Codificació de canal I: introducció i codis de bloc

Francesc Tarrés
Margarita Cabrera

PID_00185014

Universitat Oberta de Catalunya

Índex

Introducció

Codificar és definir amb precisió el conjunt de símbols amb què s'enviaran els missatges d'emissor a receptor. Aquesta definició tan genèrica admet que el conjunt de símbols pugui elegir-se amb l'objectiu de solucionar problemes molt diferents. En efecte, el disseny de codis eficients es pot utilitzar per a resoldre tres problemes clàssics dels sistemes de comunicació: la compressió de les dades, la protecció davant eventuals errors de canal i l'encriptació de la informació perquè no pugui ser interpretada per un observador extern. En molts sistemes de comunicació, buscarem simultàniament la resolució de més d'un dels problemes esmentats, per la qual cosa és possible aplicar aquestes tècniques en cascada. Així, per exemple, la transmissió d'un vídeo pot suposar en primer lloc codificar la font mitjançant un compressor MPEG, encriptar les dades rebudes mitjançant un sistema de claus públiques i privades per limitar l'accés a la informació a qui hagi pagat per ella (accés condicional) i finalment codificar les dades per transmetre-les i protegir-les d'acord amb les característiques del canal.

En aquest mòdul ens ocuparem d'aquest problema, és a dir, de com podem establir una comunicació robusta de tal manera que el receptor sigui capaç d'adonar-se que les dades rebudes han estat alterades pel canal, s'han produït errors i fins i tot que en alguns casos sigui capaç de deduir quina era la informació original que s'havia transmès. Hi ha moltes causes que poden originar l'aparició de soroll i distorsions, i generar errors en el canal de comunicacions. Entre aquestes causes hem de destacar l'atenuació del canal, les interferències produïdes per altres sistemes de comunicació o fenòmens naturals, la propagació multicamí, el soroll, etc. En tots aquests casos es tracta de problemes originats en el canal de comunicacions, per la qual cosa la gravetat d'aquests i les tècniques utilitzades per a solucionar-los dependran de la naturalesa del sistema de comunicacions. És molt diferent un sistema de comunicacions per cable, en què la informació està fortament protegida d'interferències i sorolls, que un sistema de comunicacions per satèl·lit, en què s'han de cobrir distàncies molt grans sense utilitzar repetidors i en un canal sotmès a diferents fenòmens atmosfèrics.

Per aquest motiu, les tècniques i estratègies utilitzades per protegir la informació davant eventuals errors es coneixen amb el nom de *codificació de canal*, ja que el seu objectiu és condicionar la informació a les característiques del canal per tal d'assolir una comunicació fiable.

En dissenyar estratègies per a protegir la informació davant d'errors eventuals, no tenim gaire marge de maniobra. En efecte, l'única solució possible és que el receptor pugui distingir entre un missatge produït per l'emissor i un missat-

ge en què s'han produït errors. Per això, és necessari que introduïm elements addicionals en la informació que ens proporcionin pistes sobre la integritat dels missatges. Aquesta introducció d'informació addicional rep el nom de *redundància estructurada* i suposa que el nombre de bits d'informació que finalment es transmet pel canal sigui superior al mínim necessari. Així doncs, la inserció de codis de protecció davant errors comporta la introducció de bits addicionals en el transmissor. Aquests bits representen un augment de l'amplada de banda del senyal que transmetem. El preu que s'ha de pagar per protegir el missatge és, per tant, augmentar l'amplada de banda de la informació.

El mòdul comença amb uns conceptes elementals sobre mètodes per a introduir redundància estructurada en els missatges i una discussió genèrica sobre les estratègies per a detectar i corregir errors. S'estableixen i discuteixen els escenaris d'aplicació en què pot ser convenient detectar o corregir errors. També es presenten algunes tècniques elementals de retransmissió de dades. Posteriorment, es cobreixen els elements matemàtics bàsics per als codis de bloc que constitueixen el nucli d'aquest mòdul. L'objectiu principal és proporcionar una idea general sobre els mecanismes de codificació i descodificació. Es presentaran alguns elements i eines matemàtiques bàsiques que no pretenen ser completes sinó simplement il·lustrar part de la tecnologia implicada en el disseny dels codis de protecció d'errors. Els detalls matemàtics d'alguns dels codis utilitzats en la pràctica són excessivament complexos per als objectius d'aquest text, per la qual cosa molts dels resultats es presenten sense demostració ni justificació.

Els codis de bloc no són els únics codis utilitzats per a corregir errors. Hi ha altres variants com els codis convolucionals o els turbo codis, els principis de funcionament dels quals són una mica més complexos i, a més, es deixen per a cursos de comunicacions més avançats.

Objectius

Els objectius a què ha d'arribar l'estudiant en acabar aquest mòdul són els següents:

1. Comprendre la necessitat de protegir la informació davant els errors del canal.

2. Distingir entre sistemes i aplicacions que requereixen detectar o corregir els errors.

3. Conèixer les tècniques bàsiques de redundància estructurada i paritat.

4. Disposar de capacitat per fer càlculs bàsics de probabilitats de detecció i correcció d'errors.

5. Comprendre el compromís entre amplada de banda, potència transmesa, probabilitat d'error i taxa de codi.

6. Conèixer els principis i elements matemàtics bàsics dels codis de bloc.

7. Conèixer les característiques i el procés de generació dels codis de Hamming.

8. Conèixer les propietats i característiques dels codis BCH i de Reed-Solomon.

1. Redundància estructurada: conceptes bàsics

Introduir redundància en un codi és afegir un conjunt de bits que posteriorment ens permetin verificar que el grup de bits que rebem (al qual anomenarem *paraula* o *paraula codi*) concorda amb la transmesa. Es diu que la redundància s'introdueix de manera estructurada perquè els bits addicionals es determinen utilitzant procediments ben definits i coneguts tant pel receptor com pel transmissor.

1.1. Exemple 1. Codis de paritat simple

L'exemple més senzill d'introducció de redundància estructurada són els codis de paritat simple. A la figura 1 es mostra com es determina la paritat simple per a diferents paraules codi. En el nostre exemple, les paraules codi tenen originalment una longitud de 6 bits, per la qual cosa després d'afegir la redundància tindrem paraules amb una longitud de 7 bits. Per a afegir el bit de paritat s'utilitza el criteri que el nombre total d'uns de qualsevol paraula sigui parell. Així, si una paraula ja té originalment un nombre parell d'uns, el bit de paritat prendrà el valor zero. En canvi, si a la paraula original el nombre d'uns és imparell, el bit de paritat prendrà el valor u. D'aquesta manera, després d'introduir el bit de redundància podem garantir que totes les paraules codi tenen un nombre parell d'uns. El codi descrit es denomina de paritat parell. També es podria definir un codi de paritat imparell, en què el nombre d'uns de la paraula codificada sempre és imparell.

La redundància s'ha introduït seguint un procediment sistemàtic (el nombre total d'uns ha de ser parell) que confereix una propietat única a totes les paraules codi. Aquesta propietat és la que aprofita el receptor per a verificar que no s'ha produït cap error en la transmissió. En efecte, si la paraula rebuda té un nombre parell d'uns, el receptor la donarà per bona, mentre que, si el nombre d'uns és imparell, el receptor podrà assegurar que s'ha produït algun error (almenys un) durant la transmissió.

Aquest procediment resulta eficient sempre que el nombre d'errors que es produeixi sigui un nombre imparell. En efecte, si es produeixen dos errors, la paraula codi tornarà a tenir un nombre parell d'uns, per la qual cosa no es podrà detectar com a paraula incorrecta. El mateix ocorre quan es produeix qualsevol nombre parell d'errors. Tots aquests casos es corresponen amb situacions d'errors que no poden ser detectades pel nostre codi. En general, per a qualsevol codi, sempre hi haurà patrons d'errors que no podran ser detectats. Per tant, es diu que el codi de paritat simple permet detectar qualsevol nombre imparell d'errors en una paraula codi.

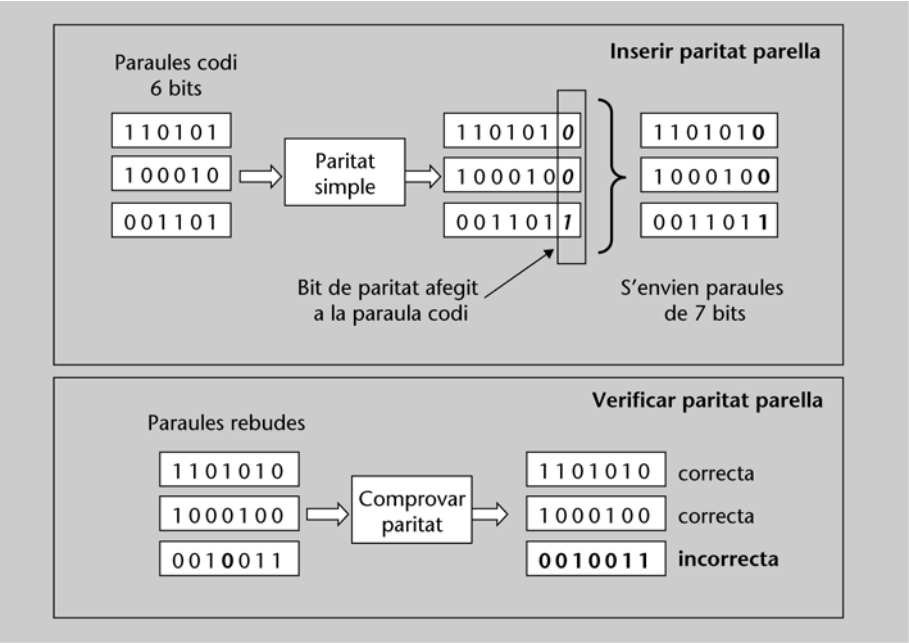

Figura 1. Codis de paritat parell: inserció i verificació

Problema 1

Suposem que en un determinat sistema de comunicacions la probabilitat, *p*, de cometre un error en un bit és $p = 10^{-4}$. Les paraules codi originals són de 7 bits i s'introdueix un bit de paritat parell addicional. Determineu la probabilitat que en una paraula codi s'hagin produït errors que no puguin ser detectats.

Solució

La probabilitat que s'hagin produït errors que no puguin ser detectats pel codi de paritat és:

$$p_{\text{error no detectat}} = p_{2\text{ errors}} + p_{4\text{ errors}} + p_{6\text{ errors}} + p_{8\text{ errors}}$$

És a dir, és la probabilitat que es produeixin dos errors més la que es produeixin 4, 6 o 8, ja que qualsevol nombre imparell d'errors serà detectat correctament.

Per a determinar la probabilitat que es produeixin dos errors en la paraula codi, hem de tenir en compte totes les possibles situacions en què tindrem 2 bits erronis en una paraula de 8 bits i ponderar cada una d'aquestes situacions per la probabilitat que es produeixi. Així

$$p_{2\,errores} = \binom{8}{2} \cdot p^2 (1-p)^6$$

On el nombre combinatori ens indica la totalitat de casos en què es produeixen dos errors en una paraula de 8 bits, essent p^2 la probabilitat que es produeixin dos errors en aquestes posicions i $(1-p)^6$ la probabilitat que no es produeixin errors en la resta de les posicions. Tenint en compte l'expressió anterior, la probabilitat que es produeixin errors en una paraula i que no siguin detectats pel codi serà:

$$p_{\text{error no detectado}} = \binom{8}{2} \cdot p^2 \cdot (1-p)^6 + \binom{8}{4} \cdot p^4 \cdot (1-p)^4 + \binom{8}{6} \cdot p^6 \cdot (1-p)^2 + \binom{8}{8} \cdot p^8$$

Recordeu que...

... el nombre combinatori $\binom{8}{2}$ representa totes les possibles combinacions per a prendre dos elements en una paraula de 8 bits. El càlcul es fa tenint en compte l'expressió següent:

$$\binom{n}{m} = \frac{n!}{m! \cdot (n-m)!}$$

Particularitzant per al valor $p = 10^{-4}$, obtenim:

$$p_{\text{error no detectado}} = \frac{8!}{2!6!}\cdot 10^{-8}\cdot(1-10^{-4})^6 + \frac{8!}{4!4!}\cdot 10^{-16}\cdot(1-10^{-4})^4 + \frac{8!}{6!2!}\cdot 10^{-24}\cdot(1-10^{-4})^2 + 10^{-32}$$
$$= 2{,}7983\cdot 10^{-7}$$

Observeu que únicament el primer terme de la suma és significatiu.

1.2. Exemple 2. Codis rectangulars

Els codis rectangulars, també coneguts com *codis de producte*, són una variant directa dels codis de paritat. Aquesta variant permet visualitzar, de manera molt senzilla, una estratègia per a la correcció d'errors. Per a aplicar un codi rectangular, els bits del missatge original s'han d'organitzar en una matriu. A la figura 2, es mostra un missatge original de 20 bits organitzat en una matriu de 5 columnes i 4 files. Els bits de redundància es calculen com un codi de paritat simple, aplicant-lo primer a les files i després a les columnes (o viceversa). A la figura es mostren tots els bits resultants en què la darrera columna i la darrera fila es corresponen als bits de paritat. Els bits es poden transmetre en l'ordre que es vulgui, sempre que el transmissor i el receptor es posin d'acord. Així, podrem transmetre totes les files, una després de l'altra. El receptor anirà situant els bits rebuts en una matriu de 6 columnes per 5 files, per la qual cosa els bits d'informació i de redundància estaran disposat en el mateix ordre que en el transmissor.

