Capitolo 17

Capacità e codifica di canale

Prima di affrontare lo studio dei mezzi trasmissivi, e dopo aver approfondito le tecniche di modulazione analogica e numerica, applichiamo alla *trasmissione* dei segnali i principi di *teoria dell'informazione* esposti al cap. 9. Lo scopo è innanzitutto quello di stabilire *i limiti* entro i quali è possibile operare, ovvero quale sia *il massimo* teorico del tasso di informazione R trasmissibile su un determinato canale *rumoroso*, a cui è dunque associata una probabilità di errore P_e. Tale massimo è noto come *capacità* C del canale, espressa in bit/secondo anche se in definitiva dipende da grandezze di natura continua come *potenza*, *banda*, e *livello di rumore* in ricezione; le conclusioni a cui si giunge sono quindi applicate ai casi studiati di trasmissione analogica e numerica, fornendo la motivazione al compromesso banda-potenza più volte citato. Il capitolo prosegue illustrando le tecniche necessarie per approcciare il più possibile da vicino il limite individuato, ovvero gli algoritmi denominati *codici di canale* che, attraverso l'introduzione (in forma appropriata) di ridondanza, consentono di rivelare e correggere gli errori di trasmissione, a patto di aumentare l'occupazione di banda, od il tempo di trasmissione. In tale ambito sono dettagliati i metodi che ricadono nelle categorie dei codici *a blocco*, *convoluzionali* ed a *decodifica iterativa*.

17.1 Dove arrivare, e come partire

E' più che lecito chiedersi ora di quanto si possa ridurre la P_e, e quanta ridondanza sia necessario aggiungere. La teoria che affronteremo risponde che finché l'intensità informativa $R = f_s \cdot H_s$ in uscita dal codificatore di sorgente (eq. (9.8)) si mantiene inferiore al valore della *capacità di canale* C (§§ 17.2 e 17.3), l'informazione può essere trasportata (teoricamente) *senza errori!* Mentre se al contrario $R > C$, non è possibile trovare nessun procedimento in grado di ridurre gli errori - che anzi, divengono praticamente *certi*. Infine (pur senza spiegare come fare) la teoria assicura che la ridondanza che occorre aggiungere può essere resa *trascurabile!*

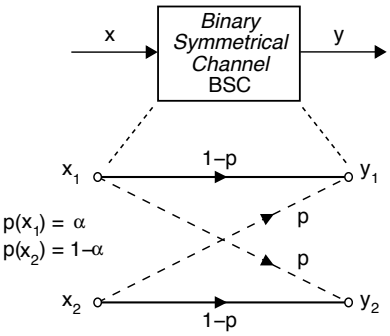
Ma prima di approfondire questi risultati a dir poco *fenomenali*, svolgiamo alcune riflessioni su come

- la probabilità ed il tipo di errori introdotti da un canale numerico possono essere descritti, noto l'ingresso, nei termini di una matrice di *probabilità di transizione*;

- la decisione relativa al simbolo trasmesso si può basare, oltre che sulla conoscenza di tale matrice, anche sulle probabilità di *come sono emessi* i simboli della sorgente;
- al verificarsi di errori corrisponde una *perdita di informazione*.

17.1.1 Canale binario simmetrico

Mentre al § 15.4 si è sviluppato un lungo ragionamento per arrivare ad un valore di probabilità di errore (eq. (15.21)), in questa sede ci riferiamo al solo risultato finale, il valore $P_e^{bit} = p$ che caratterizza il *modello* raffigurato a lato e descritto dal termine BSC o *binary symmetric channel* che rappresenta appunto un canale numerico *binario* con probabilità p di introdurre errore, *indipendentemente* dal simbolo di ingresso, e per questo *simmetrico*.



In termini più formali indichiamo con x_1 e x_2 i due possibili ingressi e, qualora (con prob. $1-p$) non si verifichi errore, con y_1 e y_2 le rispettive uscite, mentre in presenza di errore (con probabilità p), in uscita si presenta il simbolo opposto.

Probabilità a priori Qualora i simboli di ingresso x_1 e x_2 non siano equiprobabili[1], indichiamo con α e $1-\alpha$ le relative prob. a priori (§ 6.1.4).

Probabilità in avanti Individuano le probabilità condizionate $p_{ji} = p\left(y_j/x_i\right)$ di osservare y_j in uscita quando in ingresso è presente x_i, e per questo dette *in avanti*.

Matrice di transizione Π I suoi elementi sono le prob. p_{ji}, e nel caso BSC la matrice è *simmetrica* in quanto

$$\begin{cases} p\left(y_2/x_1\right) = p\left(y_1/x_2\right) = p \\ p\left(y_1/x_1\right) = p\left(y_2/x_2\right) = 1-p \end{cases} \qquad \text{ovvero} \quad \Pi = \left[p_{ji}\right] = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

Osserviamo due cose: la prima è che le prob. $\boldsymbol{p}_y = \left(p\left(y_1\right), p\left(y_2\right)\right)^\top$ dei simboli di uscita si calcolano come $\boldsymbol{p}_y = \Pi \cdot \boldsymbol{p}_x$; la seconda è che le definizioni date si estendono immediatamente al caso di canale $L-$ario, come nel caso multilivello.

17.1.2 Decisione a verosimiglianza ed a posteriori

Il simbolo y_i in uscita dal canale numerico NON È una variabile aleatoria, bensì una osservazione effettiva, e la decisione su quale x_j l'abbia prodotto avviene secondo un procedimento di *verifica di ipotesi* (§ 6.6.1), basata sul valore assunto da un rapporto tra valori di probabilità.

Decisione di massima verosimiglianza Qualora siano note solamente le probabilità in avanti p_{ij} ma non quelle a priori, la decisione avviene sulla base del *rapporto di verosimiglianza* (§ 6.6.2). Supponiamo che l'uscita del BSC sia ad es. il valore y_1: la

[1] Notiamo che in presenza di una codifica di sorgente efficace (PAG. 255) i simboli di ingresso dovrebbero essere pressoché equiprobabili.

decisione su quale delle ipotesi x_1 od x_2 sia più probabile in questo caso avviene in base al rapporto R_{ML} tra le probabilità *in avanti*, e prende il nome di decisione di *massima verosimiglianza* (vedi § 6.6.2.1) o *Maximum Likelihood*, ovvero

$$R_{ML}(y_1) = \frac{p(y_1/x_1)}{p(y_1/x_2)} = \frac{1-p}{p} \underset{x_2}{\overset{x_1}{\gtrless}} 1 \tag{17.1}$$

decidendo quindi per l'ipotesi *più verosimile* in funzione del valore maggiore o minore di uno per R_{ML}. Nel caso risulti $p < \frac{1}{2}$ la regola (17.1) equivale a scegliere l'ingresso concorde con l'uscita, oppure l'opposto se $p > \frac{1}{2}$ (!). Qualora invece si riceva y_2, il rapporto e la relativa regola di decisione sono definiti come $R_{ML}(y_2) = \frac{p(y_2/x_2)}{p(y_2/x_1)} \underset{x_1}{\overset{x_2}{\gtrless}} 1$.

