Coduri detectoare si corectoare de erori. Coduri Hamming

În domeniul compresiei de date (considerata în teoria codarii ca si "**codarea sursei**") codarea s-a facut în scopul reducerii dimensiunii reprezentarii, presupunând un canal de comunicatie ideal, fara pierderi. În cele ce urmeaza vom aborda pe scurt o problema diferita, si anume codarea în scopul detectiei si eventual corectiei erorilor ce pot apare pe un canal de comunicatie real, cu erori (considerata în teoria codarii ca si "**codarea canalului**").

În acest al doilea caz este evident ca, pentru a obtine detectie si corectie de erori, trebuie sa încarcam suplimentar fluxul cu biti dedicati acestui lucru, având de a face cu o crestere a dimensiunii reprezentarii (trebuie platit un pret...). În timp ce la compresia datelor se elimina cât mai mult posibil redundanta, codurile corectoare adauga acel nivel de redundanta necesar pentru a transmite datele în mod eficient si cu fidelitate pe un canal cu zgomot (cu erori).

1. Distanta Hamming

Pe lânga distantele prezentate într-un material anterior, o distanta de un interes deosebit în problema noastra este distanta Hamming, numita astfel dupa Richard Hamming, cel care a introdus-o în 1950. **Distanta Hamming** între doi vectori de dimensiuni egale este data de **numarul de pozitii în care acestia difera**. Ea masoara astfel numarul de schimbari care trebuie facute într-un vector pentru a îl obtine pe celalalt, sau reformulat numarul de *erori* care transforma un vector în celalalt.

Exemple:

vector 1	**codare**	**126359**	**01101011**
vector 2	**notate**	**226389**	**01001110**
distanta Hamming	3	2	3

Desi definirea este generala, în cele ce urmeaza von considera doar cazul vectorilor cu elemente binare, fiind vorba de fluxuri de biti transmise pe canalul de comunicatie. În acest caz distanta este data de numarul de 1 din rezultatul obtinut prin XOR.

Pentru început vom considera doar cazul simplu al **erorilor singulare** - adica avem un singur bit eronat.

2. Un exemplu simplu de detectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de doua ori. Introducem astfel în mod evident o redundanta care însa ne va permite sa detectam erori singulare.

Daca la receptie obtinem coduri 00 respectiv 11 am receptionat corect 0 respectiv 1 iar daca obtinem 01 sau 10 am detectat o eroare singulara (fara a putea decide însa nimic în sensul corectiei ei).

Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	00.11.00.00.11.00.11.11.00.11
mesaj receptionat cu eroare:	00.11.00.00.**10**.00.11.11.00.11 => **detectie** de eroare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 2.

3. Un exemplu simplu de corectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de trei ori. Introducem astfel în mod evident o redundanta si mai mare care însa ne va permite sa detectam si sa corectam erori singulare.

Daca la receptie obtinem coduri 000 respectiv 111 am receptionat corect 0 respectiv 1. Daca obtinem 001 sau 010 sau 100 am detectat o eroare singulara si putem **corecta** spre 000 (deci am receptionat 0) iar daca obtinem 011 sau 101 sau 110 am detectat o eroare singulara si putem **corecta** spre 111 (deci am receptionat 1).

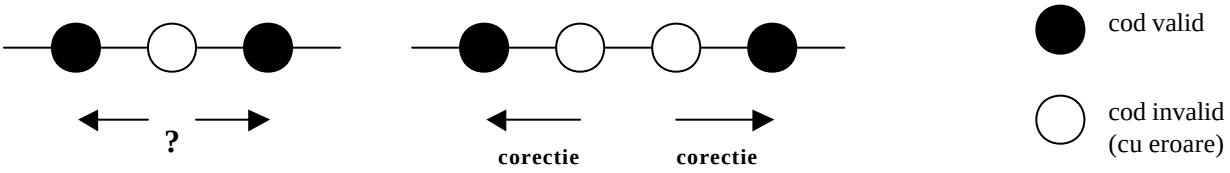
Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	000.111.000.000.111.000.111.111.000.111
mesaj receptionat cu eroare:	000.**101**.000.000.111.000.111.111.**001**.111
mesaj corectat:	000.**111**.000.000.111.000.111.111.**000**.111

=> **detectie** si corectie de erori singulare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 3.

