixeli negri. Se poate face o estimare rezonabilă a situației pentru un anume pixel, analizând valorile pixelilor din vecinătatea pixelului ce urmează a fi codat. De exemplu, dacă pixelii din vecinătatea pixelului de codat sunt în majoritate albi, atunci cu probabilitate mare, pixelul ce urmează a fi codat va fi, de asemenea, alb. Pe de altă parte, dacă cei mai mulți pixeli din vecinătate sunt negri, atunci este o mare probabilitate ca pixelul ce urmează a fi codat să fie tot negru.

Dacă se tratează fiecare caz separat, folosind un cod aritmetic diferit pentru fiecare din cele două situații, se vor obține îmbunătățiri față de cazul în care se foloseşte acelaşi cod aritmetic pentru toți pixelii.

Exemplul 4.13.

Se presupune că probabilitatea de a întâlni un pixel negru este 0,2, iar probabilitatea de a întâlni un pixel alb este 0,8. Entropia pentru această sursă este dată de:

$$H = -0,2\log 0,2 - 0,8\log 0,8 = 0,722$$

Dacă se foloseşte un singur cod aritmetic pentru a coda această sursă, se va obține o rată medie apropiată de 0,722 biți/pixel.

În continuare se presupune că, pe baza vecinătății pixelilor, se pot împărți pixelii în două grupuri, unul având 80% din pixeli, iar celălalt 20%. În primul grup probabilitatea de a întâlni un pixel alb este de 0,95, iar în al doilea grup probabilitatea de a întâlni un pixel negru

este 0,7. Entropiile acestor grupuri sunt de 0,286 şi, respectiv, 0,881. Dacă se folosesc două codoare arimetice diferite pentru cele două grupuri, se va obține o rată apropiată de 0,286 biți/pixel pentru 80% din cazuri şi apropiată de 0,881 biți/pixel pentru 20%. Rata medie va fi de aproximativ 0,405 biți/pixel, ceea ce este aproape jumătate din rata necesară dacă s-ar folosi un singur cod aritmetic.

Metoda codării aritmetice este atractivă pentru folosirea mai multor codoare. Toate codoarele folosesc aceeaşi unitate de calcul, dar fiecare codor este implementat pe o mulțime de mesaje cu distribuții de probabilitate diferite. Algoritmul JBIG foloseşte acest aspect al codării aritmetice. În loc să verifice dacă majoritatea pixelilor din vecinătate sunt albi sau negri, codorul JBIG foloseşte un model pentru pixelii din vecinătate (sau context) pentru a decide care set de probabilități să fie folosit în codarea unui anumit pixel. Dacă vecinătatea constă din 10 pixeli şi fiecare pixel poate lua două valori diferite, numărul posibil de modele este 2^{10}=1024. Codorul JBIG foloseşte între 1024 şi 4096 de codoare, în funcție de rezoluția, mică sau mare, a codării. Pentru codare de rezoluție mică codorul JBIG foloseşte una din cele două vecinătăți diferite prezentate în Fig. 4.3.

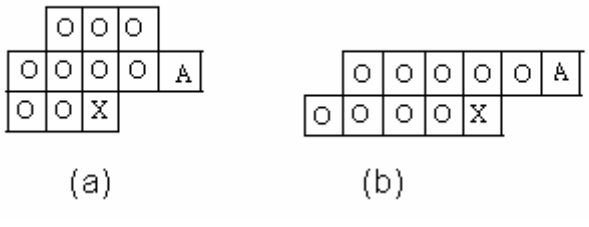

Fig. 4.3. a) vecinătate de 3 linii, b) vecinătate de 2 linii

Pixelul ce urmează a fi codat este marcat cu X, în timp ce pixelii folosiți pentru model sunt marcați cu O sau A. Pixelii A şi O sunt pixeli codați anterior şi sunt disponibili atât codorului cât şi decodorului.