Nel caso di trasmissione $L-$aria, infine, la ricezione di y_j porta alla decisione per $x_{\bar{i}} : \bar{i} = \arg\max_{i=1,2,\cdots L} \left\{ p\left(y_j/x_i\right) \right\}$

Decisione di massima probabilità a posteriori (MAP) Conoscendo anche le probabilità *a priori* $p(x_1)$ e $p(x_2)$, se i due simboli x_1 ed x_2 non sono equiprobabili[2], la decisione può avvenire confrontando le probabilità *a posteriori*[3] $p\left(x_j/y_i\right)$, calcolabili applicando il teorema di Bayes (vedi § 6.1.4). Facendo di nuovo il caso di aver ricevuto il simbolo y_1, scriviamo dunque

$$\begin{aligned} R_{MAP}(y_1) &= \frac{p(x_1/y_1)}{p(x_2/y_1)} = \frac{p(y_1/x_1)\,p(x_1)}{p(y_1)} \cdot \frac{p(y_1)}{p(y_1/x_2)\,p(x_2)} = \\ &= \frac{p(y_1/x_1)}{p(y_1/x_2)} \frac{p(x_1)}{p(x_2)} \underset{x_2}{\overset{x_1}{\gtrless}} 1 \end{aligned} \tag{17.2}$$

o più in generale, comprendendo anche il caso di canale $L-$ario, il criterio di decisione MAP qualora si riceva y_i è espresso come

$$x_{\bar{i}} : \bar{i} = \arg\max_{i=1,2,\cdots L} \left\{ p\left(y_j/x_i\right) p(x_i) \right\}$$

Il modo con cui le probabilità *a priori* $p(x_1)$ e $p(x_2)$ correggono la decisione ML (17.1) in MAP (17.2) per un BSC si presta a due osservazioni

- x_1 potrebbe essere *così raro* che, in presenza di una moderata probabilità di errore, si preferisce decidere sempre x_2, attribuendo l'eventuale ricezione di y_1 ad un errore del canale, piuttosto che all'effettiva trasmissione di x_1.
- in assenza di canale (ossia senza ricevere nulla) l'unica decisione possibile si basa sul confronto tra le p. a priori $p(x_1)$ e $p(x_2)$. La ricezione di un simbolo y_i apporta nuova informazione, alterando il rapporto di decisione R in misura tanto maggiore quanto minore è la probabilità di errore.

[2] In caso contrario (ovvero $p(x_1) = p(x_2) = 0.5$) la (??) è equivalente alla (17.1). Nei casi in cui *non si conoscano* le prob. a priori, non si può quindi fare altro che attuare una *decisione di massima verosimiglianza*.

[3] Sono indicate come *a posteriori* perché misurano la probabilità del simbolo trasmesso x *dopo* la conoscenza di quello ricevuto y.

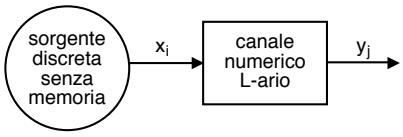
Esempio Verifichiamo le ultime osservazioni esplicitando una probabilità a posteriori in funzione di p:

$$\begin{aligned} p(x_1/y_1) &= \frac{p(x_1,y_1)}{p(y_1)} = \frac{p(y_1/x_1)\,p(x_1)}{p(y_1/x_1)\,p(x_1)+p(y_1/x_2)\,p(x_2)} = \\ &= \frac{(1-p)\cdot p(x_1)}{(1-p)\cdot p(x_1)+p\cdot p(x_2)} = \frac{p(x_1)}{p(x_1)+\frac{p}{1-p}p(x_2)} \end{aligned}$$

Se $p = 1-p = \frac{1}{2}$, il canale *è inservibile* e non trasferisce informazione: infatti si ottiene $p(x_1/y_1) = p(x_1)$ pari a quella a priori, in quanto $p(x_1)+p(x_2) = 1$. D'altra parte se $p < \frac{1}{2}$ si ottiene $p(x_1/y_1) > p(x_1)$ dato che ora $\frac{p}{1-p} < 0.5$: si assiste pertanto ad un *aumento* della probabilità di x_1 rispetto a quella a priori; se poi la probabilità di errore tende a zero ($p \to 0$) si ottiene $p(x_1/y_1) \to 1$.

17.1.3 Informazione mutua media per canale numerico $L-$ario

Approfondiamo questa nozione introdotta al § 9.4.3 e li utilizzata per definire la funzione velocità distorsione (§ 9.5.2), mostrando come *l'informazione condivisa* tra ingresso ed uscita di un canale consenta di determinare anche la quantità di informazione che viene *persa* a causa degli errori che si sono verificati.



Consideriamo una sorgente discreta che emette simboli x appartenenti ad un alfabeto finito di cardinalità L, ossia $x \in \{x_i\}$ con $i = 1, 2, \cdots, L$, ed indichiamo con $y \in \{y_j\}$ (sempre per $j = 1, 2, \cdots, L$) il corrispondente simbolo ricevuto mediante un canale discreto, in generale diverso da x, a causa di errori introdotti dal canale. Conoscendo le densità di probabilità $p(x_i)$, $p(y_j)$, e le probabilità congiunte $p(x_i, y_j)$, possiamo definire la quantità di informazione *in comune* tra x_i e y_j, denominata *informazione mutua*, come[4]

$$I(x_i,y_j) = \log_2 \frac{p(x_i,y_j)}{p(x_i)\,p(y_j)} = \log_2 \frac{p(x_i/y_j)}{p(x_i)} = \log_2 \frac{p(y_j/x_i)}{p(y_j)} \quad \text{bit} \tag{17.3}$$

da cui deriva che

1. se ingresso ed uscita del canale sono *statisticamente indipendenti* si ha $p(x_i,y_j) = p(x_i)\,p(y_j)$, e di conseguenza l'informazione mutua *è nulla;*
2. se $p(y_j/x_i) > p(y_j)$ significa che l'essere a conoscenza della trasmissione di x_i rende la ricezione di y_j *più probabile* di quanto non lo fosse a priori, e corrisponde ad una informazione mutua *positiva*;
3. la definizione di informazione mutua è *simmetrica*, ovvero $I(x_i,y_j) = I(y_j,x_i)$;
4. rifrasando la 2. in virtù della 3., se $p(x_i/y_j) > p(x_j)$ allora ricevere y_j rende la trasmissione di x_i *più probabile* di quanto non lo fosse a priori, manifestando lo stesso valore di informazione mutua *positiva* del punto 2.

Per giungere ad una grandezza $I(X,Y)$ che tenga conto del comportamento *medio* del canale, ovvero per coppie ingresso-uscita qualsiasi, occorre pesare i valori di $I(x_i,y_j)$

[4]Per ottenere le diverse forme della (17.3) si ricordi che $p(x_i,y_j) = p(x_i/y_j)\,p(y_j) = p(y_j/x_i)\,p(x_i)$

con le relative probabilità congiunte, ossia calcolarne il valore atteso rispetto a tutte le possibili coppie (x_i, y_j):

$$I(X,Y) = E_{X,Y}\left\{I\left(x_i, y_j\right)\right\} = \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{p\left(x_i/y_j\right)}{p\left(x_i\right)} \tag{17.4}$$

$$= \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{p\left(y_j/x_i\right)}{p\left(y_j\right)} \tag{17.5}$$

ri-ottenendo così l'*informazione mutua media* (§ 9.4.3), misurata in bit/simbolo, e che rappresenta (in media) quanta informazione ogni simbolo ricevuto trasporta a riguardo di quello trasmesso. In virtù della simmetria di questa definizione, ci accorgiamo che il valore di $I(X,Y)$ può essere espresso[5] nelle due forme alternative

$$I(X,Y) = H(X) - H(X/Y) \tag{17.6}$$

$$= H(Y) - H(Y/X) \tag{17.7}$$

in cui l'entropia *condizionale* (§ 9.4.2)

$$H(X/Y) = \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{1}{p\left(x_i/y_j\right)} \tag{17.8}$$

prende il nome di *equivocazione* e rappresenta la quantità media di informazione *persa*, rispetto all'entropia di sorgente $H(X)$, a causa della rumorosità del canale. Nel caso in cui il canale non introduca errori, e quindi $p\left(x_i/y_j\right)$ sia pari a 1 se $j = i$ e zero altrimenti, è facile vedere[6] che $H(X/Y)$ è pari a zero, e $I(X,Y) = H(X)$, ossia tutta l'informazione della sorgente si trasferisce a destinazione. D'altra parte

$$H(Y/X) = \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{1}{p\left(y_j/x_i\right)} \tag{17.9}$$

prende il nome di *noise entropy* dato che considera il processo di rumore come se fosse un segnale informativo: infatti, sebbene si possa essere tentati di dire che l'informazione media ricevuta è misurata dalla entropia $H(Y)$ della sequenza di osservazione, una parte di essa $H(Y/X)$ *è falsa*, perché in realtà è introdotta dagli errori.