Din exemplele anterioare constatam ca, daca distanta Hamming între oricare cuvinte de cod valide este 2, putem detecta erori singulare dar nu le putem corecta - nu stim care cod valid aflat la distanta 1 este cel corect. Daca distanta Hamming între oricare cuvinte de cod valide este 3 putem detecta erori singulare si putem corecta înspre codul valid aflat la distanta 1 (remarcam faptul ca, în acest caz, putem detecta chiar erori duble ! – fara însa a le putea corecta).



Daca scopul este doar de a detecta erori se pot folosi si alte solutii cum ar fi simpli biti de paritate, sume de control, coduri ciclice de tip CRC, functii hash criptografice, etc.

4. Codul Hamming (7,4)

Unul din cele mai cunoscute coduri detectoare si corectare de erori singulare este codul numit Hamming (7,4). Acesta notatie indica faptul ca avem un cod de 7 biti din care 4 sunt biti de date independenti (restul fiind biti redundanti, reprezentând paritatea a diferite combinatii a bitilor de date). Codul contine deci 4 biti de date d1, d2, d3, d4 si 3 biti de paritate p1, p2, p3. Bitii de paritate sunt calculati astfel:

p1 sumeaza d1, d2, d4
p2 sumeaza d1, d3, d4
p3 sumeaza d2, d3, d4

iar organizarea bitilor în cuvântul de cod este urmatoarea:

p1	p2	d1	p3	d2	d3	d4

De obicei se definesc doua matrici în legatura cu acest cod: matricea generatoare de cod G si matricea de detectie a paritatii H

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Se constata în G ca liniile 1,2 si 4 calculeaza sumele aferente paritatilor respective iar liniile 3,5,6,7 simplu copiaza bitii de date. În H cele 3 linii calculeaza paritatile corespunzatoare.

La codare: se determina cuvântul de cod calculând paritatile corespunzatoare (se poate utiliza atât paritatea para cât si paritatea impara - în exemplele urmatoare vom folosi paritatea impara).

La decodare: se calculeaza paritatile corespunzatoare si se verifica cu cele corecte (în fapt se sumeaza si cu paritatile corecte si se verifica sa rezulte 0).

Exemplu de codare-decodare pentru codul Hamming (7,4)

La codare: fie cuvântul de cod 1001. Calculam:

p1 = 1+0+1 = 0

p2 = 1+0+1 = 0

p3 = 0+0+1 = 1

rezultând codul Hamming **0 0 1 1 0 0 1** (am reprezentat subliniat bitii de paritate).

La decodare (cazul 1): presupunem ca am receptionat 0011001 (deci fara erori)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 0 + 1 + 0 + 1 = 0

p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0

Cum toate aceste valori sunt 0 rezulta ca nu am avut eroare.

La decodare (cazul 2): presupunem ca am receptionat 0011011 (deci cu o eroare în zona de date)

Reconstituim bitii de date: 1 0 1 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 0 + 1 + 1 + 1 = 1

p3 + d2 + d3 + d4 = 1 + 0 + 1 + 1 = 1

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 110 indica bitul 6 ca fiind eronat deci al treilea bit de date trebuie corectat (deci am realizat **corectie** de eroare). Deci, bitii de date corectati sunt 1 0 0 1.

Pozitionarea initiala aparent ciudata a bitilor de paritate în cadrul codului Hamming este necesara pentru a obtine simplu, direct indicele bitului eronat.

La decodare (cazul 3): presupunem ca am receptionat 0111001 (deci cu o eroare în zona de paritate)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 1 + 1 + 0 + 1 = 1

p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 010 indica bitul 2 ca fiind eronat deci al doilea bit de paritate era eronat iar bitii de date erau corecti (deci am realizat **corectie** de eroare).