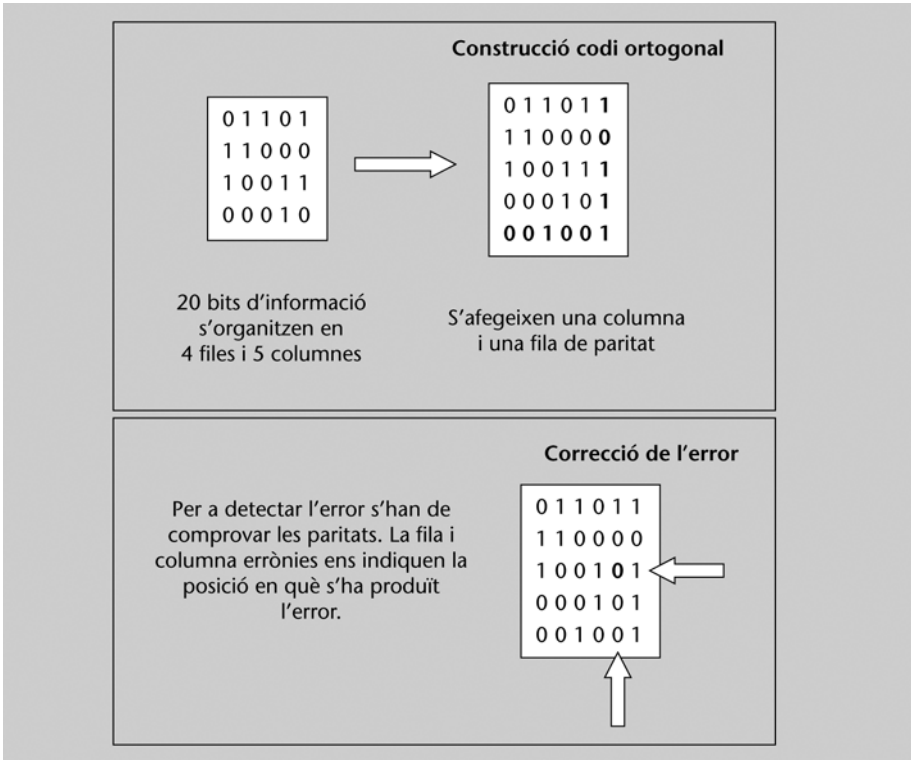

Figura 2. Construcció i correcció d'errors en un codi ortogonal

En aquesta mateixa figura es mostra com es pot fer la correcció d'un bit erroni. El receptor fa la comprovació de paritat de cada una de les files, i *detecta* com

a errònia la fila en què s'ha produït l'error. Posteriorment, en fer la comprovació de la paritat de cada una de les columnes, *detectarà* la columna en què s'ha produït el mateix error. D'aquesta manera, podem aprofitar aquesta informació per saber la fila i la columna en què s'ha produït l'error. És obvi que, si sabem la posició en què s'ha produït l'error, corregir-lo consistirà simplement a canviar-ne el valor.

El codi proposat solament pot garantir la **correcció d'un error en un bit**. És fàcil pensar en situacions en què es produeix més d'un error i en què seria possible determinar les posicions en què s'ha produït (sempre que la columna i la fila dels dos errors siguin diferents). No obstant això, si es produeixen en una mateixa fila o columna no serà possible detectar-los. En aquests casos, es diu que els errors superen les capacitats de correcció del codi.

1.3. Taxa del codi

Els codis de paritat i els codis ortogonals són molt simples, però ens permeten il·lustrar algunes de les definicions sobre els codis de protecció d'error que tenen validesa general. Prenent com a referència la figura 3 podem veure que, en general, a partir d'un paquet de *k* bits d'informació (missatge original) s'afegeixen *r* bits de redundància. Els bits de redundància també reben sovint el nom de *bits de paritat*. El nombre total de bits de cada paraula codi és la suma dels bits de redundància més els bits del missatge original n = k + r. Un codi amb aquestes característiques s'identifica com a codi (n, k). La *taxa de redundància* d'un codi es defineix com el quocient entre els bits de redundància i els bits totals (n – k/n) i proporciona una idea del percentatge de bits de redundància que conté un codi. Anàlogament, la *taxa d'un codi* es defineix com el quocient entre el nombre de bits del missatge i el nombre de bits totals d'una paraula codi (k/n). La taxa del codi ens proporciona una idea de la relació entre la informació útil i la informació total que contenen els missatges. Així, per exemple, un codi amb una taxa 2/5 ens indica que conté dos bits d'informació útil per cada 5 bits que rebem. A la figura 3 es representen de manera esquemàtica aquests conceptes.

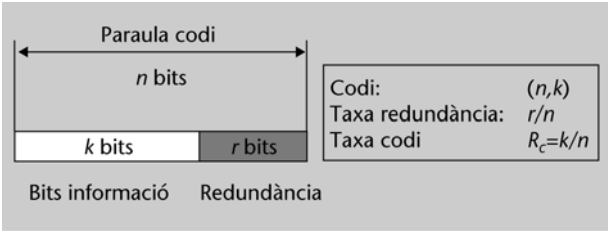

Figura 3. Definicions bàsiques de redundància i taxa de codi

La taxa del codi ens proporciona una idea del cost que té protegir la informació dels possibles errors de canal. En efecte, en afegir els bits de redundància hem de transmetre més bits que els estrictament necessaris. Això significa que, si volem fer la transmissió en temps real, haurem de transmetre un total de n = *k* + *r*

bits en el mateix temps que abans transmetíem *k* bits, cosa que suposa que necessitarem una amplada de banda més gran. L'augment de l'amplada de banda degut a la introducció de la redundància està directament relacionat amb la taxa de bits. En efecte, si abans havíem de transmetre *k* bits en un temps T i ara hem de transmetre'n *n*, la durada d'un bit passa de ser T/*k* a T/n, per la qual cosa l'amplada de banda augmenta aproximadament en un factor n/k. Així, doncs, l'augment de l'amplada de banda coincideix amb l'invers de la taxa del codi, que també es coneix com a factor d'expansió.

La *taxa d'un codi* és la relació entre el nombre de bits d'informació i el nombre de bits totals que conté:

$$R_c = k / n$$

L'invers de la taxa d'un codi proporciona una idea de com l'ús del codi augmenta l'amplada de banda del senyal i es denomina *factor d'expansió* d'un codi:

$$B = \frac{W_{coding}}{W_{no\ coding}} = \frac{1}{R_c} = \frac{n}{k}$$

Veurem com s'apliquen tots aquests conceptes en el problema següent per als codis de paritat i els codis ortogonals:

Problema 2

Determineu la notació del codi i les taxes de codi i redundància per als exemples 1 i 2 presentats en aquest apartat.

Solució

Per al codi de paritat simple, hem vist que cada sis bits d'informació útil (k =6) afegim un bit de paritat (r = 1). El nombre total de bits d'una paraula codi és de n = 7. Tenint en compte aquests valors, el codi es pot designar com (7,6) amb una taxa de redundància d'1/7 i una taxa de codi $R_c = 6/7$.

Per al codi ortogonal, hem vist que la informació útil s'organitzava en una matriu de 4 files per 5 columnes; per tant, cada paraula codi (o bloc d'informació) té un total de 20 bits útils (k = 20). En calcular les paritats per files i per columnes s'afegeixen 10 bits d'informació (r = 10; 4 files + 6 columnes). Amb aquests resultats obtenim que el codi s'ha de denotar com (30,20), la seva taxa de redundància és 10/30 = 1/3 i la taxa del codi és de $R_c = 20/30 = 2/3$.

2. Estratègies per al control d'errors

La introducció de redundància de forma estructurada en una paraula codi permet que el receptor pugui detectar o corregir, segons el cas, la presència d'errors. En aquest apartat veurem les possibles estratègies de control d'error que es poden utilitzar en funció de les característiques del canal de comunicacions o de l'aplicació, és a dir, intentarem justificar en quin tipus de situacions resulta més convenient utilitzar tècniques de detecció d'errors o tècniques de correcció.

Com a primera consideració, cal aclarir que *detectar* la presència d'un error consisteix a analitzar la paraula codi rebuda i determinar que no compleix les regles mitjançant les quals s'ha afegit la redundància. Per tant, sabem que la paraula és incorrecta, però no coneixem en quina posició s'ha produït l'error.

En canvi, la *correcció* d'errors implica no solament determinar que la paraula codi és incorrecta, sinó també conèixer les posicions en què s'han produït els errors per poder procedir a corregir-los.

A partir d'aquests raonaments, sembla lògic que la correcció d'errors serà sempre més complexa que la detecció, i entenem per *complexa* que requereix un major nombre de bits de redundància i que augmenta la complexitat computacional per fer la codificació i la descodificació. La necessitat d'un major nombre de bits de redundància també es pot interpretar com un augment de l'amplada de banda del senyal que es vol transmetre.

S'han de distingir, doncs, dues estratègies principals per a controlar els errors en el canal. La primera es basa a utilitzar codis que permetin que el receptor *detecti* els errors que s'han produït en el canal sol·licitant la retransmissió d'aquelles paraules que s'han rebut de manera incorrecta. Aquesta tècnica es coneix amb l'acrònim anglès ARQ (*automatic repeat request* o també *automatic retransmission query*), que indica que el receptor demana la retransmissió del missatge original quan detecta que s'ha produït un error.

Aquesta estratègia té les avantatges principals següents:

- Els codis de detecció d'errors són fàcils de calcular i detectar.
- La redundància afegida al missatge és menor que per a un codi de correcció.
- L'augment de l'amplada de banda degut a la introducció de la redundància també és menor.

Els principals inconvenients que té són els següents:

- En algunes aplicacions, com en la transmissió d'àudio i vídeo, no s'accepten endarreriments en la recepció dels paquets, ja que aquests endarreriments podrien interrompre el flux continu del reproductor.

- En moltes aplicacions, el canal de retorn no està disponible.

Les tècniques **ARQ** tenen diverses variants i s'utilitzen amb profusió en la descàrrega d'arxius per Internet(TCP/IP).

L'altra alternativa es coneix amb el nom de **FEC** (*forward error correction*) i utilitza codis de correcció d'errors que permeten que el receptor pugui restaurar per ell mateix la informació original en cas que s'hagin produït errors (sempre que els errors estiguin dins de les capacitats correctores del codi).

La necessitat del canal de retorn imposa una restricció molt important a les estratègies de control ARQ. En efecte, pensem per exemple en aplicacions com l'enregistrament de senyals de vídeo o àudio en suport òptic o magnètic. Una vegada que la informació ha estat enregistrada, si han aparegut errors en el senyal enregistrat (generalment degut a la degradació del suport), és impossible demanar que es torni a enregistrar la informació original. Per aquest motiu, en tots els sistemes d'enregistrament d'àudio i vídeo s'utilitzen tècniques FEC que permeten corregir els errors en el mateix descodificador. Els sistemes digitals de registre de senyals d'àudio (CD-audio *compact disc audio*, DAT *digital audio tape*, minidisc, DVD-Audio *digital versatile disc-audio*, etc.) o de registre de vídeo (DVD-Vídeo) utilitzen diferents variants de codis amb estratègies FEC que s'analitzaran més endavant en aquest mòdul. Un altre exemple típic són els sistemes de difusió (televisió digital, àudio digital) que tampoc permeten que el receptor torni a sol·licitar la transmissió de les paraules rebudes de manera incorrecta. Així doncs, la Televisió Digital Terrestre (DVB-T, *digital video broadcasting-terrestrial*) o la ràdio digital DAB (*digital audio broadcasting*) també utilitzen codis amb estratègies FEC. Un altre exemple típic és la difusió de continguts audiovisuals per Internet (*streaming*). En aquest cas, el motiu principal per a utilitzar estratègies FEC és que la mateixa naturalesa dels senyals audiovisuals requereix la reproducció continua sense que apareguin les possibles interrupcions que pot introduir la sol·licitud de retransmissions.

Elegir si en un determinat sistema de comunicacions és més convenient utilitzar estratègies de correcció ARQ o FEC és un problema complex que depèn de molts factors, entre els quals destaquem, a tall de resum, els següents.

Complexitat del codificador/descodificador. Alguns sistemes per a la correcció d'errors FEC poden tenir complexitats computacionals considerables, per la qual cosa solament poden ser utilitzats en equips terminals de cost elevat que puguin permetre's incorporar processadors avançats. En canvi, les estratè-

gies ARQ tenen una implementació més simple que les FEC, per la qual cosa poden ser utilitzades en equips més econòmics.

Augment de l'amplada de banda. En general, per a corregir errors és necessari introduir més redundància que per a permetre'n la detecció. Això suposa que l'augment d'amplada de banda degut al codi sol ser més gran quan s'utilitzen estratègies FEC.

Probabilitat d'error del canal. El nombre d'errors que es produeixen en el sistema de comunicació (abans de corregir-los) pot ser molt elevat, cosa que comporta que les tècniques ARQ resultin molt ineficients (vegeu com a exemple el problema 3).

Contingut del missatge. En general, quan es transmeten arxius de dades podem permetre'ns endarrerir-nos en els blocs per als quals sol·licitem la retransmissió. El receptor pot gestionar un *buffer* de memòria en què es recompon l'ordre original dels blocs. En aplicacions en què intervenen senyals d'àudio o vídeo, aquests endarreriments no són admissibles, per la qual cosa no acostumen a utilitzar-se tècniques d'ARQ.

Característiques del canal de comunicacions. Hem vist que en els sistemes en què no hi ha canal de retorn és necessari utilitzar estratègies FEC. Els exemples més típics són els sistemes de difusió audiovisual (com televisió i ràdio), l'*streaming* d'Internet i l'enregistrament de dades en suports òptics o magnètics. Quan hi ha canal de retorn, es poden utilitzar diferents variants de l'estratègia ARQ.

Veurem dos exemples de com poden incidir les característiques del canal en les estratègies utilitzades en les tècniques ARQ. En el primer cas, considerem un canal *full-duplex* i, en el segon, un canal *half-duplex*. Recordem aquestes definicions en el requadre de la dreta.

Un canal és *half-duplex*...

... quan pot transmetre en els dos sentits però no de manera simultània: primer ha de fer-ho en un sentit i després en l'altre. En un canal *full-duplex es* pot transmetre en ambdós sentits de manera simultània.

2.1. Exemple 3. ARQ en canals *full-duplex*

A la figura 4 es mostra de manera esquemàtica una estratègia ARQ denominada *de repetició selectiva* (*selective repeat* ARQ) i que requereix un canal *full-duplex* (la comunicació es pot fer en els dos sentits de manera simultània).

En aquest exemple, el transmissor va enviant contínuament els blocs del missatge en l'ordre preestablert i rep el reconeixement (ACK –*acknowledge*) del receptor. Quan l'ACK és negatiu (NAK – *non-acknowledge*) el transmissor torna a repetir el bloc que s'ha rebut de manera errònia. Si no apareixen errors en el canal, l'endarreriment entre el transmissor i el receptor depèn del temps de propagació del bloc. Si es produeixen errors (bloc número 4), el receptor haurà d'esperar a rebre aquest bloc de manera correcta si desitja realitzar la descodificació seqüencial del missatge. Observeu que aquesta estratègia ARQ requereix

que es faci una comunicació bilateral completa (*full-duplex*) entre el transmissor i el receptor. És interessant comparar aquesta variant d'ARQ amb la que s'introdueix en el problema següent.