Calcolo dell'informazione mutua media per il BSC Torniamo al caso binario descritto al § 17.1.1 ed usiamo la (17.7) per calcolare l'informazione mutua media in funzione della probabilità a priori $p(x_1) = \alpha$ e di quella in avanti p_e, valutando innanzitutto $H(Y)$ e $H(Y/X)$. Dal punto di vista dell'uscita del canale, i simboli y_1, y_2 costituiscono l'alfabeto di una sorgente binaria senza memoria, la cui entropia si

[5] Infatti

$$\sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{p\left(x_i/y_j\right)}{p\left(x_i\right)} = \sum_i \sum_j p\left(x_i, y_j\right) \left[\log_2 \frac{1}{p\left(x_i\right)} - \log_2 \frac{1}{p\left(x_i/y_j\right)}\right] =$$

$$= \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{1}{p\left(x_i\right)} - \sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{1}{p\left(x_i/y_j\right)}$$

L'ultimo termine è indicato come entropia condizionale $H(X/Y)$ (eq. (17.8)), mentre il penultimo è pari all'entropia di sorgente $H(X)$ dato che *saturando* la prob. congiunta $p\left(x_i, y_j\right)$ rispetto ad j, ovvero $\sum_j p\left(x_i, y_j\right) = p\left(x_i\right)$, si perviene alla (17.6) in base al risultato $\sum_i \log_2 \frac{1}{p(x_i)} \sum_j p\left(x_i, y_j\right) = \sum_i p\left(x_i\right) \log_2 \frac{1}{p(x_i)}$. Per la (17.7) il passaggio è del tutto simile.

[6] Infatti in tal caso la (17.8) diviene $\sum_i \sum_j p\left(x_i, y_j\right) \log_2 \frac{1}{p(x_i/y_j)} = \sum_i p\left(x_i, y_i\right) \log_2 1 = 0$

esprime in termini di $p(y_1)$ mediante la (9.6), ovvero $H(Y) = H_b(p(y_1))$, in cui

$$\begin{aligned} p(y_1) &= p(y_1/x_1)\,p(x_1) + p(y_1/x_2)\,p(x_2) = \\ &= (1-p_e)\,\alpha + p_e\,(1-\alpha) = p_e + \alpha - 2\alpha p_e \end{aligned}$$

e dunque $H(Y) = H_b(p_e + \alpha - 2\alpha p_e)$. Per quanto riguarda la *noise entropy* $H(Y/X)$, sostituendo $p(x_i, y_j) = p(y_j/x_i)\,p(x_i)$ nella (17.9) otteniamo

$$H(Y/X) = \sum_i p(x_i) \left[\sum_j p(y_j/x_i) \log_2 \frac{1}{p(y_j/x_i)} \right] = H_b(p_e)$$

dato che il termine tra parentesi quadre rappresenta appunto l'entropia di una sorgente binaria con simboli a probabilità p_e e $1 - p_e$. Possiamo quindi ora scrivere l'espressione cercata

$$I(X,Y) = H(Y) - H(Y/X) = H_b(p_e + \alpha - 2\alpha p_e) - H_b(p_e) \tag{17.10}$$

che dipende sia dalla probabilità di errore p_e, sia dalla prob. a priori dei simboli della sorgente: osserviamo che se $p_e \ll 1$ il canale (quasi) non commette errori, e risulta $I(X,Y) \simeq H_b(\alpha) = H(X)$, mentre se $p_e \to \frac{1}{2}$ allora $I(X,Y) \to 0$.

17.2 Capacità di canale discreto

Le relazioni fin qui discusse permettono di valutare la perdita di informazione causata dai disturbi, ma dipendono sia dalle probabilità *in avanti* $p(y_j/x_i)$ che descrivono il comportamento del canale, sia da quelle *a priori* $p(x_i)$, che invece attengono alle caratteristiche della sorgente. Vogliamo invece definire una grandezza che esprima esclusivamente l'attitudine (o *capacità*) del canale a trasportare informazione, indipendentemente dalle caratteristiche della sorgente. Questo risultato può essere ottenuto variando le prob. a priori in tutti i modi possibili, fino a trovare il valore

$$C_s = \max_{p(x)} I(X,Y) \quad \text{bit/simbolo} \tag{17.11}$$

che definisce la *capacità di canale per simbolo* come il massimo valore dell'informazione mutua media, ottenuto in corrispondenza della migliore sorgente possibile. Il pedice $_s$ sta per *simbolo*, e serve a distinguere il valore ora definito da quello che esprime la massima *intensità* di trasferimento dell'informazione espressa in bit/secondo, ottenibile una volta nota la frequenza f_s con cui sono trasmessi i simboli, fornendo per la capacità di canale il nuovo valore[7]

$$C = f_s \cdot C_s \quad \text{bit/secondo} \tag{17.12}$$

L'importanza di questa quantità risiede nel *teorema fondamentale per canali rumorosi*[8] già anticipato più volte, che asserisce che per ogni canale discreto senza memoria di capacità C

- esiste una tecnica di codifica che consente la trasmissione di informazione a velocità R e con probabilità di errore per simbolo p_e *piccola a piacere*, purché

[7] Notiamo l'invarianza di (17.12) rispetto al numero di livelli con cui è effettuata la trasmissione: se M bit sono raggruppati per generare simboli ad $L = 2^M$ livelli, come noto f_s si riduce di M volte, mentre C_s aumenta della stessa quantità, dato che ogni simbolo trasporta ora M bit anziché uno.

[8] `http://it.wikipedia.org/wiki/Secondo_teorema_di_Shannon`

risulti $R < C$;

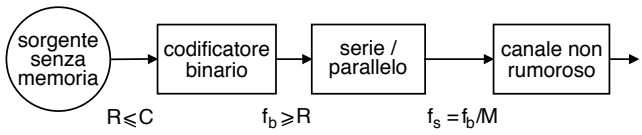
- se è accettabile una probabilità di errore p_e, si può raggiungere (con la miglior codifica possibile) una velocità $R(p_e) = \frac{C}{1-H_b(p_e)} > C$ in cui $H_b(p_e)$ è l'entropia di una sorgente binaria (9.6);
- per qualsiasi valore di p_e, non è possibile trasmettere informazione a velocità maggiore di $R(p_e)$.

Il teorema non suggerisce come individuare la tecnica di codifica, né fa distinzioni tra codifica di sorgente e di canale, ma indica le prestazioni limite ottenibili mediante la migliore tecnica possibile, in grado di ridurre a piacere la p_e purché $R < C$, mettendoci al tempo stesso in guardia a non tentare operazioni impossibili. Da questo punto di vista, le prestazioni conseguibili adottando le tecniche di codifica note possono essere valutate confrontandole con quelle *ideali* predette dal teorema. Inoltre, dato che la capacità di canale è definita come massimo valore di $I(X,Y)$ per la migliore $p(x)$, qualora la statistica dei messaggi prodotti dal codificatore di sorgente differisca da quella ottima per il canale, l'effettiva informazione mutua media risulterà ridotta rispetto al valore della capacità, così come la massima velocità R.

Illustriamo l'applicazione di questi risultati con un paio di esempi.

17.2.1 Capacità di un canale $L-$ario non rumoroso

Consideriamo il caso mostrato in figura, ovvero un canale che trasporta *senza errori* simboli con $L = 2^M$ livelli: in tal caso l'equivocazione $H(Y/X)$ è nulla, e la (17.6) permette di scrivere $I(X,Y) = H(X)$, che è massima se $P(x_i) = 1/L$ per tutti gli i, risultando così $C_s = H_{max}(X) = \log_2 L = M$ bit/simbolo, e $C = f_s \cdot C_s = f_s \cdot M$ bit/secondo.



I simboli ad L livelli sono ottenuti raggruppando M dei bit prodotti da una codifica binaria a velocità f_b, risultando $f_b \geq R = H_x$ (vedi eq. (9.9)) in funzione della ottimalità o meno del codificatore; pertanto, risulta $R \leq f_b = f_s \cdot M = C$ con l'uguaglianza valida nel caso in cui il codificatore riesca a rimuovere tutta la ridondanza dei messaggi della sorgente[9], conseguendo in tal caso il massimo trasferimento di informazione.