Codul Hamming (7,4) descris anterior are distanta Hamming între oricare cuvintele de cod valide minim 3 si deci poate **detecta si corecta erori singulare** sau poate doar detecta erori duble (fara a putea corecta nici un fel de erori).

Calculele anterioare se pot face si în forma matriciala considerând la codare înmultirea matricii G cu vectorul coloana al bitilor de date iar la verificarea paritatii înmultirea matricii H cu vectorul coloana al codului Hamming.

5. Coduri Hamming generale

Codul Hamming descris anterior a fost generalizat pentru orice numar de biti. Algoritmul general pentru proiectarea unui cod Hamming **corector de erori singulare** (”single error correcting” - SEC) este urmatorul:

- Numerotam bitii începând cu 1: 1, 2, 3, 4, etc.
- Fiecare pozitie care este putere a lui 2 reprezinta **bit de paritate** (pozitiile care au un singur bit în reprezentarea binara)
- Celelalte pozitii reprezinta **biti de date** (pozitiile care au mai mult de un bit pe 1 în reprezentarea binara).
- Fiecare bit de paritate acopera biti astfel: bitul k de paritate acopera acele pozitii care au bitul k cel mai semnificativ pe 1.

Desi modul de calcul al paritatii (pare sau impare) nu este esential, de regula însa se foloseste paritatea impara (paritatea e 1 daca numarul de biti de 1 este impar).

Algoritmul general poate fi înteles mai bine din figura urmatoare:

pozitie		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
cod		p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12	d13	...
componenta biti paritate	p1	X		X		X		X		X		X		X		X		X		...
	p2		X	X			X	X			X	X			X	X			X	...
	p3				X	X	X	X					X	X	X	X				...
	p4								X	X	X	X	X	X	X	X				...
	p5																X	X	X	...

S-au reprezentat doar 5 biti de paritate si 13 biti de date având de a face cu codul care se noteaza Hamming(18,13).

Daca avem m biti de paritate putem acoperi pâna la $2^m - 1$ biti. Daca scadem cei m biti de paritate ramân $2^m - m - 1$ biti care pot fi folositi pentru date. În functie de valoare lui m avem urmatoarele coduri Hamming:

Biti de paritate	Total biti	Biti date	Nume cod	Eficienta utilizare biti date
2	3	1	Hamming(3,1)	1/3 = 0.333
3	7	4	Hamming(7,4)	4/7 = 0.571
4	15	11	Hamming(15,11)	11/15 = 0.733
5	31	26	Hamming(31,26)	26/31 = 0.839
m	2^m-1	2^m-m-1	Hamming(2^m-1, 2^m-m-1)	$1 - m / (2^m-1)$

Codurile Hamming descrise au distanta Hamming minima 3 deci permit:

- detectie si corectie a erorilor simple
- detectie a erorilor duble daca se renunta la corectie.

Uneori se adauga un **bit suplimentar de paritate** a întregului cod facând distanta minima 4. În acest caz se poate distinge între erorile simple (care se pot corecta) si erorile duble care doar se

detecteaza. Codurile astfel rezultate se numesc SECDED ("single error correction, double error detection".

De un interes deosebit este codul (72,64) care reprezinta o versiune trunchiata a codului cu 7 biti de paritate (m=7) adica Hamming (127,120) varianta cu bit aditional de paritate. Acesta este folosit pe circuitele de memorie de 72 biti pentru detectia erorilor duble si corectia erorilor singulare. Codul respectiv are o eficienta de utilizare a bitilor de date de 64/72 = 0.8888 identica cu situatia simpla a utilizarii unui bit de paritate pe fiecare byte, situatie corespunzatoare unui cod de tip (9,8). În plus fata de codul (9,8) care permite doar detectia erorilor singulare, codul (72,64) este de tip SECDEC adica permite detectia erorilor duble si corectia erorilor simple.