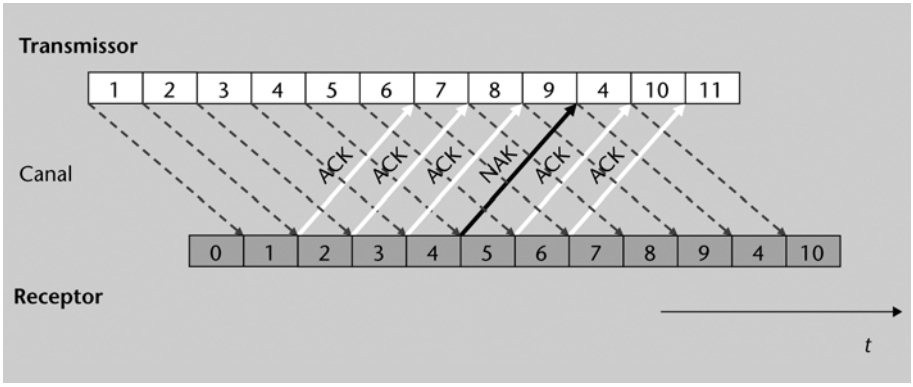

Figura 4. Exemple d'una variant d'estratègia ARQ (*selective repeat*). En aquest cas, es requereix disposar d'un canal de comunicacions *full-duplex*.

2.2. Exemple 4. ARQ en canals *half-duplex*

En aquest exemple s'il·lustra el mecanisme de control d'errors d'una variant d'ARQ denominada Parada i Espera (*Stop&Wait*). A la figura 5 es mostra l'esquema general dels paquets enviats, la seva propagació pel canal i la resposta de reconeixement del receptor. Aquest mecanisme consisteix en el fet que el transmissor envia un paquet de dades i espera a rebre la confirmació que s'han rebut correctament. Durant el temps d'espera no es transmet més informació, per la qual cosa el canal pot ser *half-duplex*, de manera que el transmissor, una vegada enviat el paquet, passa (commuta) a mode de recepció per esperar l'ACK/NAK del receptor.

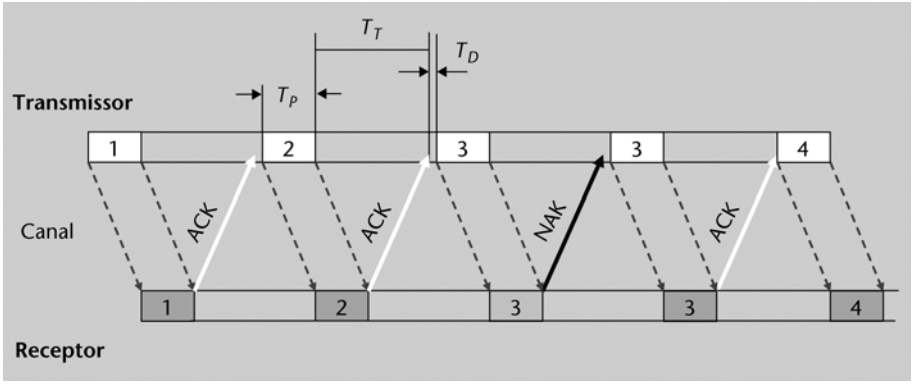

Figura 5. Esquema simplificat de la variant *Stop&Wait* per a l'estratègia de control d'errors ARQ

En aquest exemple volem veure els efectes que tenen en aquest esquema de control d'errors alguns dels paràmetres bàsics com la distància entre l'emissor i el receptor i la probabilitat d'error. Suposem que l'emissor envia paquets de 210 bits (amb els codis de detecció d'error ja inclosos) a una velocitat de 100.000 bits/s, és a dir, la durada de cada bit és de 10^{-5} s. La distància entre el transmissor i el receptor és de 300 km i podem suposar, en primera aproximació, que el missatge es propaga a la velocitat de la llum, que prendrem com c = 3 10^8 m/s.

El paquet d'ACK o NACK té una durada de 10^{-4} s, i l'emissor respon amb la repetició del paquet anterior o amb un nou paquet sense endarreriment.

a) Determinarem la taxa efectiva de transmissió de bits entre l'emissor i el receptor, suposant que no es produeixen errors en els paquets.

Per a determinar la taxa efectiva de transmissió de bits entre l'emissor i el receptor, hem de calcular el temps total que s'inverteix per a enviar els 210 bits de cada paquet. Aquest temps ha d'incloure:

T_P = Temps per a transmetre els 210 bits a 100 kbits/s

$T_P = 210 \text{ bits} \times 10^{-5} = 2{,}10\ 10^{-3}$ s

T_T = Temps d'anada i tornada d'un missatge

$T_T = 2\text{x}300\text{x}10^3\text{(m)}/3\text{x}10^8\text{ (m/s)} = 2\ 10^{-3}$ s

T_D = Durada del paquet ACK + endarreriments de resposta

$T_D = 10^{-4}$ s

Així, el temps total necessari per a la transmissió dels 210 bits és $T = 4{,}2\ 10^{-3}$ s, per la qual cosa la taxa efectiva de transmissió de bits és:

$$R = 210 \text{ bits} / 4{,}2\ 10^{-3} \text{ s} = 50.000 \text{ bits/s}$$

Per al nostre exemple, la taxa de bits s'ha reduït a la meitat degut a l'estratègia de control d'errors.

b) Determinarem ara la taxa efectiva de transmissió suposant que la probabilitat que es produeixi algun error en un paquet sigui de 10^{-2}.

En aquest cas, hem de tenir en compte que alguns paquets de 210 bits es rebran de manera incorrecta, per la qual cosa es tornaran a trametre i el temps mitjà per a la transmissió d'un paquet augmenta.

Podem calcular el temps mitjà per a la recepció d'un paquet mitjançant l'equació següent:

$$T_m = T \cdot (1 - p) + 2 \cdot T \cdot p(1 - p) + 3 \cdot T \cdot p^2 \cdot (1 - p) + \ldots.$$

On T representa el temps total calculat a l'apartat anterior. Observeu que el primer terme d'aquesta equació és el temps total per la probabilitat que no es cometi error. El segon terme és el temps que trigaríem a transmetre un paquet i tornar-lo a transmetre per la probabilitat que la primera vegada es rebi incorrectament i la segona es rebi correctament. El tercer terme correspon a haver

rebut dos paquets amb error i el tercer correctament, i així successivament. Per a determinar el valor de la progressió anterior, reordenem els termes:

$$T_m = T \cdot (1-p) \cdot \left[1 + 2 \cdot p + 3 \cdot p^2 + \ldots\right] = T(1-p) \cdot \sum_{k=0}^{\infty} (k+1) p^k = T(1-p) \cdot S(p)$$

On hem definit S(p) com:

$$S(p) = \sum_{k=0}^{\infty} (k+1) p^k = \frac{d}{dp}\left(\sum_{k=0}^{\infty} p^{k+1}\right) = \frac{d}{dp}\left(\frac{p}{1-p}\right) = \frac{1}{(1-p)^2}$$

En aquesta última línia, utilitzem diverses propietats matemàtiques que val la pena de comentar. A la segona igualtat ens adonem que els termes $(k+1)p^k$ poden expressar-se com la derivada respecte de p de p^{k+1}. Posteriorment, utilitzem la fórmula d'una progressió geomètrica convergent per sumar tots els termes (el primer terme dividit per 1 menys la raó). Finalment, derivem respecte de *p* el resultat.

Substituint aquest resultat en l'expressió anterior, obtenim:

$$T_m = \frac{T}{1-p}$$

Observeu que, per a la probabilitat d'error que tenim (10^{-2}), el temps total no es modifica de manera gaire apreciable, cosa que comporta una taxa efectiva de 49.500 bits/s. No obstant això, la dependència de T_m amb l'invers de $(1 - p)$ ens diu que, si les taxes d'error per paquet són importants, el control d'errors mitjançant aquestes tècniques pot resultar molt ineficient.

Per acabar, cal comentar que el mètode *half-duplex* té una dependència directa amb la distància entre els dos extrems de la comunicació. En efecte, el temps mitjà per a rebre un paquet depèn de manera directa del temps de transmissió, que al seu torn depèn de la distància entre els terminals. Això no era així per a l'estratègia ARQ de l'exemple 3, en què la distància solament podia afectar l'endarreriment global però no la velocitat de transmissió.

3. Per a què serveixen els codis de protecció d'errors?

La resposta a aquesta pregunta pot semblar un tant immediata, encara que, si ens aturem a reflexionar-hi, veurem que no resulta tan evident. En efecte, hem vist que l'estratègia per a protegir-nos davant de possibles errors consisteix a introduir redundància addicional als missatges que pretenem transmetre. Això significa que per a protegir-nos hem de pagar un preu: transmetre més bits que els estrictament necessaris. La conseqüència immediata és que l'amplada de banda del canal necessari per a transmetre la informació ha d'augmentar. En general, veurem que, si volem protegir la informació original davant un gran nombre d'errors, haurem d'usar codis amb taxes baixes, que suposen un augment considerable de l'amplada de banda. La taxa d'un codi és un indicatiu del cost en amplada de banda que representa la introducció d'aquest codi.

Si continuem reflexionant sobre el resultat anterior, ens adonarem que en augmentar l'amplada de banda es redueix el temps disponible per a transmetre cada un dels bits. Per tant, l'energia dedicada a la transmissió d'un bit en el missatge codificat és menor que la dedicada a transmetre un bit en la informació sense codificar (suposem que la potència del transmissor es manté constant). Això significa que la probabilitat que es produeixi un error en el missatge codificat és més gran que en el missatge sense codificar. La qüestió clau és si aquest augment de la probabilitat d'error queda compensat per la capacitat de correcció d'errors del codi.

La resposta a la pregunta és que generalment sí surt a compte, que obtenim un guany net, que la capacitat de correcció dels codis de canal compensa la pèrdua d'energia en cadascun dels bits del missatge codificat. La clau de la resposta a aquesta pregunta es pot trobar a la figura 6, en què es compara la probabilitat d'error que s'obté amb un missatge sense codificar i amb un missatge codificat. Les probabilitats d'error es representen en funció de la relació E_b/N_0 que representa la relació entre l'energia dedicada a un bit E_b i la densitat espectral de soroll N_0. És evident que, a mesura que augmentem l'energia dels bits, la probabilitat d'error decreix, ja que el missatge es veu menys afectat pel soroll. La gràfica també indica que hi ha una regió en què el codi de protecció resulta rentable, ja que s'obtenen probabilitats d'error menors que amb el missatge sense codificar. En canvi, hi ha una zona en què no resulta rentable codificar el missatge, ja que la probabilitat d'error amb el codi és més gran que sense el codi. En aquesta regió es diu que els errors superen les capacitats de correcció del codi. Es produeixen tants errors que la correcció no surt a compte.

En resum, és important comprendre que hi ha un delicat compromís entre diversos factors implicats i relacionats: la redundància introduïda, l'augment de l'amplada de banda, la taxa del codi i la potència amb que es transmeten els missatges. Tots aquests factors incideixen en la probabilitat d'error final del sistema de comunicacions, i la seva selecció adequada exigeix estudis rigorosos sobre les característiques dels canals i els codis que s'han d'utilitzar. L'elecció d'un codi o un altre en un sistema de comunicacions real exigeix profunds estudis i simulacions de les condicions en què es faran les comunicacions, les característiques del canal i els criteris de qualitat que volem obtenir.

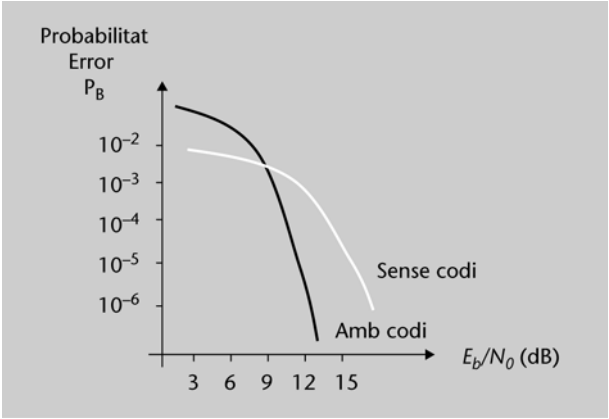

Figura 6. Exemple de la millora en probabilitat d'error en un sistema de comunicacions amb codis correctors

4. Decisions toves i decisions dures

Quan rebem un senyal per un canal amb soroll, el valor que obtenim es pot considerar una variable aleatòria. En el cas dels canals coneguts com *de soroll gaussià*, aquesta variable aleatòria té una funció densitat de probabilitat gaussiana. A la figura 7 es representa un exemple de la funció densitat de probabilitat que es rep en un canal gaussià quan es transmeten els símbols corresponents a un bit en un determinat sistema de modulació.

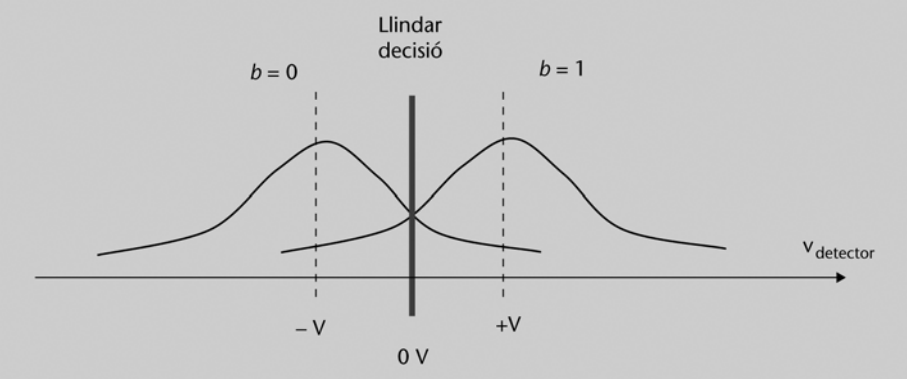

Figura 7. Resultat de la detecció d'un bit en un canal gaussià

En aquest exemple concret, estem suposant que el transmissor condiciona els bits al canal enviant tensions positives (de valor +V) quan es vol enviar un bit de valor 1 i tensions negatives (de valor –V) quan es vol enviar un bit de valor 0. Suposarem, per tant, que el detector hauria de detectar un nivell de tensió +V quan el bit transmès és un 1 i un nivell –V quan el nivell lògic és 0.

No obstant això, a la pràctica, quan es transmet un 1, la corba de la dreta ens indica la probabilitat que pot prendre la tensió rebuda en el receptor. De manera anàloga, l'altra corba ens indica la probabilitat dels valors de tensió quan es transmet un 0.

Diem que prenem un decisió "dura" quan per a cada bit rebut en decidim el valor, independentment dels valors que prenen la resta dels bits. La intersecció entre les dues corbes de densitat de probabilitat ens indica que, si el valor de tensió rebut és més gran que 0 (o, en general, més gran que el punt on es creuen les dues funcions densitat de probabilitat – f. d. p.), llavors la probabilitat que s'hagi transmès un 1 és més gran que la probabilitat que s'hagi rebut un 0. Anàlogament, si la tensió que rebem és menor que 0 hauríem de decidir que el bit transmet ha esta un zero lògic.