Al contrario, volendo realizzare una velocità $R > C$, il codificatore di sorgente dovrebbe produrre codeword con lunghezze tali da violare la disuguaglianza di Kraft (9.13)[10], e quindi la regola del prefisso non sarebbe rispettata, causando in definitiva errori di decodifica anche in assenza di rumore!

17.2.2 Capacità del canale binario simmetrico

Esaminiamo l'effetto della presenza di rumore per questo caso particolare, per il quale a pag. 556 abbiamo valutato l'espressione dell'informazione mutua media, data dalla

[9] Ad esempio se L non è una potenza di due, un codificatore di sorgente che operi simbolo per simbolo produce necessariamente $f_b > R$, mentre se concatena più simboli (§ 9.1.4), può avvicinarsi a $f_b = R$.

[10] Infatti, potrebbe risultare $R > C$ solo se $f_b < R$, ovvero il codificatore dovrebbe produrre *meno* binit/secondo di quanti bit/secondo produca la sorgente

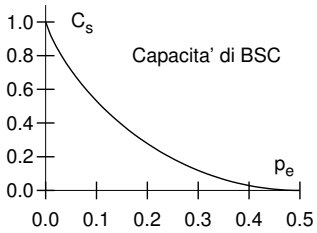
(17.10), e pari a

$$I(X,Y) = H_b(p_e + \alpha - 2\alpha p_e) - H_b(p_e)$$

in cui $H_b(p_e)$ dipende solo dalla probabilità di errore, mentre il termine $H_b(p_e + \alpha - 2\alpha p_e)$ dipende anche dalla statistica di sorgente, e risulta massimizzato e pari ad 1 se $p_e + \alpha - 2\alpha p_e = \frac{1}{2}$, come avviene per qualunque p_e se $\alpha = \frac{1}{2}$, ossia per simboli equiprobabili. Pertanto la capacità del BSC risulta pari a

$$C_s = H_b(1/2) - H_b(p_e) = 1 - H_b(p_e)$$

il cui grafico è rappresentato alla figura a lato[11], evidenziando che $C_s \simeq 1$ bit/simbolo se $p_e \simeq 0$, ma che poi decade rapidamente a zero se $p_e \to 0.5$.

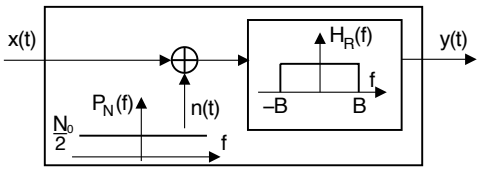


Quest'ultimo esempio in particolare ci conferma l'esigenza, in presenza di un canale rumoroso, di attuare tecniche di codifica di canale in grado di ridurre la probabilità di errore, in modo da poter sfruttare appieno la capacità che il canale presenta nel caso di p_e ridotta, e di preferire tra queste le tecniche che vi riescono mantenendo al minimo la quantità dei bit aggiuntivi, dato che altrimenti come noto aumenta la banda occupata dal segnale dati.

17.3 Capacità di canale continuo

Come anticipato fin dal § 1.2.2 un canale numerico è in realtà *una astrazione* che ingloba internamente un codificatore di linea o *modem* che, a partire da una sequenza numerica, produce un segnale trasmissibile su di un canale analogico, che a sua volta può essere caratterizzato da un valore di capacità, espresso nei termini dei parametri che descrivono la trasmissione analogica soggiacente.

Canale gaussiano additivo bianco Una situazione tipica è quella raffigurata a lato, in cui al segnale ricevuto è sommato un rumore $n(t)$ gaussiano, bianco e a media nulla, mentre il filtro di ricezione $H_R(f)$ impone una limitazione di banda $2B$, in modo che la potenza di rumore in ingresso al decisore vale $P_n = \sigma_n^2 = N_0 B$. Tale situazione viene indicata come *canale* AWGN (*additive white gaussian noise*) *limitato in banda.*



Calcolo della capacità Indicando con $p(x)$, $p(y)$, $p(x/y)$, $p(y/x)$ le d.d.p. marginali e condizionali che descrivono *un campione* dei processi di ingresso $x(t)$ ed uscita $y(t)$, entrambi limitati in banda $\pm B$, l'applicazione formale della (17.4) al caso continuo porta a scrivere l'espressione dell'informazione mutua media come

$$I(X,Y) = \int \int_{-\infty}^{\infty} p_{XY}(x,y) \log_2 \frac{p_Y(y/x)}{p_Y(y)} dx dy \quad \text{bit/campione} \tag{17.13}$$

[11] Sono mostrati solo i valori per $0 \le p_e \le 0.5$ dato che successivamente l'andamento di C_s si riflette in modo speculare.

che è una misura *assoluta*[12] del trasferimento di informazione per campione di uscita. Il massimo valore di (17.13) al variare di $p_X(x)$ consente anche questa volta di definire la capacità di canale per campione $C_s = \max_{p(x)} I(X,Y)$; in virtù della limitazione di banda, i campioni prelevati ad una frequenza di campionamento $f_c = 2B$ risultano indipendenti tra loro (vedi § 7.2.4), cosicché la capacità di canale risulta definita come

$$C = 2B \cdot \max_{p(x)} \{I(X,Y)\} \quad \text{bit/secondo} \tag{17.14}$$

Riscrivendo la (17.13) nella forma

$$I(X,Y) = h(Y) - h(Y/X) \tag{17.15}$$

si ottiene una espressione analoga alla (17.7) ma i cui termini sono ora da intendersi come entropia differenziale, definita al § 9.3.1. Osserviamo ora che il termine di *noise entropy* $h(Y/X) = \int\int_{-\infty}^{\infty} p_{XY}(x,y) \log_2 \frac{1}{p_Y(y/x)} dx dy$ dipende esclusivamente dal rumore additivo, in quanto $y(t) = x(t) + n(t)$ e quindi $p_Y(y/x) = p_N(x+n)$: infatti $p_Y(y/x)$ altro non è che la gaussiana del rumore, a cui si somma un valor medio fornito dal campione di x; quindi $h(Y/X)$ si riduce all'entropia differenziale di un processo gaussiano (9.20), che non dipende dal valor medio, ma solo dall'andamento di $p_N(n)$; pertanto

$$h(Y/X) = \int_{-\infty}^{\infty} p_N(n) \log_2 \frac{1}{p_N(n)} dn = \frac{1}{2} \log_2 \left(2\pi e \sigma_n^2\right) \tag{17.16}$$

come risulta per l'entropia differenziale di sorgenti gaussiane (9.20). Quindi ora il termine della (17.15) che deve essere massimizzato rispetto a $p(x)$ è solo il primo, ossia $h(Y)$, che come sappiamo, è massimo se $y(t)$ è gaussiano. Dato che il processo ricevuto $y(t)$ è composto da due termini $x(t) + n(t)$ di cui il secondo è già gaussiano, si ottiene $y(t)$ gaussiano a condizione che anche $x(t)$ sia gaussiano. Indicando con σ_x^2 la potenza di quest'ultimo, ed in virtù della indipendenza statistica tra $x(t)$ e $n(t)$, risulta $\sigma_y^2 = \sigma_x^2 + \sigma_n^2$, e quindi

$$h(Y) = \tfrac{1}{2} \log_2 \left[2\pi e \left(\sigma_x^2 + \sigma_n^2\right)\right] \tag{17.17}$$

cosicché mettendo assieme (17.15), (17.16) e (17.17), la (17.14) si riscrive come

$$\begin{aligned} C &= 2B \cdot \left\{ \tfrac{1}{2} \log_2 \left[2\pi e \left(\sigma_x^2 + \sigma_n^2\right)\right] - \tfrac{1}{2} \log_2 \left(2\pi e \sigma_n^2\right) \right\} = \\ &= B \cdot \log_2 \frac{\sigma_x^2 + \sigma_n^2}{\sigma_n^2} = B \cdot \log_2 \left(1 + \frac{P_x}{P_n}\right) \quad \text{bit/secondo} \end{aligned}$$

che è proprio il risultato tanto spesso citato, che prende il nome di *legge di Shannon-Hartley*[13] e che esprime la capacità di canale per un canale additivo gaussiano. Tenendo conto che $P_n = \sigma_n^2 = N_0 B$ e che P_x è la potenza del segnale ricevuto P_s, riscriviamo l'espressione della capacità nella sua forma più nota:

$$C = B \cdot \log_2 \left(1 + \frac{P_s}{N_0 B}\right) \quad \text{bit/secondo} \tag{17.18}$$

[12] Per il fatto di avere una ddp di y sia a numeratore che a denominatore del logaritmo, la (17.13) non soffre dei problemi discussi alla nota 26 a pag. 267.