Pixelul A poate fi privit ca un membru flotant al vecinătății. Plasarea sa depinde de intrarea care va fi codată. Se presupune că o imagine este formată din linii verticale aflate la distanță de 30 de pixeli. Pixelul A ar fi plasat cu 30 de pixeli la stânga pixelului ce este codat. Pixelul A poate fi mutat pentru a capta orice structură posibilă existentă în imagine. Acest lucru este util în special pentru imaginile cu contrast slab, în care pixelii A sunt folosiți pentru a capta o structură periodică. Localizarea şi deplasarea pixelului A sunt transmise decodorului ca informație colaterală.

În Fig. 4.4 mesajele din vecinătate au fost înlocuite cu 0 şi 1. Se consideră că simbolul 0 reprezintă un pixel alb, iar 1 reprezintă un pixel negru.

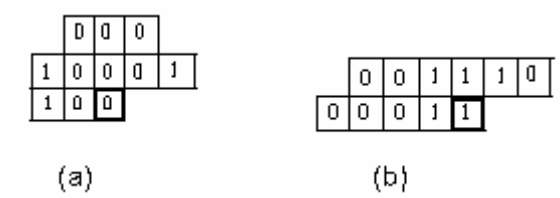

Fig. 4.4. Context de a) trei linii, b) de două linii

Pixelul ce urmează a fi codat este reprezentat printr-un contur mai îngroşat. Modelul ce conține mesajele 0 şi 1 este interpretat ca un număr binar ce este folosit ca un index pentru setul de probabilități. În cazul unei vecinătăți formate din 3 linii, contextul (citirea se face de la stânga la dreapta şi de sus în jos) este 0001000110, care corespunde indexului 70. Pentru o vecinătate de două linii contextul este 0011100001, adică indexul 225. Deoarece sunt 10 biți în acest model, vor exista 2^{10}=1024 codări aritmetice diferite.

În standardul JBIG cele 1024 de codoare aritmetice sunt o versiune a codorului aritmetic ce poartă numele de codor QM, care se bazează pe un codor aritmetic binar adaptiv numit, codor Q [73, 74, 75].

În descrierea codării aritmetice s-a actualizat intervalul etichetei prin actualizarea limitelor intervalului, $u^{(n)}$ şi $l^{(n)}$. Se poate urmări însă numai o limită a intervalului şi mărimea intervalului. Aceasta este metoda adoptată de codorul QM care urmăreşte limita inferioară a intervalului, $l^{(n)}$, şi mărimea intervalului $A^{(n)}$, unde:

$$A^{(n)} = u^{(n)} - l^{(n)} \tag{4.69}$$

Eticheta pentru o secvență este reprezentarea binară a lui $l^{(n)}$. Se poate obține o actualizare pentru $A^{(n)}$, scăzând relația (4.21) din (4.22).

$$A^{(n)} = A^{(n-1)}(F_X(x_n) - F_x(x_n - 1)) = A^{(n-1)} p(x_n) \tag{4.70}$$

Înlocuind $u^{(n)} - l^{(n)}$ cu $A^{(n)}$ în (4.21), se va obține ecuația de actualizare pentru $l^{(n)}$.

$$l^{(n)} = l^{(n-1)} + A^{(n-1)} F_x(x_n - 1) \tag{4.71}$$

În loc de a lucra direct cu biții de 0 sau 1 de la ieşirea sursei, codorul QM îi transformă în două categorii de simboluri: cel mai probabil simbol (MPS-More Probable Symbol) sau cel mai puțin probabil simbol (LSP-Less Probable Symbol). Dacă 0 reprezintă pixelii negri, iar 1 reprezintă pixelii albi, atunci într-o imagine în majoritate întunecată, 0 va fi cel mai probabil simbol, în timp ce într-o imagine majoritar luminoasă, 1 va fi cel mai probabil simbol. Notând probabilitatea de apariție a simbolului cel mai puțin probabil în contextul C prin q_c şi efectuând corespondența dintre simbolul cel mai probabil şi cel mai mic subinterval, apariția unui simbol MPS conduce la următoarele ecuații de actualizare:

$$l^{(n)} = l^{(n-1)} \tag{4.72}$$

$$A^{(n)} = A^{(n-1)}(1 - q_c) \tag{4.73}$$

în timp ce apariția unui simbol LPS conduce la următoarele ecuații de actualizare:

$$l^{(n)} = l^{(n-1)} + A^{(n-1)}(1 - q_c) \tag{4.74}$$

$$A^{(n)} = A^{(n-1)} q_c \tag{4.75}$$

Pentru a simplifica implementarea codorului QM, comitetul JBIG a recomandat unele modificări față de algoritmul codării aritmetice standard. Ecuațiile de actualizare implică multiplicări, care sunt costisitoare atât pentru hardware cât şi pentru software. În codorul QM, multiplicările sunt evitate prin presupunerea că $A^{(n-1)}$ are valoarea apropiată de 1. Astfel, ecuațiile actualizate devin:

Pentru MPS:

$$l^{(n)} = l^{(n-1)} \tag{4.76}$$

$$A^{(n)} = 1 - q_c \tag{4.77}$$

Pentru LPS:

$$l^{(n)} = l^{(n-1)} + (1 - q_c) \tag{4.78}$$

$$A^{(n)} = q_c \tag{4.79}$$

Pentru a nu încălca presupunerea făcută despre $A^{(n)}$ atunci când valoarea sa scade sub 0,75, codorul QM face o serie de rescalări până când valoarea lui $A^{(n)}$ este mai mare sau egală cu 0,75. Rescalarea ia forma unei dublări repetate, ceea ce corespunde unei deplasări spre stânga a reprezentării binare a lui $A^{(n)}$.

Pentru a păstra relația între parametri, aceeaşi scalare este aplicată şi lui $l^{(n)}$. Biții eliminați din buffer ce conțin valoarea lui $l^{(n)}$ reprezintă ieşirea codorului. Din ecuațiile de reactualizare ale codorului QM se poate vedea că se va produce o rescalare la fiecare apariție a unui LPS. Apariția unui MPS poate avea, sau nu, ca rezultat o rescalare, în funcție de valoarea lui $A^{(n)}$.

Probabilitatea q_c a lui LPS, pentru contextul C, este actualizată de fiecare dată când are loc o rescalare şi contextul C este activ. Lista

ordonată a valorilor lui q_c este dată într-un tabel. De fiecare dată când are loc o rescalare, valoarea lui q_c este schimbată la următoarea valoare din tabel, mai mică sau mai mare, după cum rescalarea a fost cauzată de apariția unui LPS sau MPS.

Într-o situație nestaționară, se poate întâmpla ca un mesaj atribuit lui LPS să apară mai des decât un mesaj atribuit lui MPS. Această condiție este întâlnită când $q_c > (A^{(n)} - q_c)$. În această situație atribuirile sunt inversate; simbolul atribuit etichetei LPS este atribuit etichetei MPS, şi invers. Testarea se face de fiecare dată când are loc o rescalare.

Transmisia progresivă

În unele aplicații, nu este nevoie mereu ca o imagine să se vadă la rezoluție maximă. De exemplu, dacă ne uităm la aşezarea în pagină a unei foi, nu este nevoie să ştim fiecare cuvânt sau literă din această pagină.

Standardul JBIG permite generarea progresivă a imaginilor de rezoluție mică. Dacă suntem interesați să vedem dacă există o anume figură într-o anumită pagină, putem cere o imagine de rezoluție mică, care poate fi transmisă utilizînd puțini biți. Odată ce imaginea de rezoluție mică este disponibilă, se poate decide dacă este necesară o imagine de rezoluție mare. Specificațiile JBIG recomandă generarea unui pixel numit de joasă rezoluție sau de nivel inferior pentru fiecare bloc de 2 x 2 pixeli din imaginea cu rezoluție mare. Numărul imaginilor de joasă rezoluție (numite straturi) nu este menționat de JBIG.