La decisió dura, contrasta amb les decisions toves, en el fet que no es decideix el valor de cada un dels bits rebuts fins que s'hagi rebut la paraula completa. Per

a comprendre millor la diferència entre les decisions toves i les decisions dures, considerarem l'exemple 5.

4.1. Exemple 5. Comparació entre decisions dures i toves

Suposem un codi de canal en què els únics missatges possible són el {000} i el {111}. Sabem que el valor mitjà de tensió que detecta el desmodulador quan transmetem un 1 és d'1 V i que quan transmetem un 0 el detector obté –1 V.

En un canal amb soroll gaussià rebem els 3 bits amb els valors de tensió {0,8; 0,8; –3}. Vegem quines serien les possibles decisions sobre el valor de la paraula transmesa:

- Decisió dura: en aquest cas decidim per a cada bit el valor transmès. Si el valor de tensió transmès és més gran que 0, decidirem que s'ha transmès un 1, i si és menor, un 0. Per tant, en el nostre exemple decidim que la paraula rebuda és {1 1 0}. Així doncs, si calculem la distància d'aquesta paraula amb les dues paraules codi veurem que la decisió seria que la paraula transmesa ha estat {1 1 1}

- Decisió tova: per a decidir, mantindrem el valor de tensió que realment s'ha detectat en el receptor i determinarem la distància euclídia entre la paraula rebuda i els dues paraules que hauríem rebut en absència de soroll.

$$D_{111} = \sqrt{(1-0,8)^2 + (1-0,8)^2 + (1-(-3))^2} = 4,009$$
$$D_{000} = \sqrt{(-1-0,8)^2 + (-1-0,8)^2 + (-1-(-3))^2} = 3,23$$

Cosa que ens indica que, si utilitzem una tècnica de decisió tova, decidirem que la paraula transmesa és {000}, el contrari que si s'utilitza una tècnica de decisió dura.

És molt important esmentar que no hi ha un criteri clar sobre la conveniència d'utilitzar una tècnica de decisió dura o tova. En general, per a sistemes amb soroll gaussià s'acostuma a considerar que la decisió tova produeix millors resultats, encara que sol ser computacionalment més complexa.

Noteu que en la majoria dels casos, les dues tècniques produeixen el mateix resultat final. Solament en algunes situacions el resultat que s'obté amb l'ús d'un mètode o d'un altre és diferent. L'exemple anterior volia posar èmfasi en els casos en què les dues tècniques produeixen resultats diferents per a aclarir les seves diferències.

5. Codis de bloc

Un **codi de bloc** consisteix a assignar una paraula o vector de n bits a cada un dels 2^k possibles missatges de k bits. En aquest text, únicament considerarem els codis de bloc binaris, en què cada paraula codi està formada per un conjunt finit de bits.

En aquest cas, el codi es pot interpretar com una taula en què per a cada possible missatge d'entrada tenim emmagatzemada la seva paraula codi corresponent. De fet, algunes implementacions del codificador de canal es poden realitzar mitjançant accessos a memòries LUT (*look-up tables* – memòries d'assignació). En la figura 8 es mostra un exemple d'un codi de bloc (6,3).

Missatges $K = 3$ bits		Paraules codi $n = 6$ bits	
x_0	000	000000	c_0
x_1	001	110110	c_1
x_2	010	010101	c_2
x_3	011	100011	c_3
x_4	100	101010	c_4
x_5	101	011100	c_5
x_6	110	111111	c_6
x_7	111	001001	c_7

Figura 8. Exemple d'una taula d'assignació per a un codi de bloc (6,3)

A continuació veurem un conjunt de definicions bàsiques que s'apliquen als codis de blocs i que utilitzarem posteriorment per a derivar-ne les característiques per a la correcció i detecció d'errors.

5.1. Definicions bàsiques

Distància de Hamming. La distància de Hamming entre dues paraules codi es defineix com el nombre de components en què les dues paraules són diferents. Prenent com a exemple el codi de la figura 8, tenim:

$$d(c_2,c_3) = 4;\ d(c_1,c_2) = 3$$

Distància mínima de Hamming. La distància mínima d'un codi és la menor distància de Hamming que hi pot haver entre dues paraules qualssevol diferents pertanyents al codi. Formalment:

$$d_{\min} = \min_{\substack{c_i,c_j \\ i \neq j}} d\,(c_i,c_j)$$

Pes d'una paraula codi. El pes d'una paraula codi és el nombre d'elements diferents de zero. Per al nostre exemple de codi de blocs:

$$\omega(c_3) = 3;\ \omega(c_0) = 0$$

Pes mínim d'un codi. És la paraula que té un pes mínim, exceptuant la paraula codi tot zeros. És, per tant, la paraula que té el mínim nombre d'uns.

5.2. Distància mínima, correcció i detecció d'errors

La distància mínima d'un codi ens indica quin és el nombre mínim de bits que hem de canviar en una paraula codi perquè es converteixi en una altra paraula codi.

Segons aquesta definició, sembla natural que la distància mínima determina les capacitats de correcció i detecció d'errors d'un codi. En efecte, suposem que prenem una decisió dura i que rebem una paraula en què és possible que s'hagin produït alguns errors respecte a la paraula x transmesa originalment.

Si solament desitgem detectar errors, conclourem que la paraula y conté errors sempre que no coincideixi amb una paraula codi. En canvi, si el que volem és corregir errors, el procés de descodificació més lògic és suposar que la paraula corregida és aquella que presenta una distància de Hamming més pròxima a la paraula rebuda.

En la figura 9 es mostra un exemple gràfic simplificat d'un codi que té una distància mínima de Hamming igual a 4. Les paraules codi es representen amb cercles, mentre que les combinacions binàries que no són una paraula codi es representen per punts. Les rectes uneixen paraules binàries que difereixen en un únic bit i que per tant estan a una distància de Hamming unitat. Observem que, per anar d'una paraula codi a una altra, s'han de produir un mínim de 4 transicions, cosa que ens indica que la distància mínima del nostre codi és 4.

En aquesta representació esquemàtica del codi s'observa que, si es produeix un únic error durant la transmissió del missatge, serà possible fer la correc-

ció utilitzant la paraula codi que resulti més pròxima. Posem, per exemple, que s'ha transmès el codi **c1** i que s'ha produït un error en un bit, amb la qual cosa rebem la paraula **e1**. El sistema pot fer la correcció sistemàtica d'aquest error degut al fet que la paraula codi més pròxima a **e1** és la paraula correcta **c1**. No obstant això, si es produïssin dos errors arribaríem a la paraula **e2**, que està a la mateixa distància de **c1** i de **c4**, per la qual cosa no podríem decidir quina de les dues paraules ha estat transmesa. En aquest cas, sembla lògic que, quan es rebi una paraula que està a la mateixa distància de dues paraules codi, solament puguem afirmar que s'ha produït un error (detecció), però no puguem corregir-lo. En canvi, si es produeixen tres errors, podem obtenir la paraula **e3**. Si intentem corregir aquesta paraula, decidirem de manera incorrecta, ja que **c4** és el codi més pròxim. A la figura 9 es mostren amb cercles discontinus les regions en què es poden corregir els errors d'un únic bit.

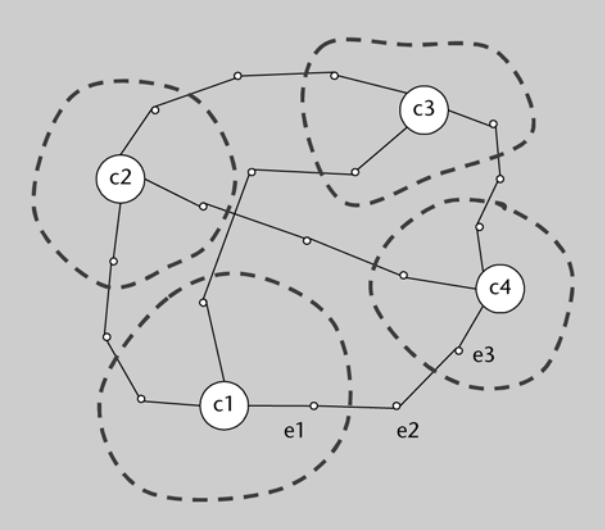

Figura 9. Exemple esquemàtic d'un codi amb distància mínima 4

Com a resum d'aquest exemple, veiem que podem aplicar dues estratègies bàsiques per a protegir-nos davant els errors. En efecte:

a) Podem utilitzar una estratègia només per a detectar els errors. En aquest cas, serem capaços de detectar correctament l'existència de fins a tres errors.

b) Podem utilitzar una estratègia per a corregir un error i detectar-ne dos.

L'elecció d'una d'aquestes estratègies o de l'altra depèn, en part, de les característiques del sistema. En sistemes en què hi ha un canal de retorn i en què els temps real no és prioritari, pot resultar recomanable aplicar estratègies de detecció d'errors, ja que és evident que sempre tindrem una capacitat de detecció més gran que de correcció. En sistemes en què prima el temps real, o en què no existeix un canal de retorn, sembla recomanable

utilitzar estratègies de correcció d'errors. A la figura 10 es mostra un gràfic simplificat de les dues estratègies que podem utilitzar en el nostre exemple.

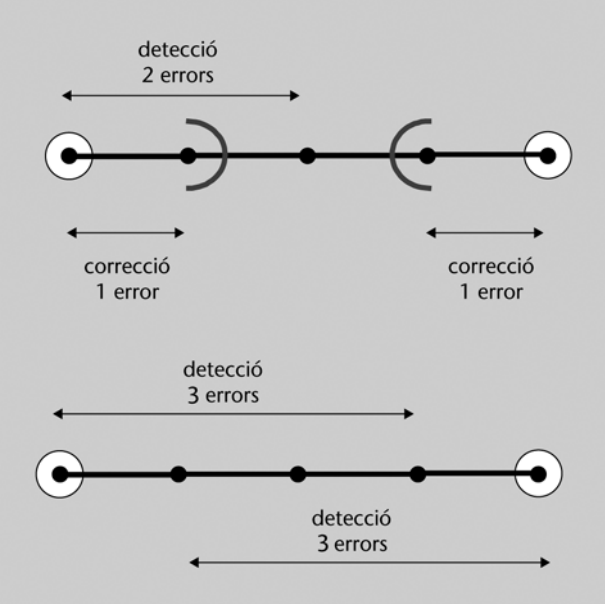

Figura 10. Exemple de les capacitats de correcció i detecció d'errors d'un codi amb distància mínima 4

Per a aclarir una mica més la figura 10, vegem amb detall com s'ha d'interpretar l'estratègia de correcció d'un error i detecció de dos, que es proposa a la gràfica superior.

- Quan el receptor rep una paraula, comprova quina és la distància mínima respecte de totes les possibles paraules codi que hagués pogut rebre.

- Si la distància mínima és zero, llavors pressuposem que no s'ha comès cap error, i no es corregeix.

- Si la distància mínima és un, llavor corregirem substituint el missatge rebut per la paraula codi amb què té aquesta distància mínima.

- Si la distància mínima obtinguda és dos, llavors no corregirem, però sabrem que hem comès un error.

Aquest tipus d'estratègies es coneixen amb el nom d'*estratègies de correcció/detecció mixtes* degut al fet que, en funció de la distància mínima obtinguda, decidim si es corregiran o es detectaran els errors.

A la pràctica hi ha algorismes més eficients que el que hem proposat i que estudiarem en l'apartat de descodificació. No obstant això, a efectes de comprendre les estratègies de correcció i detecció, l'algorisme anterior pot ajudar a entendre el procediment que se segueix. Hem d'esmentar que aquestes estra-

tègies solament funcionen si els errors que s'han produït en el canal estan dins de les capacitats correctores del codi. Així, si es produeixen tres errors, estimaren que la distància mínima de la paraula rebuda amb una paraula codi és un i que fem una correcció equivocada.

La diferència entre l'estratègia de correcció/detecció mixta i l'estratègia de solament detecció és que en aquesta última mai no corregim: solament detectem que s'ha produït un error.

És possible generalitzar els resultats anteriors per a codis amb qualsevol distància mínima. En cas que únicament fem detecció d'errors és evident que el nombre d'errors que podem detectar és igual a la distància mínima menys u.

$$e_d = d_{min} - 1$$

Això és així perquè qualsevol de les $d_{\min} - 1$ transicions que h ha entre dos paraules codi podran ser detectades com a errors.

Si solament es fa la correcció, hem de considerar per separat els casos en què la distància mínima és un nombre parell o imparell. Les relacions entre la distància mínima i el nombre d'errors que és possible corregir les dóna:

$$\begin{cases} 2e_c + 1 \le d_{\min} & d_{\min}\ imparell \\ 2e_c + 1 < d_{\min} & d_{\min}\ parell \end{cases}$$

Aquestes dues equacions es poden agrupar en:

$$e_c = INT\left[\frac{d_{\min} - 1}{2}\right]$$

On INT representa la part entera.

En cas que es faci una correcció/detecció mixta, es pot demostrar que s'ha de complir que la suma entre els errors que es corregeixen més els que es detecten sigui igual a la distància mínima menys u.

$$e_c + e_d = d_{\min} - 1$$

Observem que, en el cas de l'exemple amb distància mínima 4, podíem corregir un error i detectar-ne dos.

En l'exemple següent, intentarem aclarir totes les estratègies de correcció/detecció que podem dur a terme amb un codi amb distància mínima (de Hamming) igual a 5.

5.3. Exemple 5. Estratègies de correcció/detecció d'errors

Ara determinarem totes les possibles estratègies de correcció/detecció que podem aplicar utilitzant un codi amb distància mínima 5.

Amb un codi de distància mínima 5 tenim les possibilitats següents:

a) Corregir un màxim de dos errors.

b) Corregir un únic error i detectar-ne tres.

c) Detectar quatre errors.

Les tres possibilitats es mostren esquemàticament en la figura 11.