[13] `http://en.wikipedia.org/wiki/Shannon-Hartley_theorem`

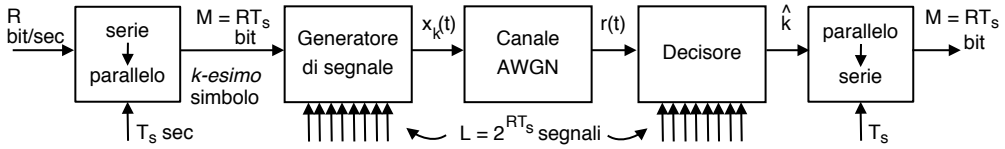


Figura 17.1: Schema ideale di codifica di canale ad errore asintoticamente nullo

che, associata al teorema fondamentale della codifica espresso al § 17.2, stabilisce il massimo tasso informativo trasmissibile senza errori su di un canale AWGN limitato in banda come $R \leq B \cdot \log_2 (1 + {}^{P_s}/_{N_0 B})$. Discutiamo ora delle conseguenze di questo risultato.

17.3.1 Sistema di comunicazione ideale

Una volta noto il massimo tasso di informazione $R < C$ che il canale può trasportare senza errori, come fare per evitare, appunto, questi ultimi? Il metodo suggerito da Shannon, anziché introdurre ridondanza come avviene per le tecniche di codifica di canale classiche, effettua invece la trasmissione semplicemente ripartendo l'informazione in blocchi codificati mediante simboli di durata elevata. In pratica, si tratta di realizzare una sorta di *trasmissione multilivello* (vedi § 15.1.2.4) come mostrato alla figura 17.1 dove l'informazione generata ad una velocità R bit/secondo viene trasmessa mediante simboli emessi con periodo T_s secondi, ognuno dei quali rappresenta un gruppo di $M = RT_s$ bit, e dunque occorrono $L = 2^M$ simboli diversi.

Nella dimostrazione di Shannon ogni simbolo, anziché essere rappresentato da un valore costante come nella trasmissione multilivello, è costituito da un segnale $x_k(t)\,,\ k = 1, 2, \ldots, L$ di durata T_s, ottenuto prelevando una finestra temporale T_s da una realizzazione di processo gaussiano bianco limitato in banda. Il ricevitore possiede una copia di tali forme d'onda, e per ogni periodo di simbolo calcola l'errore quadratico $\varepsilon_k = \frac{1}{T_s} \int_0^{T_s} (r(t) - x_k(t))^2\, dt$ tra il segnale ricevuto $r(t)$ ed ognuna delle forme d'onda associate ai simboli, decidendo per la trasmissione del simbolo $\hat{k}$ la cui forma d'onda $x_{\hat{k}}(t)$ fornisce l'errore ε_k minimo. Mantenendo R fisso e pari al tasso informativo della sorgente, all'aumentare di T_s anche $M = RT_s$ aumenta di pari passo, mentre il numero di simboli $L = 2^M$ aumenta esponenzialmente. Claude Shannon ha dimostrato[14] che, per $T_s \to \infty$, lo schema indicato riesce effettivamente a conseguire una $P_e \to 0$, tranne per il piccolo particolare che... occorre attendere un tempo che tende a infinito!

[14] Senza pretendere di svolgere l'esatta dimostrazione, tentiamo di dare credibilità a questo risultato. Osserviamo quindi che se $r(t) = x_k(t) + n(t)$, il valore atteso dell'errore ε_k si riduce a $\frac{1}{T_s} \int_0^{T_s} [n(t)]^2\, dt \to \sigma_n^2$, dato che essendo $n(t)$ stazionario ergodico, le medie di insieme coincidono con le medie temporali. Viceversa, se il segnale trasmesso è $x_h(t)$ con $h \neq k$, allora il relativo errore quadratico vale $\varepsilon_k^{(h)} = \frac{1}{T_s} \int_0^{T_s} (x_h(t) + n(t) - x_k(t))^2\, dt$, ed il suo valore atteso $E\left\{\varepsilon_k^{(h)}\right\} \to \sigma_n^2 + 2\sigma_x^2$ essendo le forme d'onda dei simboli ortogonali tra loro e rispetto al rumore. I valori limite mostrati sono in realtà grandezze aleatorie, ma la loro varianza diviene sempre più piccola all'aumentare di T_s, e quindi in effetti con $T_s \to \infty$ risulta sempre $\varepsilon_k < \varepsilon_k^{(h)}$, azzerando la probabilità di errore.

17.3.2 Minima energia per bit

In realtà uno schema di trasmissione numerica che approssima piuttosto bene quello ideale discusso al § precedente esiste veramente, ed è quello esposto al § 16.5.1 ed denominato FSK ortogonale, in cui le forme d'onda di fig. 17.1 sono sinusoidali: il grafico delle sue prestazioni a pag. 512 mostra infatti come, aumentando L, lo stesso valore di E_b/N_0 permetta di conseguire valori di P_e via via più piccoli. Lo stesso grafico mostra però l'esistenza di un valore limite sotto cui E_b/N_0 non può scendere, dovendo comunque risultare

$$\frac{E_b}{N_0} \geq \ln 2 = 0,693 \quad \text{ovvero} \quad \left.\frac{E_b}{N_0}\right|_{dB} \geq -1.6 \text{ dB} \tag{17.19}$$

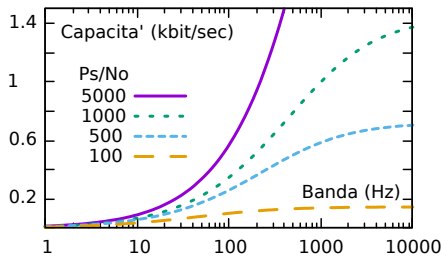
Ciò deriva dall'occupazione di banda via via crescente necessaria all'FSK qualora L aumenti: considerando che la capacità di canale per $B \to \infty$ fornita dalla (17.20) vale $C_\infty = \frac{P_s}{N_0 \ln 2}$, e che deve risultare $R \leq C$, risulta allora $\ln 2 = \frac{P_s}{N_0 C_\infty} \leq \frac{P_s}{N_0 R} = \frac{E_b}{N_0}$, ovvero la (17.19).

Ma per arrivare all'espressione di C_∞ ora citata, affrontiamo il prossimo §.

17.3.3 Compromesso banda-potenza e capacità massima

Il valore limite (17.19) trae origine da una conseguenza della (17.18) già fatta notare al § 15.4.7, ovvero la possibilità di risparmiare potenza aumentando l'occupazione di banda (o viceversa), dato che in entrambi i casi a ciò corrisponde un aumento di C. Ma ciò non avviene all'infinto, ovvero *non si può oltrepassare* un valore massimo di capacità! Infatti se nella (17.18) si aumenta B il filtro di ricezione si *allarga*, e dunque aumenta la potenza di rumore, e l'effetto finale è che per un canale con *banda infinita* non si ottiene una capacità infinita, bensì il valore

$$\begin{aligned} C_\infty &= \lim_{B\to\infty} B \cdot \log_2\left(1 + \frac{P_s}{N_0 B}\right) = \\ &= \frac{P_s}{N_0 \ln 2} \simeq 1.44 \frac{P_s}{N_0} \end{aligned} \tag{17.20}$$



che individua anche il limite *assoluto* al massimo tasso informativo R trasmissibile. In figura è mostrato l'andamento effettivo della (17.18) in funzione di B, per alcuni valori di $\frac{P_s}{N_0}$ di esempio, mentre la dimostrazione della (17.20) è riportata alla nota[15].