O metodă simplă de generare a imaginilor cu rezoluție mică este înlocuirea fiecărui bloc de 2 x 2 pixeli cu valoarea medie a celor patru pixeli, aceasta reducând rezoluția de două ori atât pe orizontală, cât şi pe

verticală. Metoda funcționează bine atât timp cât trei din cei patru pixeli sunt fie negri, fie albi. Când sunt doi pixeli de fiecare fel, înlocuirea lor cu pixeli fie albi, fie negri, duce la pierderea detaliilor, iar înlocuirea aleatoare cu un pixel alb sau negru are ca urmare introducerea unei cantități considerabile de zgomot în imagine.

În loc de a calcula media fiecărui bloc de 2 x 2 pixeli, JBIG furnizează o metodă bazată pe tabele pentru reducerea rezoluției. Tabelul este indexat de pixelii vecini arătați în Fig. 5.5, în care cercurile reprezintă pixelii stratului de rezoluție mică, iar pătratele pixelii stratului de rezoluție mare.

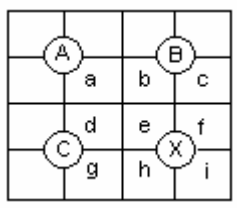

Fig. 4.5. Pixeli folosiți în determinarea valorii unui pixel de nivel inferior

Fiecare pixel contribuie într-o anumită măsură la index. Tabelul este alcătuit prin calculul expresiei [98]

$$X = 4e + 2(b + d + f + h) + (a + c + g + i) - 3(B + C) - A$$

unde a, b, c, d, e, f, g, h, i sunt valorile pixelilor din imaginea de rezoluție înaltă, iar A, B, C valorile pixelilor din stratul de rezoluție joasă. Dacă valoarea acestei expresii este mai mare decât 4,5, pixelul X este declarat a fi 1. Tabelul are câteva excepții de la regulă, pentru a reduce mărimea distorsiunii (petei) care apare la marginea imaginii, care trebuie luată în considerare, în general, într-o operație de filtrare. Există însă excepții care păstrează modelele periodice şi oscilante.

În standardul JBIG în codarea staturilor de diverse rezoluții, pixelii din imaginea de joasă rezoluție fac parte din contextul pentru codarea imaginilor de rezoluție mare. Contextele folosite în codarea straturilor de rezoluție joasă sunt cele arătate în Fig. 4.3. Contextele folosite în codarea straturilor de rezoluție mare sunt arătate în Fig. 4.6. În fiecare context sunt folosiți zece pixeli. Dacă se includ 2 biți necesari pentru a indica care model de context urmează a fi folosit, înseamnă că pentru a indica contextul vor fi folosiți 12 biți. Aceasta înseamnă că pot fi 2^{12}=4096 contexte diferite.

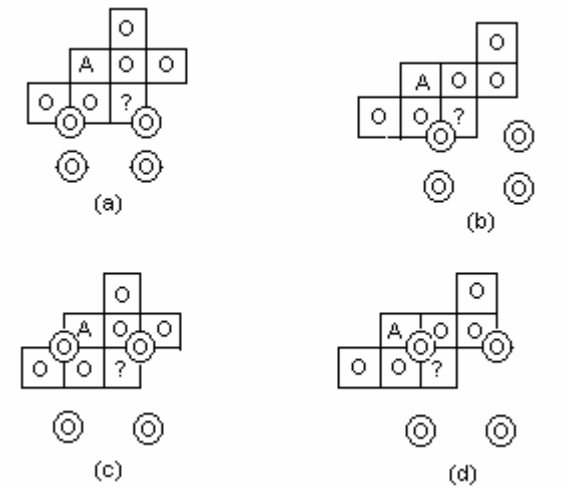

Fig. 4.6. Contexte folosite în codarea straturilor de rezoluție înaltă