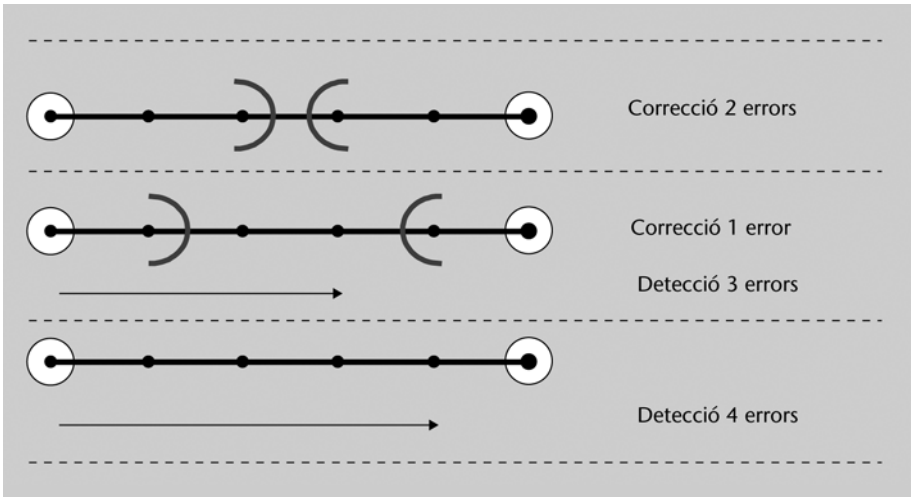

Figura 11. Possibilitats de correcció/detecció d'errors amb un codi de distància mínima 5

5.4. Codis de bloc lineals

Recordem de l'apartat anterior que restringim la nostra discussió de codis de bloc a codis binaris, és a dir, a codis en què totes les paraules estan formades per vectors d'1 i 0.

Es diu que un codi de bloc és **lineal** si la suma de dues paraules codi qualssevol és també una nova paraula codi. Podem expressar aquesta condició matemàticament com:

Per a tot *i,j*, existeix *k* tal que $\mathbf{c}_k = \mathbf{c}_i \oplus \mathbf{c}_j$; (1)

On el símbol $\oplus$ indica la suma en mòdul 2 component a component.

En l'exemple següent, intentem interpretar de manera pràctica el concepte de suma en mòdul 2 i el de codi de blocs lineal.

5.5. Exemple 6. Codi de blocs lineals

Exemple. Prenem com a exemple el codi de bloc que s'ha mostrat a la figura 8. En aquest cas, si prenem les paraules codi c_3 i c_4 obtenim:

$$\mathbf{c}_3 \oplus \mathbf{c}_4 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1] \oplus [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0] = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1] = \mathbf{c}_7$$

Es podria comprovar de manera exhaustiva que, per a dues paraules codi qualssevol, quan en fem la suma, obtenim una nova paraula que també forma part del codi. Per aquest motiu el codi anterior és un codi lineal.

Observem com es fa la suma de dues paraules codi: el resultat que obtenim per a cada component és independent de les altres, la suma de dos símbols diferents és sempre 1 ($1 \oplus 0$, $0 \oplus 1$) mentre que la suma de dos símbols iguals és sempre 0 ($0 \oplus 0$, $1 \oplus 1$). L'operació de suma és, doncs, idèntica a una porta OR exclusiva aplicada bit a bit.

Aquest exemple pot resultar enganyós, ja que hem comprovat que un codi que originalment havíem presentat com a codi de bloc genèric també és un codi lineal, cosa que portaria a entendre que pràcticament tots els codis de blocs són lineals.

Això és absolutament fals: si s'elegeixen a l'atzar les paraules codi en un codi de blocs, el més probable és que el codi resultant no sigui lineal. Considereu, per exemple, que en el codi de la figura 8 haguéssim pres la paraula c_7 = [0 1 1 0 0 1]. Es pot comprovar que, ara, la suma entre c_3 i c_4 ja no és dins del codi i que tindrem moltes altres sumes que tampoc no són paraules codi. El codi resultant en substituir aquesta paraula seria, per tant, un codi de bloc no lineal.

Una característica específica de tots els codis de bloc lineals és que contenen la paraula **0** (és a dir, el vector que té totes les components igual a zero). En efecte, atès que la suma de dues paraules qualssevol ha se ser també una paraula codi, podem veure que, quan fem la suma de qualsevol vector amb ell mateix ($c_j \oplus c_j$), sempre obtenim com a resultat la paraula **0**, de manera que aquesta paraula haurà de formar part del codi si volem que sigui lineal.

Un codi de **bloc binari** assigna una paraula codi (vector) de n bits a cada un dels $M = 2^k$ possibles missatges d'entrada de k bits ($k \leq n$; evidentment, la igualtat no introdueix redundància i no permet detectar/corregir errors).

Un codi de **bloc és lineal** si la suma de qualsevol de dues de les seves paraules codi és també una paraula codi.

Els codis de bloc solament han de verificar l'equació (1) per tenir tot el caràcter i propietats dels codis lineals. En canvi, en molts caos, s'estén el concepte de linealitat a l'aplicació que ens associa els missatges originals a les paraules codi. La majoria dels codis que s'utilitzen a la pràctica també tenen aquesta propietat de linealitat en la transformació, per la qual cosa és habitual que entenguem per codi lineal aquell que verifica l'equació (1) i a més:

$$Si\ \mathbf{x}_i \rightarrow \mathbf{c}_i\ y\ \mathbf{x}_j \rightarrow \mathbf{c}_j\ entonces\ \mathbf{x}_i \oplus \mathbf{x}_j \rightarrow \mathbf{c}_i \oplus \mathbf{c}_j \tag{2}$$

Bàsicament, aquesta equació estableix que, quan el missatge d'entrada es pot expressar com la suma d'altres dos missatges, llavors la paraula codi també es pot expressar com la suma de les dues paraules codi associades a cada missatge.

En la resta del text, per abús del llenguatge, ens referirem a codis de bloc lineals com als que compleixen l'equació (1), que ens diu que el codi és lineal, i l'equació (2), que ens diu que l'aplicació que associa els missatges originals a paraules codi és lineal.

5.6. Matriu generadora de codis de bloc lineals

En aquest apartat veurem una manera sistemàtica de generar codis lineals (en el sentit ampli definit en el paràgraf anterior). Per a això, hem d'identificar la paraula codi (*n* elements) que associarem a cada un dels elements de la base canònica dels blocs d'informació (*k* elements). Recordem que la base canònica està definida com aquella els elements de la qual tenen tots els components nuls, excepte un, que pren el valor unitat.

Probablement es veu més clar en l'equació següent, on els components $\{\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_N}\}$ formen la base canònica dels blocs de missatge.

$$\begin{aligned} \mathbf{e}_1 &= (100\ldots00) \rightarrow \mathbf{g}_1 \\ \mathbf{e}_2 &= (010\ldots00) \rightarrow \mathbf{g}_2 \\ \mathbf{e}_3 &= (001\ldots00) \rightarrow \mathbf{g}_3 \\ &\ldots \\ \mathbf{e}_k &= (000\ldots01) \rightarrow \mathbf{g}_k \end{aligned} \tag{3}$$

Els vectors $\{e_1, \ldots, e_k\}$ formen la base canònica de l'espai de paraules d'entrada al codificador. Qualsevol paraula d'entrada es pot expressar de manera directa com una combinació lineal dels elements de la base canònica. Els vectors $\{g_1, \ldots, g_k\}$ corresponen a les paraules codi que s'associen a cada un dels vectors de la base canònica. Són, per tant, vectors amb un total de *n* elements.

Qualsevol vector d'entrada es pot expressar com una combinació lineal dels vectors de la base canònica. En efecte:

$$\mathbf{x} = (x_1, x_2, \cdots, x_k) = \sum_{i=1}^{k} x_i \cdot \mathbf{e}_i \tag{4}$$

De manera que la paraula codi resultant es pot expressar com la mateix combinació lineal de les paraules g_i.

$$\mathbf{c} = \sum_{i=1}^{k} x_i \cdot \mathbf{g}_i \tag{5}$$

Resulta útil usar notació matricial per a representar les combinacions lineals implícites en les equacions anteriors. Per això, es defineix la matriu generadora del codi com a:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix} \tag{6}$$

Es tracta, per tant, d'una matriu amb k-files i n-columnes en què cada una de les files és la paraula codi associada a un dels vectors de la base canònica. Podem obtenir l'expressió (5) en forma matricial utilitzant la definició (6), amb la qual cosa s'obté:

$$\boldsymbol{c} = \boldsymbol{x} \cdot \boldsymbol{G} \tag{7}$$

5.7. Exemple 7. Matriu generadora d'un codi de blocs lineal

Considerem el codi (6,3), la taula d'assignació del qual s'ha definit a la figura 8. Per a obtenir la matriu generadora del codi, hem d'identificar els elements de la base canònica i les paraules codi que tenen assignades. D'aquesta manera:

$$\begin{aligned} \boldsymbol{e}_3 = \boldsymbol{x}_1 = (100) \rightarrow (101010) = \boldsymbol{g}_3 \\ \boldsymbol{e}_2 = \boldsymbol{x}_2 = (010) \rightarrow (010101) = \boldsymbol{g}_2 \\ \boldsymbol{e}_1 = \boldsymbol{x}_4 = (001) \rightarrow (110110) = \boldsymbol{g}_1 \end{aligned} \tag{8}$$

La matriu generadora del codi es construeix ordenant les paraules codi com a files:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_3 \\ \mathbf{g}_2 \\ \mathbf{g}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{9}$$

Finalment, per a calcular la paraula codi associada al missatge (1 1 1) podem utilitzar la relació (7). En efecte:

$$\mathbf{c} = \mathbf{x} \cdot \mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Que concorda amb la paraula codi corresponent de la figura 8.

5.8. Codis sistemàtics

Els codis sistemàtics que estudiarem en aquest apartat són codis en què la part del missatge i la part de la redundància estructurada es poden identificar de manera directa. En un codi sistemàtic, els *k* primers bits de la paraula codi corresponen al missatge, mentre que els següents $r = n - k$ corresponen a la redundància.

En general, els codis de bloc converteixen missatges de *k* bits d'informació en paraules codi de *n* bits. Es diu que la redundància afegida és de $(n - k)$ bits, tot i que no sempre resulta possible identificar els bits que es corresponen amb la redundància. En efecte, en l'exemple 7 hem vist que el missatge (1 1 1) es converteix en la paraula codi (0 0 1 0 0 1), o que el missatge (1 0 0) es converteix en (1 0 1 0 1 0). Per tant, es tracta d'un codi que introdueix 3 bits de redundància addicionals, tot i que no és possible identificar-los de manera directa en la paraula codi. Podríem dir que la redundància està dispersa en tots els bits de la paraula codi.

Un cas molt especial de codis de bloc són els codis sistemàtics en què els bits de redundància es poden localitzar de manera immediata. En un codi sistemàtic, els *k* primers bits es corresponen amb els del missatge original i els $(n - k)$ següents amb la redundància, en una forma anàloga a la que es representa a la figura 3.

Per a un codi sistemàtic, la matriu generadora sempre es pot descompondre en la forma següent:

$$\mathbf{G} = [\mathbf{I_k} \,|\, \mathbf{P}] \tag{11}$$

On I_k és una matriu identitat de *k* files i *k* columnes, la funció de la qual és repetir els *k* primers bits del missatge original. La matriu P indica com es poden calcular els bits de redundància a partir del missatge original. Intentem aclarir aquests conceptes mitjançant un exemple.

Exemple. Considerem la següent matriu generadora d'un codi sistemàtic (7,3).

$$\mathbf{G} = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

Podem veure amb detall el procés de codificació multiplicant la matriu per un missatge genèric $\mathbf{x} = (x_1, x_2, x_3)$. En efecte:

$$\mathbf{x}\cdot\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \cdot \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] = \begin{bmatrix} x_1 & x_2 & x_3 & x_1 + x_2 & x_2 + x_3 & x_1 + x_3 & x_1 + x_2 + x_3 \end{bmatrix}$$

Observem que els tres primers bits del missatge codificat coincideixen amb els del missatge original, degut al fet que la matriu identitat copia el valor original en aquestes posicions. La resta dels bits de redundància s'obté combinant els valors indicats per la matriu P. Cada columna d'aquesta matriu estableix les operacions que s'han de fer per a obtenir el bit de redundància corresponent. Aquest resultat suggereix també una implementació del codi lineal, mitjançant la realització de sumes entre els bits d'informació per a obtenir els bits de redundància. Entenem per *implementació del codi lineal* un diagrama de blocs amb elements lògics que poden fer-se en maquinari o en programari. A la figura 12 es representa de manera esquemàtica una possible implementació del codi.

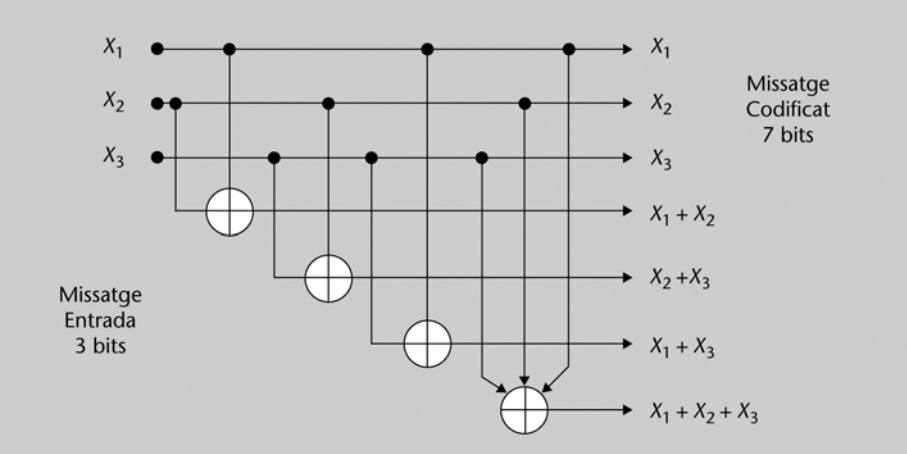

Figura 12. Esquema d'implementació d'un codi de bloc lineal sistemàtic

5.9. Matriu de revisió de paritat

La matriu generadora d'un codi de bloc lineal està formada per un total de *k* vectors fila, linealment independents i amb *n* components cada un d'ells. Per això, podem dir que aquests vectors fila generen un subespai vectorial de dimensió *k* (el nombre de vectors) dins de l'espai vectorial de n-components. Així, serà possible trobar un conjunt de (n – k) vectors que siguin ortogonals als vectors generadors i que permetin obtenir el complement ortogonal a aquest espai vectorial. Aquests (n – k) vectors seran ortogonals a totes les paraules codi, de manera que podem escriure:

$$\mathbf{c} \cdot \mathbf{H}^{\mathbf{t}} = \mathbf{0} \qquad (12)$$

On **c** representa qualsevol paraula codi i $\mathbf{H}^{\mathbf{t}}$ és la matriu les columnes de la qual estan formades pels **(n –** k) vectors que generen el complement ortogonal del nostre codi. El resultat del producte és un vector fila de (n – k) components amb tots els elements igual a zero (ortogonalitat entre c i les columnes de $\mathbf{H}^{\mathbf{t}}$). La matriu H es denomina *matriu de revisió de paritat*, ja que es pot utilitzar per a comprovar si una paraula és o no paraula codi. En efecte, si el producte de la paraula rebuda per la matriu H^t dóna com a resultat un vector amb tots els components nuls, deduirem que es tracta d'una paraula codi.