[15] La (17.20) si ottiene riscrivendo la (17.18) nella forma

$$C = \frac{P_s}{N_0 \frac{P_s}{N_0 B}} \cdot \frac{\ln\left(1 + \frac{P_s}{N_0 B}\right)}{\ln 2} = \frac{P_s}{N_0 \ln 2} \cdot \frac{\ln(1+\lambda)}{\lambda}$$

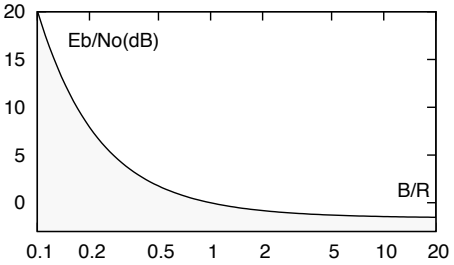
in cui ln è il logaritmo *naturale* in base e, e si è posto $\frac{P_s}{N_0 B} = \lambda$. Ricordando ora lo sviluppo di Maclaurin $f(x) = f(0) + \sum_{n=1}^{\infty}\left(\left.\frac{\partial^n f(x)}{\partial x^n}\right|_{x=0} \cdot \frac{x^n}{n!}\right)$ e che $\frac{d}{dx}\ln x = \frac{1}{x}$, il termine $\ln(1+\lambda)$ può essere espanso in serie di potenze come $\ln(1+\lambda) = \lambda - \frac{1}{2}\lambda^2 + \frac{1}{3}\lambda^3 + \cdots$; notando infine che per $B \to \infty$ si ha $\lambda \to 0$, e che $\lim_{\lambda\to 0}\frac{\ln(1+\lambda)}{\lambda} = 1$, si giunge in definitiva al risultato (17.20).

17.3.4 Limite inferiore per $\frac{E_b}{N_0}$

Una volta assegnato il tasso informativo $R \leq C$ della sorgente e la banda B del canale, partendo dalla (17.18) si può ottenere[16] una relazione che esprime il valore di $\frac{E_b}{N_0}$ necessario a conseguire una trasmissione senza errori (nel caso ideale):

$$\frac{E_b}{N_0} \geq \frac{B}{R}\left(2^{\frac{R}{B}} - 1\right) \tag{17.21}$$

e che, espressa in dB, è graficata alla figura a lato, in cui l'area grigia indica i valori di $\frac{E_b}{N_0}$ *vietati*, ossia per i quali è impossibile ottenere una trasmissione senza errori.



Compromesso banda-potenza per un sistema ideale

Mentre per $\frac{B}{R} = 1$ il sistema ideale richiede un valore di $\frac{E_b}{N_0}$ pari ad almeno 0 dB, questo si riduce nel caso in cui la trasmissione occupi una banda maggiore del tasso informativo R, fino a raggiungere (già per valori $B > 10R$) il limite (17.19) di -1.6 dB. D'altra parte, qualora la trasmissione impegni una banda inferiore ad R, il valore di $\frac{E_b}{N_0}$ necessario aumenta in modo piuttosto brusco.

17.3.5 Confronto con le prestazioni di sistemi di modulazione reali

E' possibile svolgere una verifica sperimentale della relazione (17.21) prendendo in considerazione le tecniche di modulazione numerica discusse ai capitoli precedenti, e che consentono di variare l'occupazione di banda B per trasmettere ad una data velocità $R = f_b$, ad esempio riducendone il rapporto B/R come nelle trasmissione multilivello[17], oppure aumentandolo, come nel caso dell'FSK. In questi casi il valore di $\frac{E_b}{N_0}$ necessario a conseguire una determinata prestazione (P_e) varia in funzione del rapporto B/R, e dunque può essere messo a confronto con i valori minimi di $\frac{E_b}{N_0}$ previsti dalla (17.21), come avviene nella figura 17.2 che mostra i valori di E_b/N_0 in funzione di B/R per le tecniche di modulazione numerica QAM (§ 16.3.1) e FSK ortogonale (pag. 510). Per tracciare la figura si sono ricavati i valori di E_b/N_0 necessari a ciascun metodo per ottenere una P_e pari a 10^{-5} per diversi valori di L, e messi in relazione con l'occupazione spettrale associata $B(L)$ rapportata alla velocità f_b, ossia in relazione all'efficienza spettrale ρ (pag. 494) dei metodi.

Considerando di adottare per il QAM un impulso di Nyquist a banda minima, la banda occupata risulta pari a $B_{QAM} = \frac{f_b}{\log_2 L}$, e pertanto $\frac{B}{R}\Big|_{QAM} = \frac{1}{\log_2 L}$; invece come riportato a pag. 512 per l'FSK ortogonale si ha $B_{FSK} \simeq \frac{f_b}{2}\frac{L}{\log_2 L}$, e dunque $\frac{B}{R}\Big|_{FSK} = \frac{L}{2\log_2 L}$. Possiamo osservare come per le due tecniche di trasmissione l'andamento dei valori di

[16] Riscrivendo la (17.18) come $2^{\frac{C}{B}} - 1 = \frac{P_s}{N_0 B}$, moltiplicando ambo i membri per $\frac{B}{R}$, e semplificando il risultato, si ottiene $\frac{B}{R}(2^{\frac{C}{B}} - 1) = \frac{P_s}{N_0 R}$. L'uguaglianza individua la circostanza limite in cui $R = C$, mentre se nell'esponente di 2 a primo membro sostituiamo C con R, e $R \leq C$, il primo membro diviene più piccolo, e pertanto $\frac{B}{R}(2^{\frac{R}{B}} - 1) \leq \frac{P_s}{N_0 R}$. Infine, notiamo che $\frac{P_s}{N_0 R} = \frac{E_b}{N_0}$, da cui il risultato mostrato (17.21).

[17] Vedi ad es. il caso di banda base al § 15.4.9 o quello del QAM al § 16.3.1.

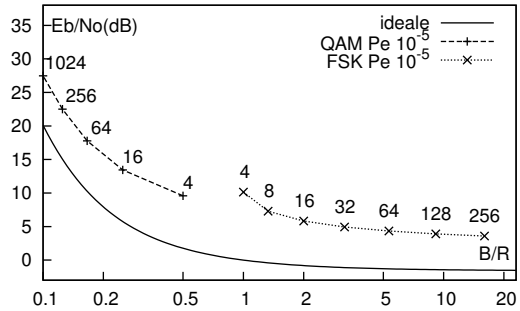


Figura 17.2: Rapporto E_b/N_0 di QAM ed FSK per $P_e = 10^{-5}$ al variare di L, in funzione della efficienza spettrale, confrontato con i valori minimi teorici

$\frac{E_b}{N_0}$ in funzione di $\frac{B}{R}$ ricalchi abbastanza fedelmente quello ideale, a parte una perdita di efficienza, che si riduce per L crescente.

17.4 Codifica di canale

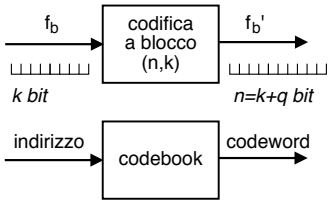
Dopo aver analizzato i risultati che la teoria dell'informazione fornisce a riguardo delle migliori prestazioni ottenibili, proseguiamo il discorso iniziato al § 15.6.2.1 su come aggiungere ridondanza ad un flusso binario a velocità f_b da trasmettere su di un canale numerico, in modo da realizzare una protezione FEC capace di ridurre la probabilità di errore per bit P_e in ricezione. Ricordiamo che al § 17.2 abbiamo mostrato come, aumentando il ritardo di codifica, la probabilità di errore può essere resa piccola a piacere, purché $f_b = R < C$, essendo C la capacità di canale. Mentre una soluzione basata su segnalazione ortogonale (ad esempio, l'FSK del § 16.5.1) determina un aumento asintoticamente *esponenziale* della banda occupata[18], le tecniche illustrate nel seguito consentono di mantenere la relazione tra grado di protezione e banda occupata di tipo *proporzionale*.