Atès que la relació (12) es compleix per a totes les paraules codi, podem substituir el vector **c** per la matriu generadora del codi. En efecte, H^t haurà de ser també ortogonal a tots els vectors base amb què es genera el codi. Així doncs:

$$\mathbf{G} \cdot \mathbf{H}^{\mathbf{t}} = \mathbf{0}$$

On, ara, la matriu **0** és una matriu amb *k* files i(n – k) columnes, amb tots els elements igual a zero.

Per a un codi sistemàtic, la matriu de revisió de paritat ve donada per:

$$\mathbf{H} = [\mathbf{P}^t \,|\, \mathbf{I}_{n-k}] \qquad (13)$$

La construcció de la matriu de revisió de paritat per a un codi sistemàtic és directa una vegada disposem de la matriu generadora. Podem comprovar que, si construïm la matriu d'acord amb aquesta equació, la matriu de revisió de paritat i la matriu generadora són ortogonals. En efecte:

$$\mathbf{G}\cdot\mathbf{H}^t = \left[\mathbf{I}_k \middle| \mathbf{P}\right]\cdot\begin{bmatrix}\mathbf{P}\\ \mathbf{I}_{n-k}\end{bmatrix} = \left[\mathbf{I}_k\cdot\mathbf{P} + \mathbf{P}\cdot\mathbf{I}_{n-k}\right] = \left[\mathbf{P}+\mathbf{P}\right] = \mathbf{0}$$

Observem que tant I_kP com $P\,I_{n-k}$ són matrius amb *k* files i(n – k) columnes.

Exemple. Considerem la matriu generadora d'un codi sistemàtic utilitzada a l'exemple anterior:

$$\mathbf{G} = \left[\mathbf{I}_k \vdots \mathbf{P}\right] = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1\\ 0 & 1 & 0 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 1\end{array}\right]$$

Podem obtenir la matriu de revisió de paritat utilitzen l'equació (13), amb la qual cosa tindrem:

$$\mathbf{H} = \left[\mathbf{P}^t \vdots \mathbf{I}_{n-k}\right] = \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 1 & 0\\ 1 & 1 & 1 & 0 & 0 & 0 & 1\end{array}\right]$$

És possible comprovar que qualsevol paraula codi és ortogonal a H^t. En particular, resulta interessant comprovar-ho per al cas general que hem obtingut en l'exemple anterior.

$$\mathbf{c}\cdot\mathbf{H}^t = \left[x_1, x_2, x_3, x_1+x_2, x_2+x_3, x_1+x_3, x_1+x_2+x_3\right]\cdot\begin{bmatrix}1&0&1&1\\1&1&0&1\\0&1&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} =$$

$$= \left[x_1+x_2+(x_1+x_2), x_2+x_3+(x_2+x_3), x_1+x_3+(x_1+x_3), x_1+x_2+x_3+(x_1+x_2+x_3)\right] =$$

$$= [0,0,0,0]$$

5.10. Codis de Hamming

Els codis de Hamming són codis de bloc lineals que tenen una distància mínima 3 i la matriu de revisió de paritat dels quals es construeix d'una manera di-

recta i simple. En tenir distància mínima 3, es poden utilitzar per a corregir un error o detectar dos errors. S'utilitzen en un gran nombre d'aplicacions, ja que poden fer-se amb un cost computacional baix. Si s'usen directament, poden corregir errors aïllats (fins a un bit a cada paraula), però també es poden usar conjuntament amb altres codis per a proporcionar solidesa al sistema.

Els valors de (n,k) per als quals existeixen codis de Hamming s'obtenen en funció d'un paràmetre addicional *m*, que ha de prendre valors més grans o iguals que 2 (ja que la distància mínima del codi ha de ser 3). Les relacions entre *n*, *k* i *m* les proporciona:

$$\begin{aligned} n &= 2^{m-1};\ m \geq 2 \\ k &= 2^m - m - 1 \\ d_{\min} &= 3 \end{aligned} \qquad (14)$$

Per a construir la matriu de revisió de paritat d'un codi de Hamming, hem de posar com a columnes totes les possibles combinacions de (n – k) components, tret de la columna tot zeros.

Considerem com a exemple la creació d'un codi de Hamming sistemàtic amb m = 3. Si substituïm m = 3 en l'equació (14), obtenim un codi amb paraules de n = 7 bits i missatges originals de k = 4 bits. La matriu de revisió de paritat serà de (n–k) = 3 files i n = 7 columnes. Si volem que el codi de Hamming sigui sistemàtic, les últimes tres columnes i files hauran de formar la identitat. Podem obtenir les quatre primeres columnes utilitzant totes les paraules de 3 bits que falten (suposem que les tres últimes columnes ja han estat omplertes) llevat de la paraula tot zeros. L'ordre en què es posin aquestes columnes origina diferents ordenacions dels bits de redundància, però en tots els casos es tracta del codi de Hamming sistemàtic de (7,4). Segons això, la matriu de revisió de paritat ve donada per:

$$\mathbf{H} = \left[\mathbf{P}^{\mathbf{t}} \vdots \mathbf{I}_{\mathbf{n-k}}\right] = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Una vegada construïda la matriu H, tenint en compte l'equació (13), podem identificar les quatre primeres columnes amb la matriu P^t. D'aquesta manera, utilitzant l'equació (11) obtenim la matriu generadora del codi de Hamming (7,4).

$$\mathbf{G} = \left[\mathbf{I}_{\mathbf{k}} \vdots \mathbf{P}\right] = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

6. Descodificació de codis de bloc lineals

Hem vist que la matriu generadora d'un codi ens proporciona un mètode sistemàtic per a obtenir la paraula codi de *n* bits associada a un determinat missatge de *k* bits. Aquest mètode es pot implementar en programari o directament en maquinari, utilitzant operadors booleans simples. D'altra banda, en recepció, podríem utilitzar la matriu de revisió de paritat per a comprovar que el missatge rebut és una paraula codi, ja que solament en aquest cas el producte del missatge per la matriu H donarà un resultat nul. Si el resultat del producte entre el missatge rebut i la matriu $\mathbf{H^t}$ –vegeu l'equació (12)– és diferent de zero, podem garantir que s'han produït errors en la transmissió del missatge. Per tant, usant aquest senzill procediment podem implementar sistemes de **detecció** d'errors per a codis de bloc lineals.

Tanmateix, si el nostre objectiu és **corregir** el missatge, el procediment de descodificació és una mica més complex. Podem intuir que la clau del problema rau a analitzar el resultat que obtenim del producte entre el missatge rebut i la matriu de revisió de paritat.

$$\mathbf{s} = \mathbf{c} \cdot \mathbf{H^t} \qquad (15)$$

Si el resultat del producte és nul, el missatge rebut es correspon amb una paraula codi. Sabem també que, si el resultat és diferent de zero, s'han produït errors, però no sabem com podem fer la correcció. El valor de **s** es coneix amb el nom de **síndrome** i ens proporciona la informació suficient per a corregir el missatge. No obstant això, abans haurem d'introduir alguns conceptes i propietats nous que resulten crucials per a corregir el missatge.

6.1. L'estàndard Array

L'estàndard Array és una estructura matricial per a codis de bloc lineals que resulta clau per a definir un procés sistemàtic de correcció dels missatges. Essencialment, és una matriu amb 2^k columnes i $2^{(n-k)}$ files en què apareixen ordenades totes les paraules binàries de *n* bits. Observeu que el nombre d'elements que conté la matriu és $2^k x 2^{(n-k)} = 2^n$, cosa que permet que qualsevol combinació de *n* bits pugui estar present en la matriu. L'organització dels missatges binaris en l'estàndard Array depèn del codi de bloc, tot i que el seu procediment de construcció és sistemàtic. Es pot resumir en els dos punts següents:

a) La primera fila de la matriu conté les 2^k paraules de n bits del codi de blocs lineal. L'ordre de les paraules és indiferent sempre que la primera paraula sigui

el missatge nul (tots els bits igual a zero). Denominarem aquest primer element $\mathbf{c_1 = 0}$. La resta dels elements es denoten com $\mathbf{c_j}$ amb $2 \le j \le 2^k$.

b) Per a construir la fila r (suposem r > 1), hem d'examinar totes les possibles paraules de n bits que encara no hagin estat introduïdes en les *r*–1 files anteriors. Elegim una paraula qualsevol de les que tinguin menor pes (nombre d'uns) i la situem com a primer element de la fila r. Denotarem com a $\mathbf{e_r}$ la paraula seleccionada. La resta dels elements d'aquesta fila s'obté com la suma $\mathbf{c_{rj} = e_r \oplus c_j}$ *con* $2 \le j \le 2^k$, on c_{rj} representa l'element de la fila *r*, columna *j* de l'estàndard Array, $\mathbf{e_r}$ és el primer element de la fila *r* i $\mathbf{c_j}$ és l'element *j* de la primera fila de la matriu.

A l'equació (16) es mostra esquemàticament l'organització de l'estàndard Array per a un codi lineal genèric.

$$\begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \cdots & \mathbf{c}_{2^k} \\ \mathbf{e}_2 & \mathbf{e}_2 \oplus \mathbf{c}_2 & \mathbf{e}_2 \oplus \mathbf{c}_3 & \cdots & \mathbf{e}_2 \oplus \mathbf{c}_{2^k} \\ \mathbf{e}_3 & \mathbf{e}_3 \oplus \mathbf{c}_2 & \mathbf{e}_3 \oplus \mathbf{c}_3 & \cdots & \mathbf{e}_3 \oplus \mathbf{c}_{2^k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{e}_{2^{(n-k)}} & \mathbf{e}_{2^{(n-k)}} \oplus \mathbf{c}_2 & \mathbf{e}_{2^{(n-k)}} \oplus \mathbf{c}_3 & \cdots & \mathbf{e}_{2^{(n-k)}} \oplus \mathbf{c}_{2^k} \end{bmatrix} \quad (16)$$

Cada una de les files de l'estàndard Array es denomina **COSET**, i al primer element de la fila l'anomenem **COSET Leader**.

La construcció de l'estàndard Array es pot il·lustrar amb un exemple que ens ajudarà a entendre algunes de les propietats i característiques que analitzarem posteriorment.

6.2. Exemple 8. Construcció de l'estàndard Array

Considerem un codi de blocs lineal (4,2) definit per la matriu generadora següent:

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

Per a calcular l'estàndard Array hem de posar a la primera fila de la matriu les paraules codi, i situar en primer lloc la paraula tot zeros. En aquest cas, les paraules codi són {0000, 1010, 0101, 1111}, amb la qual cosa tenim la primera fila de la matriu. Recordeu que, per a obtenir les paraules codi, n'hi ha prou a multiplicar cada un dels possibles missatges *x* de dos bits per la matriu generadora.

Per a determinar la segona fila, hem d'elegir una de les paraules de menor pes que encara no han estat posades a la matriu com COSET Leader. Elegim la paraula 1000 i obtenim la resta dels elements de la segona fila mitjançant la suma directa indicada a l'equació (16).

Per a obtenir la tercera fila, hem de buscar les paraules de menor pes que encara no han estat disposades. En aquest moment, hi ha pendents 0100 i 0001, per la qual cosa podríem escollir-ne qualsevol. Elegim la primera i completem la tercera fila. Finalment, per a completar la darrera fila hem d'elegir una paraula de pes 2, ja que totes les paraules amb pes unitat ja han estat col·locades. En aquest cas, la paraula elegida és 1100, amb la qual cosa completem la matriu fent les sumes directes amb els elements de la primera fila. La matriu final obtinguda és:

$$\begin{bmatrix} 0000 & 1010 & 0101 & 1111 \\ 1000 & 0010 & 1101 & 0111 \\ 0100 & 1110 & 0001 & 1011 \\ 1100 & 0110 & 1001 & 0011 \end{bmatrix}$$

Observem que amb aquest procediment no apareix cap paraula repetida, i que per tant tenim totes les possibles combinacions de 4 bits en l'estàndard Array.

6.3. Propietats de l'estàndard Array

Anem a comprovar que l'estàndard Array té un conjunt de propietats que faciliten la descodificació i correcció dels missatges rebuts.

Una de les propietats més importants és que tots els elements de l'estàndard Array són diferents (vegeu-ne la demostració en els exercicis resolts). Podem comprovar-ho en l'exercici concret de l'apartat anterior. Una conseqüència directa que tots els elements siguin diferents és que l'estàndard Array inclou totes les possibles paraules de *n* bits. En efecte, l'estàndard Array, per construcció, té un total de 2^k columnes i 2^{n-k} files, per la qual cosa el nombre total d'elements és 2^n, cosa que inclou totes les possibles combinacions de *n* bits.

Una altra propietat de l'estàndard Array és que tots els elements pertanyents a un mateix COSET tenen la mateixa síndrome. En efecte, suposem dos missatges $\mathbf{y_1}$ i $\mathbf{y_2}$ que pertanyen al mateix COSET, i en calculem la síndrome:

$$\begin{aligned} \mathbf{y}_1 = \mathbf{e}_l \oplus \mathbf{c}_i \quad &\Rightarrow \quad \mathbf{y}_1 \cdot \mathbf{H}^t = (\mathbf{e}_l \oplus \mathbf{c}_i) \cdot \mathbf{H}^t = \mathbf{e}_l \cdot \mathbf{H}^t = \mathbf{s}_l \\ \mathbf{y}_2 = \mathbf{e}_l \oplus \mathbf{c}_j \quad &\Rightarrow \quad \mathbf{y}_2 \cdot \mathbf{H}^t = (\mathbf{e}_l \oplus \mathbf{c}_j) \cdot \mathbf{H}^t = \mathbf{e}_l \cdot \mathbf{H}^t = \mathbf{s}_l \end{aligned}$$

L'estàndard Array inclou totes les possibles combinacions binàries de *n* bits (n-tuples). Les files de l'estàndard Array es denominen COSET i es caracteritzen pel fet que totes les paraules tenen la mateixa síndrome. La primera fila de la matriu, amb síndrome nul, està formada per les paraules del codi de bloc.