Classificazione delle tecniche di codifica di canale Una categoria molto vasta è quella dei codici *a blocco*, che operano suddividendo il messaggio da proteggere in blocchi disgiunti, codificati in modo indipendente; una diversa classe è quella dei codici *convoluzionali*, che invece trattano il messaggio come una sequenza priva di suddivisioni, da cui *calcolare* una nuova sequenza (a velocità maggiore) che ne rappresenta la codifica. Mentre i codici convoluzionali sono descritti al § 17.4.2, la trattazione dei codici a blocco è iniziata al § 15.6.2.1, che si suggerisce di consultare prima di proseguire, riassumendone qui solo alcuni concetti.

[18] Infatti partendo dall'espressione della banda occupata dall'FSK $B \to \frac{f_b}{2} \cdot \frac{L}{\log_2 L}$ (eq. (16.21)) e considerando che $L = 2^M = 2^{f_b T}$ si ottiene $B = \frac{1}{2T} \cdot 2^{f_b T}$ ovvero un aumento esponenziale di B al crescere di T. Un diverso esempio può essere l'uso di forme d'onda ortogonali realizzate come $rect_{T/L}$ posti all'interno del periodo di simbolo T in modo che non si sovrappongano se associati a simboli diversi (vedi fig. 7.8 a pag. 218). Anche qui, aumentando T il numero di simboli $L = 2^M = 2^{f_b T}$ aumenta esponenzialmente, e la durata $T/L = T/2^{f_b T}$ di ogni *rect* tende esponenzialmente a zero se $T \to \infty$, mentre la banda occupata tende ad infinito, sempre con legge esponenziale rispetto a T.

Tasso di codifica, velocità binaria ed E_b/N_0 Riprendiamo la notazione introdotta al § 15.6.2.1 per i *codici a blocco*, in cui ad ogni k bit della sequenza di ingresso (da proteggere)

si genera una *codeword*[19] con lunghezza $n = k + q > k$ bit, in cui sono stati *aggiunti* q bit *di protezione* in funzione dei k del blocco: tale procedura viene indicata come *codice* (n, k), la cui efficienza è misurata dal *tasso di codifica* (o CODE RATE)



$$R_c = \frac{k}{n} < 1$$

che rappresenta la frazione di bit informativi sul totale di quelli trasmessi, nonché *l'inverso* del fattore di *espansione di banda*[20]: la nuova velocità di trasmissione in presenza di codifica vale infatti

$$f_b' = \frac{f_b}{R_c}$$

Osserviamo che all'aumento della velocità di segnalazione (essendo $f_b' > f_b$) corrisponde una eguale diminuzione del rapporto $E_b/N_0 = \frac{\mathcal{P}_x}{N_0 f_b'} = R_c \frac{\mathcal{P}_x}{N_0 f_b}$, e conseguentemente si assiste ad un *peggioramento* della probabilità di errore *grezza* del decisore: pertanto, la capacità correttiva del codice deve essere tale da compensare anche questo aspetto. Per mantenere limitato l'effetto descritto, così come l'espansione di banda, vorremmo trovare codificatori per cui R_c sia il più possibile vicino ad uno.

Distanza di Hamming, distanza minima e capacità correttiva Al § 15.6.2.1 è stata definita $d_H(x_i, x_j)$ come il numero di bit in cui le codeword x_i e x_j differiscono, pari al numero di errori sul bit necessario affinché una si trasformi nell'altra. Valutando la distanza di Hamming d_H tra tutte le possibili coppie di codeword, si definisce la *distanza minima del codice* $d_m = \min_{i \neq j} d_H(x_i, x_j)$, che descrive quanto le codeword siano *vicine* nel caso peggiore.

Esempio Con $k = 4$ e $q = 3$ esistono $2^4 = 16$ codeword su $2^7 = 128$ possibili configurazioni degli $n = k + q$ bit della codeword. Ciò significa che ci sono $2^3 = 8$ *non*-codeword per ogni codeword. Sarebbe bene scegliere queste ultime in modo che risultino *distanti* (nel senso di Hamming) l'una dall'altra!

Come discusso al § 15.6.2.1, la capacità di correzione del codice è direttamente legata alla *minima distanza* d_m, sussistendo le relazioni

- per rivelare l (o meno) errori per codeword occorre $d_m \geq l + 1$
- per correggere t (o meno) errori per codeword occorre $d_m \geq 2t + 1$

Un codice è tanto più *potente* quanti più errori è in grado di correggere, e dunque deve possedere d_m elevato. In un codice a blocchi (n, k) i k bit del messaggio originale

[19] Si ricorda che l'insieme delle 2^k codeword costituisce un *codebook*.

[20] Notiamo la differenza tra queste tre grandezze dall'aspetto simile: l'efficienza spettrale $\rho = f_b/B$ indica quanto una tecnica di modulazione numerica faccia buon uso dello spettro, il rapporto $\frac{B}{R}$ esprime l'inverso del grado di utilizzo della banda B di un canale numerico, mentre $R_c = k/n$ misura l'efficienza del codice adottato.

assumono tutte le configurazioni possibili, e quindi contribuiscono alla distanza tra codeword per un solo bit; per ottenere $d_m > 1$ occorre pertanto sfruttare gli $n - k = q$ bit di protezione, portando a scrivere

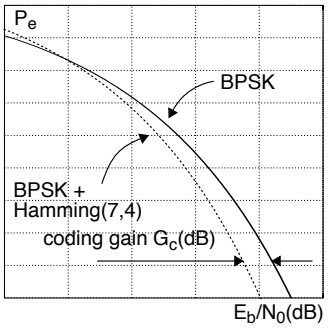
$$d_m \leq q + 1 = n - k + 1$$

che evidenzia la relazione tra d_m e la quantità di bit aggiunti q. L'uguaglianza sussiste solo per una particolare classe di codici[21] tra cui il codice a ripetizione, discusso al § 15.6.2.2, che adottando una dimensione di blocco in ingresso $k = 1$ ha però un tasso di codifica $R_c = k/n = 1/n$ molto inefficiente.

Guadagno di codifica G_c La distanza d_m ed il tasso R_c non possono essere scelti indipendentemente, dato che per aumentare d_m occorre aumentare i bit di protezione q, a cui corrisponde una diminuzione del valore di R_c. Una quantità che tiene conto di entrambi i termini è il *guadagno di codifica asintotico*

$$G_c^a = d_m R_c \tag{17.22}$$

che esprime il fattore di aumento della potenza di segnale che avrebbe prodotto lo stesso miglioramento in termini di P_e dovuto all'adozione del codice.



Esempio Se $G_c = 2$ significa che l'uso del codice porta ad un valore di P_e pari a quello ottenibile con una trasmissione a potenza doppia, ma non codificata.

Tale risultato è valido nel caso di *soft-decoding*[22] (pag. 581), ed è aggettivato come *asintotico* in quanto è valido solo per valori di E_b/N_0 elevati: infatti all'aumentare della potenza di rumore, per qualunque codice e metodo di decisione il valore di G_c si riduce, fino a divenire inferiore ad uno, quando gli errori sono così numerosi da non poter più essere corretti.

Mostriamo ora soluzioni che consentono di ottenere un adeguato potere di correzione, senza per questo aumentare di molto la velocità di trasmissione del flusso codificato.

17.4.1 Codifica a blocco

Le proprietà di questa classe di codici possono essere meglio analizzate interpretando l'insieme delle possibili codeword da un punto di vista algebrico, ed adottando una notazione matriciale idonea a descrivere la classe di codici *a blocco*, mentre per la sottoclasse dei codici *ciclici* (§ 17.4.1.2) interviene una notazione polinomiale.