6.4. Correcció d'un missatge

Finalment, suposem que rebem una paraula y, en què s'han produït errors ja que no concorda amb cap de les paraules codi. Per a fer la correcció, hem de determinar la paraula codi, $\mathbf{c_j}$, la distància a la paraula rebuda de la qual sigui mínima. És a dir, el problema que pretenem resoldre és trobar el valor de *j* que fa que sigui mínima la distància següent:

$$\mathbf{c_j} \textit{ tal que } d\ (\mathbf{y}, \mathbf{c_j}) \textit{ sigui mínim}$$

Com que **y** és una n-tupla binària, podem trobar-la en l'estàndard Array, i per tant podem escriure-la com:

$$\mathbf{y} = \mathbf{e_l} \oplus \mathbf{c_i}$$

On c_i és qualsevol paraula codi i $\mathbf{e_l}$ es pot calcular com la síndrome de **y.** Per tant, hem de buscar el mínim de la funció de distància següent:

$$\mathbf{c_j} \textit{ tal que } d\left(\mathbf{e_l} \oplus \mathbf{c_i}, \mathbf{c_j}\right) \textit{ sigui mínim}$$

La distància entre dues paraules coincideix amb el pes de la suma, per la qual cosa podem escriure:

$$d(\mathbf{e_l} \oplus \mathbf{c_i}, \mathbf{c_j}) = w(\mathbf{e_l} \oplus \mathbf{c_i} \oplus \mathbf{c_j}) = w(\mathbf{e_l} \oplus \mathbf{c_m})$$

On hem utilitzat que, en tractar-se d'un codi lineal, la suma de dues paraules donarà lloc a una nova paraula codi. Ara, el terme de la dreta té pes mínim quan la c_m és la paraula tot zeros. En efecte, recordeu que totes les n-tuples corresponen al mateix COSET i que l'element que té pes mínim (pel mode de construir l'estàndard Array) és el COSET Leader.

Per tant, si c_m ha de ser zero, c_i haurà de coincidir amb $c_{j,}$ per la qual cosa podem escriure:

$$\mathbf{c_j} = \mathbf{c_i} = \mathbf{c_i} \oplus \mathbf{e_l} \oplus \mathbf{e_l} = \mathbf{y} \oplus \mathbf{e_l}$$

Per a entendre com es deriva aquesta equació, hem de tenir en compte que sempre que sumem el mateix vector dues vegades estem sumant zeros, per la qual cosa no modifiquem el valor original. L'equació anterior ens diu que el valor $\mathbf{c_j}$ que hem de codificar quan rebem qualsevol missatge *y* és el resultat de sumar el missatge amb el seu COSET Leader.

Simplificant-ho, podem descompondre el procediment en els passos següents:

- Determinar la síndrome del missatge rebut:

$$\mathbf{s_1} = \mathbf{y} \cdot \mathbf{H^t}$$

- Determinar el COSET Leader que té la mateixa síndrome:

$$\mathbf{s_1} = \mathbf{e_1} \cdot \mathbf{H^t}$$

- Corregir el missatge rebut:

$$\mathbf{c_j} = \mathbf{y} \oplus \mathbf{e_l}$$

Aquest resultat és certament interessant, ja que ens indica com es pot utilitzar la informació que proporciona la síndrome per a corregir la n-tupla rebuda. El procediment consisteix a calcular la síndrome del missatge rebut i a sumar al mateix missatge la n-tupla de pes mínim que tingui la mateixa síndrome. Es pot trobar certa analogia amb algunes tècniques mèdiques en què, una vegada coneguts els símptomes que presenta la malaltia (síndrome), s'aplica un element correctiu que produeix els mateixos símptomes amb l'objectiu de fer-ne la cura.

6.5. Exemple 9. Descodificació d'una paraula errònia

Vegem com es fa el procés de descodificar una paraula errònia, utilitzant el mateix codi de l'exemple previ. Suposem que hem rebut la paraula $y = 0111$, que no correspon a cap de les paraules del codi.

A partir de la matriu generadora i utilitzant l'equació (13), podem obtenir la matriu de revisió de paritat. En el nostre exemple, a causa de les simetries d'aquest codi concret, les dues matrius són idèntiques.

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

Si calculem la síndrome del missatge rebut, obtenim:

$$\mathbf{S} = [0111] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [10]$$

El valor de la síndrome coincideix amb la síndrome del COSET Leader 1000, per la qual cosa hauríem de descodificar la paraula com c = 0111 + 1000 = 1111.

Una manera alternativa i més simple de fer la descodificació és buscar el missatge en l'estàndard Array i donar com a resultat de la descodificació la primera paraula codi de la mateixa columna.

Hem de deixar clar que únicament hem utilitzat aquest exemple per a il·lustrar el procés sistemàtic de descodificació. En el nostre exemple no hauríem de fer la correcció de les paraules rebudes atès que estem utilitzant un codi amb dis-

tància mínima igual a 2, que pot detectar dos errors però no pot fer cap correcció. En efecte, observeu que el resultat que hem obtingut en aplicar el procediment de descodificació a 0111 és la paraula 1111, que està a una distància unitat de la paraula rebuda. No obstant això, 0101 també és una paraula codi que està a una distància unitat de la paraula que hem rebut.

7. Codis cíclics

Els codis cíclics són un cas particular dels codis de bloc lineals per als quals hi ha algoritmes molt eficients tant per a la codificació com per a la descodificació dels missatges. Els processos de codificació i descodificació dels codis de bloc vistos en els nostres exemples poden induir a pensar que la complexitat dels sistemes de correcció i detecció és moderada o baixa, ja que sempre hem tractat amb exemples manejables, amb un nombre reduït de paraules. No obstant això, especialment en el procés de descodificació, la complexitat computacional pot ser molt important, sobretot en aquells casos en què els valors de n i k i (n – k) són elevats, ja que es requereix fer processos de cerca de paraules codi o de síndromes que estaran emmagatzemades en taules de memòria de mida gran.

En aquest apartat, el nostre objectiu és el d'introduir propietats bàsiques dels codis cíclics, perquè us hi familiaritzeu i tingueu una perspectiva general de les seves característiques. No obstant això, no volem abordar un estudi detallat d'aquests codis, que exigiria una matemàtica rigorosa i relativament complexa. Lamentablement, aquest enfocament ens impedirà abordar la descripció dels algoritmes de descodificació que es poden utilitzar. Aquests algoritmes, tot i que són molt simples des del punt de vista computacional, requereixen un fort component matemàtic per a la seva comprensió. Des del punt de vista tecnològic, els codis cíclics són molt importants, a causa del fet que s'utilitzen en diferents aplicacions, especialment les variants dels codis de BCH i les dels codis de Reed-Solomon.

Un codi cíclic és un codi lineal de bloc que es caracteritza perquè, si c és una paraula codi, llavors qualsevol desplaçament circular de $\mathbf{c}^{(k)}$ també és una paraula codi. Entenem per desplaçament circular d'una paraula el fet que tots els bits es desplacen cap a la posició adjacent i l'últim bit es desplaça a la primera posició. A la figura 13 es mostra un exemple d'un desplaçament circular cap a la dreta.

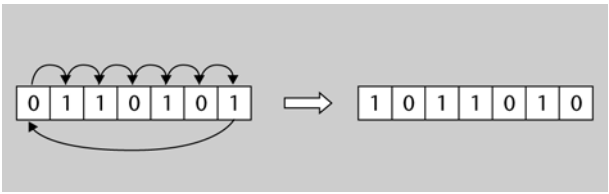

Figura 13. Exemple de desplaçament circular d'una paraula cap a la dreta

7.1. Exemple 10. Codis cíclics

El codi {000, 110, 101, 011} és un codi cíclic, ja que qualsevol desplaçament circular d'una paraula produeix una altra paraula codi. En efecte, el desplaça-

ment circular de la paraula 000 dóna lloc a la paraula 000, que és una paraula codi. El desplaçament circular 110 dóna lloc a 011, que també és una paraula codi. El mateix passa per a totes les paraules del codi.

En canvi, el codi {000, 010, 101, 111} no és un codi cíclic, ja que, per exemple, el desplaçament circular de 010 en 001 no és una paraula codi.

7.2. Representació de les paraules codi com a polinomis

Per a tractar matemàticament els codis cíclics, resulta convenient expressar les paraules codi mitjançant polinomis. En efecte, es pot establir una relació un a un entre l'espai vectorial n-tuples de *n* bits i l'espai vectorial dels polinomis de grau *n* – 1 d'acord amb l'expressió següent:

$$\mathbf{c} = (c_1, c_2, \ldots, c_n) \quad \Leftrightarrow \quad \mathbf{c}(p) = \sum_{i=1}^{n} c_i \cdot p^{n-i} = c_1 \cdot p^{n-1} + c_2 \cdot p^{n-2} + \ldots + c_{n-1} \cdot p + c_n$$

Aclarim aquesta definició amb un exemple. D'acord amb aquesta definició, per a una paraula codi de 7 bits (1, 0, 0, 1, 1, 0, 1), el polinomi associat serà d'ordre 6. El primer terme de la n-tupla binària es correspon amb el primer coeficient de la n-tupla, i així successivament.

En definitiva, el polinomi associat serà:

$$1 \cdot p^6 + 0 \cdot p^5 + 0 \cdot p^4 + 1 \cdot p^3 + 1 \cdot p^2 + 0 \cdot p^1 + 1 = p^6 + p^3 + p^2 + 1$$

Vegem com podem expressar matemàticament el desplaçament cíclic. Considerem:

$$c^{(1)}(p) = c_2 \cdot p^{n-1} + c_3 \cdot p^{n-2} + \ldots + c_n \cdot p + c_1$$

Com que estem operant amb aritmètica binària, podem afegir el factor $c_1 p^n$ dues vegades, que equival a zero:

$$c^{(1)}(p) = c_1 \cdot p^n + c_2 \cdot p^{n-1} + c_3 \cdot p^{n-2} + \ldots + c_n \cdot p + c_1 + c_1 \cdot p^n$$

Agrupant els *n* primers termes del segon membre de la igualtat, tenim:

$$c^{(1)}(p) = p \cdot c(p) + c_1 (1 + p^n)$$

Sumant $c_1(1+p^n)$ al darrer terme de la igualtat, obtenim:

$$p \cdot c(p) = c^{(1)}(p) + c_1(1 + p^n) \qquad (17)$$

Aquesta igualtat se sol interpretar dient que el polinomi corresponent al desplaçament circular d'una paraula codi es pot obtenir mitjançant el procediment següent:

- Multiplicar el polinomi de la paraula origen per p (la variable independent del polinomi)
- Dividir el polinomi resultant per $(1 + p^n)$
- El polinomi associat a la paraula desplaçada és la resta de la divisió.

D'acord amb aquest polinomi, l'equació (17) s'escriu generalment com:

$$c^{(1)}(p) = p \cdot c(p) \ (\mathbf{mod}\ (p^n + 1))$$

I es pot generalitzar per a qualsevol nombre de desplaçaments d'una paraula codi com:

$$c^{(r)}(p) = p^r \cdot c(p) \ (\mathbf{mod}\ (p^n + 1))$$

A partir d'aquesta expressió general, és evident que, quan hem realitzat n desplaçaments circulars, recuperem el polinomi inicial. En efecte:

$$c^{(n)}(p) = p^n \cdot c(p) \ (\mathbf{mod}\ (\mathrm{p^n} + 1)) = p^n \cdot c(p) + c(p) + c(p) \ (\mathbf{mod}\ (\mathrm{p^n} + 1)) =$$
$$= (\mathrm{p^n} + 1)c(p) + c(p) \ (\mathbf{mod}\ (\mathrm{p^n} + 1)) = c(p) \ (\mathbf{mod}\ (\mathrm{p^n} + 1)) = c(p)$$

7.3. Exemple 11. Desplaçament en codis cíclics

Exemple. Suposem una paraula binària de 5 bits 01001. El polinomi associat a aquesta paraula codi serà:

$$c(p) = 0 \cdot p^4 + 1 \cdot p^3 + 0 \cdot p^2 + 0 \cdot p^1 + 1 = p^3 + 1$$

Per a obtenir el polinomi associat quan fem un desplaçament circular d'un bit, hem de multiplicar per p i dividir per p^5+1. En aquest cas, l'ordre del polinomi resultant és menor que 5, per la qual cosa no és necessari fer la divisió. El polinomi resultant i la seva paraula codi associada és:

$$c^{(1)}(p) = p^4 + p \quad \Rightarrow \quad (10010)$$

Per a obtenir un nou desplaçament, multipliquem altre cop per p. Ara sí que hem de calcular el valor resultant de la divisió:

$$c^{(2)}(p) = p^5 + p^2 \quad (\mathrm{mod}(p^5 + 1)) = p^2 + 1 \quad \Rightarrow \quad (00101)$$

7.4. Polinomi generador d'un codi cíclic

En un codi cíclic, totes les paraules codi es poden expressar com el producte entre el polinomi associat a la n-tupla d'informació (ordre k – 1) i un polinomi d'ordre (n – k), denominat polinomi generador i que té la propietat de ser un divisor del polinomi $p^n + 1$.

Així, podem escriure el polinomi associat a la seqüència d'informació com:

$$X(p) = x_1 \cdot p^{k-1} + x_2 \cdot p^{k-2} + \ldots + x_{k-1} \cdot p + x_k$$

El polinomi generador es pot expressar com:

$$g(p) = p^{n-k} + g_2 \cdot p^{n-k-1} + g_3 \cdot p^{n-k-2} + \ldots + g_{n-k} \cdot p + 1$$

De manera que podem obtenir el polinomi associat a una paraula X, fent el producte entre els dos polinomis:

$$c(p) = X\,(p) \cdot g\,(p)$$

7.5. Exemple 12. Polinomi generador de codis cíclics

Anem a obtenir un codi cíclic de (7,4). Per a això necessitem un polinomi generador de grau 3. El polinomi generador ha se ser un divisor perfecte de p^7+1. Si descomponem aquest polinomi en factors, obtenim:

$$p^7 + 1 = (p+1)\,(p^3 + p^2 + 1)\,(p^3 + \mathrm{p} + 1)$$

Això ens indica que podem utilitzar qualsevol dels dos darrers polinomis de grau 3 com a polinomi generador d'un codi cíclic (7,4). Prendrem com a generador el darrer dels dos polinomis (però podríem haver pres el del mig):

$$g(p) = p^3 + p + 1$$

Ara, per a obtenir la paraula codi associada a un missatge genèric haurem de multiplicar el polinomi associat a aquest missatge per g(p). Calculem en primer lloc el polinomi associat al missatge d'informació X:

$$(x_1, x_2, x_3, x_4) \quad \Rightarrow \quad X(p) = x_1 \cdot p^3 + x_2 \cdot p^2 + x_3 \cdot p + x_4$$

Amb la qual cosa les diferents paraules codi s'obtenen fent el producte entre els dos polinomis.

$$c(p) = X\,(p) \cdot g\,(p)$$

Suposem que el missatge que volem transmetre és (0110). En aquest cas, el polinomi associat al missatge és X(p) = p^2+p. Quan multipliquem aquest polinomi pel polinomi generador, obtenim:

$$c(p) = (p^2 + p) \cdot p^3 + (p^2 + p)p + (p^2 + p) =$$
$$(p^5 + p^4) + (p^3 + p^2) + (p^2 + p) = p^5 + p^4 + p^3 + p$$

Els polinomis es multipliquen seguint les regles elementals del càlcul, però tenint en compte que els coeficients solament poden ser binaris (0 o 1). Per això, $p^k + p^k = 0$.

D'acord amb el resultat que hem obtingut, podem passar del polinomi a la paraula de 7 bits fent l'associació inversa entre (0111010).

De manera anàloga podríem completar la taula de tots els 16 missatges del codi cíclic.

7.6. Matriu generadora de codis cíclics

En aquest apartat veurem com podem obtenir la matriu generadora d'un codi cíclic sistemàtic a partir del polinomi generador. La matriu generadora es pot descompondre com:

$$\mathbf{G} = [\mathbf{I_k} \mid \mathbf{P}]$$

D'acord amb aquesta estructura, la i-èsima fila de la matriu la proporciona:

$$g_i = (0, 0, \ldots, 1, \ldots, 0, P_{i,1}, P_{i,2}, \ldots, P_{i,n-k})$$

Si expressem aquest vector com un polinomi i l'identifiquem amb una paraula codi X(p)g(p), obtenim:

$$g_i(p) = p^{n-i} + P_{i,1} \cdot p^{n-k-1} + \ldots + P_{i,n-k} = X(p) \cdot g(p)$$

Així doncs, podem escriure:

$$p^{n-i} = p_{i,1} \cdot p^{n-k-1} + \ldots + p_{i,n-k} + X(p) \cdot g(p)$$

O d'una manera més compacta:

$$p_{i,1} \cdot p^{n-k-1} + \ldots + p_{i,n-k} = p^{n-1}(\mathbf{mod}(g(p))$$

7.7. Exemple 13. Matriu generadora de codi cíclic

És important notar que el codi cíclic que obtindrem en aquest apartat és un codi sistemàtic, que encara que es construeix amb el mateix polinomi que hem uti-

litzat en l'apartat anterior, no dóna exactament el mateix codi. En efecte, el codi que hem obtingut en l'apartat anterior no és sistemàtic i aquest sí que ho serà. De fet, el codi que obtindrem té les mateixes paraules codi, però ordenades d'una manera diferent, és a dir, és el mateix conjunt de paraules codi però assignades a missatges diferents. Segueix sent, per tant, un codi cíclic, encara que aquesta vegada és sistemàtic.

Determinarem la matriu generadora associada al codi cíclic (7,4) el polinomi generador del qual és $g(p) = p^3 + p + 1$.

$$\begin{aligned}
&p^6 \mod(p^3+p+1) = p^2+1 && \Rightarrow \quad FILA\,1: \quad (1000101)\\
&p^5 \mod(p^3+p+1) = p^2+p+1 && \Rightarrow \quad FILA\,2: \quad (0100111)\\
&p^4 \mod(p^3+p+1) = p^2+p && \Rightarrow \quad FILA\,3: \quad (0010110)\\
&p^3 \mod(p^3+p+1) = p+1 && \Rightarrow \quad FILA\,4: \quad (0001011)
\end{aligned}$$

Per la qual cosa la matriu generadora serà:

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

7.8. Codificació dels codis cíclics

El principal avantatge dels codis cíclics és que tenen una estructura matemàtica més rígida que la dels codis de bloc convencionals, amb unes propietats ben definides que faciliten trobar mecanismes sistemàtics que faciliten la generació i descodificació.

En efecte, hem vist que les paraules codi es poden obtenir multiplicant el polinomi generador amb el polinomi de la informació que es va a codificar. El producte entre dos polinomis es pot considerar com una operació de convolució entre dues seqüències: la seqüència de les dades i la formada pels coeficients del polinomi generador. D'acord amb això, és possible interpretar la codificació com un filtratge de la seqüència d'informació mitjançant un registre de desplaçament (filtre) els coeficients del qual són els elements del polinomi generador. Així doncs, en resum, els codis cíclics es poden implementar mitjançant registres de desplaçament.

7.9. Codis BCH

Els codis BCH són una subclasse de codis cíclics que permeten corregir un nombre arbitrari d'errors *t*. Els codis van ser proposats per Bose, Chaudhuri y Hocquenghem, a qui deuen el nom. Aquests codis tenen una gran versalitat per al disseny, ja que hi ha un gran nombre de polinomis generadors prèviament tabulats. D'aquesta manera, l'usuari solament ha de determinar uns paràmetres de disseny, que depenen del nombre d'errors que vol corregir i de

la longitud dels blocs i, posteriorment, amb aquests paràmetres, ha de buscar el polinomi generador en una taula. Hi ha també algoritmes eficients per a la seva descodificació.

Els paràmetres de disseny es determinen a partir de les equacions:

$$n = 2^m - 1$$
$$\text{n} - \text{k} = \text{mt}$$
$$d_{min} = 2t + 1$$

D'acord amb aquests paràmetres, si volem utilitzar una mida de bloc de 31 bits (n = 31) i corregir un total de tres errors (t = 3), trobaríem un únic polinomi en les taules el valor de *k* del qual és 16. Així, el codi BCH amb una mida de bloc 31 que permet corregir tres errors conté un total de 16 bits d'informació i 15 de redundància. La taxa del codi és R = 16/31. El polinomi generador que obtenim en les taules està expressat generalment en octal. En el nostre cas concret és $G = 107657_{OCT}$.

Podem passar aquest polinomi a forma binària directament: G = 1 000 111 110 101 111, o expressar-lo directament com un polinomi convencional:

$$G(p) = p^{15} + p^{11} + p^{10} + p^9 + p^8 + p^7 + p^5 + p^3 + p^2 + p + 1$$

7.10. Codis de Reed-Solomon

Els codis de Reed-Solomon són una variant dels codis BCH i per tant també són una subclasse dels codis cíclics. Es tracta, en aquest cas, d'uns codis que trobem en moltes aplicacions, com per exemple les codificacions de canal en el CD-Àudio, el MiniDisc, el DAT, el DVD-Vídeo, els diferents sistemes de difusió de senyals audiovisuals DVB-T, DVB-S, DVB-C, etc. Desafortunadament, es tracta d'uns codis que requereixen un fort component matemàtic per a comprendre'n els detalls i propietats, per la qual cosa ens limitarem a enunciar algunes de les seves característiques sense demostrar-les.

La característica més específica d'aquests codis és que treballen a nivell de símbol i no a nivell de bit. És a dir, les paraules sobre els quals s'aplica el codi pertanyen a un alfabet amb un nombre finit de símbols. En la majoria de les aplicacions pràctiques, el símbol més utilitzat és el byte, és a dir, un símbol és una paraula de 8 bits. Els bits a l'entrada del codificador s'agrupen en paraules de 8 bits (bytes). El codi de Reed-Solomon pren un conjunt de K bytes a l'entrada i genera un total de N bytes a la sortida. Observem que la idea general és molt semblant al que hem vist fins ara, però que, en lloc de treballar a nivell de bit, es treballa a nivell de byte. La taxa del codi és R = K/N.

Els codis de Reed-Solomon poden corregir símbols complets. Això significa que, si un codi de Reed-Solomom està dissenyat per a treballar amb tres

errors per paquet, podrà corregir 3 bytes erronis complets, independentment dels bits erronis que hi hagi a cada byte. Així, per al codi de Reed-Solomon no suposa cap problema que tots els bits que formen el byte siguin erronis. La correcció és tan factible com si solament existís un bit erroni en el byte. Aquesta característica fa que els codis de Reed-Solomon siguin molt adequats en les aplicacions en què poden aparèixer errors en forma de ràfega, que afecten diversos bits consecutius. Aquesta situació és molt habitual en moltes aplicacions. En efecte, en el CD-Àudio els errors en la lectura dels bits del suport es produeixen a causa de l'acumulació de pols o ratllades en el disc que afecten diversos bits consecutius. El DAT és un sistema d'enregistrament d'àudio digital en cinta magnètica en què els errors apareixen a causa de la pèrdua de part del material magnètic en la cinta i que també produeix errors en més d'un bit. En els sistemes de radiodifusió poden aparèixer interferències o esvaïments que també afecten diversos bits consecutius. En moltes d'aquestes aplicacions, els codis de Reed-Salomon s'utilitzen en conjunció amb altres codis orientats a la correcció d'errors individuals.

Els paràmetres que defineixen un codi de Reed-Solomon (RS) són:

$$N = 2^k - 1$$
$$K = 1,3,\ldots,N-2$$
$$D_{\min} = N - K + 1$$
$$R_c = K / N$$

Longitud del bloc

Observem també que la longitud del bloc depèn directament del nombre de bits que formen un símbol. En el cas més habitual, k = 8 (els símbols són d'un byte) i el nombre total possibles símbols serà de 255. K és el nombre de símbols d'informació útil que s'utilitzen. A mida que augmentem K, augmenta la taxa del codi, però disminueix la capacitat de correcció (D_{min}). El nombre de símbols erronis que es pot corregir el proporciona:

$$t = \frac{D_{\min} - 1}{2}$$

Activitats

1. Demostreu la propietat següent de l'estàndard Array.

Propietat 1. Tots els elements de l'estàndard Array són diferents.

Demostració. Suposem que hi ha dos elements iguals, $\mathbf{y_i}$ i $\mathbf{y_j}$. Aquests dos elements poden pertànyer al mateix COSET o a COSET diferents. En el primer cas, com que pertanyen al mateix COSET han de poder ser expressats com:

$$\left.\begin{array}{l}\mathbf{y}_{\mathbf{i}} = \mathbf{e}_{\mathbf{k}} \oplus \mathbf{c}_{\mathbf{i}} \\ \mathbf{y}_{\mathbf{i}} = \mathbf{e}_{\mathbf{k}} \oplus \mathbf{c}_{\mathbf{j}}\end{array}\right\} \Rightarrow \quad y_i = y_j \quad \Rightarrow \quad \mathbf{e}_{\mathbf{k}} \oplus \mathbf{c}_{\mathbf{i}} = \mathbf{e}_{\mathbf{k}} \oplus \mathbf{c}_j \quad \Rightarrow \quad \mathbf{c}_{\mathbf{i}} = \mathbf{c}_{\mathbf{j}}$$

Cosa que resulta absurd, ja que c_i ic_j són, per construcció de l'estàndard Array, paraules codi diferents.

D'altra banda, si suposem que $\mathbf{y_i}$ i $\mathbf{y_j}$ són iguals i pertanyen a diferents COSET, tindrem:

$$e_l \oplus c_i = e_k \oplus c_j \Rightarrow e_l = e_k \oplus \left(c_j \oplus c_i\right) = e_k \oplus c_m$$

On deduïm que és contradictori que pertanyin a diferents COSET.

Exercicis d'autoavualació

1. Es vol realitzar la codificació de 25 bits d'informació útil mitjançant un codi rectangular amb paritat parell. Per a això, els 25 bits s'organitzen en una matriu de cinc files i columnes, com la que es mostra a la figura següent. Se us demana el següent:
a) Determineu els bits de paritat necessaris per a construir el codi rectangular.
b) Calculeu la taxa de redundància.
c) Determineu la taxa del codi.
d) Comparant aquest codi amb el de l'exemple 2, indiqueu quin d'ells té una protecció més gran davant possibles errors i quin d'ells suposa un augment més gran de l'amplada de banda.

2. Suposeu un codificador que admet les paraules codi següents: {0000} {1010} {0101} i {1111}. Els bits transmesos són enviats per un canal amb soroll blanc gaussià a un receptor que detecta un valor de tensió de 2 volts de mitjana quan transmetem un 1 i un valor mitjà de –2 volts quan transmetem un 0. Rebem els valors de tensió {0.1, 2.2, –0.2, 3}. Se us demana el següent:
a) Quins valors decidiríeu que s'han transmès si utilitzeu un descodificador maquinari? Hi ha ambigüitat?
b) Quins valors decidiríeu que s'han transmès si utilitzeu un descodificador programari (mínima distància euclídia)? Hi ha ambigüitat?

3. Enumereu i discutiu les possibles estratègies de correcció i detecció d'errors que es poden realitzar amb un codi de blocs de distància mínima 7.

4. Determineu l'estàndard Array d'un codi (6,3) amb la matriu generadora següent:

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

5. Considereu un codi de bloc lineal (8,3) en què les paraules codi associades als missatges de la base canònica són:

$$\mathbf{e}_1 = [1,0,0] \Rightarrow \mathbf{g}_1 = [1,0,0,1,0,1,0,1]$$
$$\mathbf{e}_2 = [0,1,0] \Rightarrow \mathbf{g}_2 = [0,1,0,0,1,0,1,0]$$
$$\mathbf{e}_3 = [0,0,1] \Rightarrow \mathbf{g}_3 = [0,0,1,0,1,1,1,0]$$

Se us demana el següent:
a) Determineu la matriu generadora del codi lineal.
b) Calculeu la distància mínima del codi lineal.
c) Determineu la matriu de revisió de paritat.
d) Representeu esquemàticament com s'obtenen els bits de redundància a partir dels bits del missatge.

6. Determineu la matriu de revisió de paritat i la matriu generadora d'un codi de Hamming amb paràmetre m = 2. Escriviu les dues paraules codi i comproveu que la distància mínima del codi és 3.

7. Determineu la matriu de revisió de paritat i la matriu generadora d'un codi de Hamming obtingut amb un paràmetre m = 4.

8. Considereu un codi de bloc lineal amb la matriu generadora següent:

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

Determineu:
a) La taxa del codi i la seva distància mínima.
b) La matriu de revisió de paritat.
c) L'estàndard Array.
d) Si es rep el missatge y = 11111, determineu la síndrome i indiqueu si s'ha produït algun error.
e) Descriviu el procediment sistemàtic que utilitzaríeu per a corregir la paraula. Quina és la paraula codi que obtindríeu?

9. Obteniu la taula completa del codi cíclic (7,4) associat al polinomi generador següent:

$$g(p) = p^3 + p^2 + 1$$

Bibliografia

Bibliografia bàsica

Benedetto S.; Biglieri, E. (1999). *Principles of Digital Transmission*. Kluwer Academic Press / Plenum publishers.

Proakis, J. G. (2003). *Digital Communications* (4a. ed.). McGraw Hill.

Proakis, J. G.; Salehi, M. (2002). *Communication Systems Engineering* (2a. ed.). Prentice Hall.

Bibliografia complementària

Carlson, A. B. (2001). *Communication Systems: An Introduction to Signals and Noise in Electrical Communication* (4a. ed.). McGraw Hill.

Gibson Jerry D., i altres (1998). *Digital Compression for Multimedia: Principles & Standards*. Morgan Kauffman.

Stix, G. (1991, setembre). "Encoding the Neatness of Ones and Zeroes". *Scientific American* (pàg. 54-58).