Iniziamo ricordando che il *codebook* (§ 15.6.2.1) di un codice a blocco (n, k) è composto da sequenze di n bit indicate come *codeword* $\mathbf{x}$ espresse mediante un *vettore* ad elementi binari

$$\mathbf{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)$$

[21] Indicati come codici MDS, vedi `http://en.wikipedia.org/wiki/Singleton_bound#MDS_codes`.

[22] Nel caso di decisioni *hard* o bit a bit, si ottiene una espressione del tipo $G_{c,hard}^a = R_c\,(t+1)$, in cui t è il numero di bit per parola che il codice è in grado di correggere.

che può assumere solo 2^k diversi valori tra i 2^n possibili, mentre le ricezione di una delle rimanenti $2^n - 2^k$ combinazioni di bit segnala la presenza di almeno un errore. Ad esempio per un codice a ripetizione 3:1 (§ 15.6.2.2, in cui $k = 1$ ed $n = 3$) vi sono solo due codeword con vettori $x_1x_2x_3$ pari a 000 ed 111, mentre le restanti 8-2=6 configurazioni *non sono* codeword.

Codice lineare Un codebook (che implementa un codice) è detto *lineare* se le sue 2^k codeword costituiscono uno *spazio lineare* (§ 2.4.2), ovvero se comprendono la codeword nulla, e la somma di due codeword è anch'essa una parola di codice. La somma tra due codeword è definita in base alla matematica binaria *modulo due*, ovvero espressa mediante l'operatore di OR *esclusivo* bit a bit $\oplus$ come

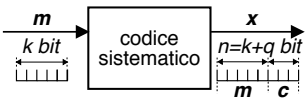
$$\mathbf{x} + \mathbf{y} = (x_1 \oplus y_1 \quad x_2 \oplus y_2 \quad \cdots \quad x_n \oplus y_n) \tag{17.23}$$

Notiamo incidentalmente che, in virtù dell'algebra modulo 2 indotta dall'operatore $\oplus$, la somma tra vettori binari produce un nuovo vettore con elementi pari ad uno nelle posizioni in cui essi differiscono, dunque in numero pari alla d_H tra i vettori.

Distanza d_m per codici lineari Definiamo ora *peso* $w(\mathbf{z})$ di una codeword $\mathbf{z}$ il numero di *uni* in essa contenuti, ovvero la sua distanza di Hamming rispetto alla CW nulla, cioè $w(\mathbf{z}) = d_H(\mathbf{z}, \mathbf{0})$. Per un codice lineare la minima distanza del codice d_m può essere valutata come il *minimo peso* tra tutte le codeword non zero[23], ossia

$$d_m = \min_{\mathbf{z} \neq 0} [w(\mathbf{z})]$$

Codice sistematico e rappresentazione matriciale Con il termine *sistematico* si intende un codice che ottiene gli n bit delle codeword $\mathbf{x}$ concatenando per primi i k bit da proteggere, indicati con $\mathbf{m}$, a cui seguono i $q = n - k$ bit di protezione, indicati con $\mathbf{c}$. Ovvero come abbiamo implicitamente assunto fino ad ora, anche se si tratta di una scelta per nulla scontata. Mostreremo più sotto che un codice sistematico è anche lineare; le sue codeword vengono quindi scritte nella forma



$$\mathbf{x} = \begin{pmatrix} m_1 & m_2 & \cdots & m_k & c_1 & c_2 & \cdots & c_q \end{pmatrix}$$

ovvero come un vettore riga partizionato $\mathbf{x} = (\mathbf{m} \mid \mathbf{c})$, in modo da poterlo calcolare a partire dal vettore $\mathbf{m}$ dei bit da proteggere moltiplicando lo stesso per una *matrice generatrice*[24] $k \times n$ con struttura generale $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$, in cui $\mathbf{I}_k$ è una matrice identità

[23] Infatti dalla definizione di somma tra CW otteniamo che $d_H(\mathbf{x}, \mathbf{y})$ è pari al peso $w(\mathbf{z})$ della codeword $\mathbf{z} = \mathbf{x} + \mathbf{y}$, ossia $\mathbf{z}$ presenta componenti $z_j = 1$ solo in corrispondenza di elementi $x_j \neq y_j$. Ma per la linearità anche $\mathbf{z}$ appartiene al codebook, ovvero sommando tra loro tutte le possibili coppie ottengo l'intero codebook, e dunque la ricerca su tutte le coppie si trasforma in una ricerca su tutte le codeword.

[24] Questa *fantomatica* matrice *generatrice* che cala dall'alto in realtà ha una genesi ben razionale.

Se infatti definiamo una base ortogonale $\{\mathbf{u}_i\}$ per lo spazio $k-$dimensionale descritto da tutti i possibili vettori $\mathbf{m}$ come i k vettori con componenti tutte nulle tranne quella in posizione $i-$esima e pari ad 1, possiamo allora scrivere un generico vettore $\mathbf{m}$ con componenti binarie m_i come una combinazione lineare di vettori della base, ovvero $\mathbf{m} = \sum_{i=1}^{k} m_i \mathbf{u}_i$.

Indicando ora con $\mathbf{g}_i$ la codeword (di n elementi) associata a ciascun vettore $\mathbf{u}_i$, otteniamo che ad un

$k \times k$ e $\mathbf{P}$ è una sotto-matrice di elementi binari $k \times q$. Le codeword si ottengono quindi come $\mathbf{x} = \mathbf{m} \cdot \mathbf{G}$, ovvero

$$\begin{bmatrix} m_1 \cdots m_k & c_1 \cdots c_q \end{bmatrix} = \begin{bmatrix} m_1 \cdots m_k \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & 0 & p_{11} & \cdots & p_{1q} \\ \vdots & 1 & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & p_{k1} & \cdots & p_{kq} \end{bmatrix} \tag{17.24}$$

in modo che $\mathbf{P}$ produca i q bit di protezione come $\mathbf{c} = \mathbf{m} \cdot \mathbf{P}$, in cui sono valide le normali regole di moltiplicazione tra matrici, tranne per l'accortezza di usare la *somma modulo due* anziché quella convenzionale. Il valore della (generica) j-esima ($j = 1, 2, \cdots, q$) componente di $\mathbf{c}$ si calcola pertanto come

$$c_j = m_1 \cdot p_{1j} \oplus m_2 \cdot p_{2j} \oplus \cdots \oplus m_k \cdot p_{kj}$$

in cui il prodotto tra cifre binarie equivale all'operatore logico di AND.

Esempio Se poniamo $\boldsymbol{m} = (0\ 1\ 1\ 0\ 1)$ e $\boldsymbol{p} = (1\ 1\ 0\ 1\ 1)$ il relativo prodotto interno vale $\boldsymbol{m} \cdot \boldsymbol{p} = 0 \cdot 1 \oplus 1 \cdot 1 \oplus 1 \cdot 0 \oplus 0 \cdot 1 \oplus 1 \cdot 1 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$.

In definitiva ciascuna colonna di $\mathbf{P}$ individua un sotto-insieme di elementi di $\mathbf{m}$ su cui calcolare una *somma di parità*, fornendo il motivo per cui questo sotto-blocco di matrice $\mathbf{G}$ è rappresentato dalla lettera $\mathbf{P}$. Ma non è ancora stato detto nulla che ci possa aiutare a scegliere i coefficienti p_{ij} allo scopo di ottenere i valori d_m e R_c desiderati: il *codice di Hamming* (§ 17.4.1.1) ci fornisce una possibile soluzione.

Linearità di un codice sistematico La linearità di un codebook sistematico può essere verificata se riusciamo a mostrare che la somma di due qualunque codeword è ancora una parola del codebook. A tale scopo, dato che la somma modulo due tra vettori binari $\mathbf{x}_1 \oplus \mathbf{x}_2$ avviene bit per bit (eq. (17.23)), consideriamo le due parti $\boldsymbol{m}$ e $\boldsymbol{c}$ di una codeword $\mathbf{x}$ in modo indipendente. Dato che $\boldsymbol{m}$ può assumere una qualunque delle 2^k configurazioni possibili, è sempre vero che $\boldsymbol{m}_3 = \boldsymbol{m}_1 \oplus \boldsymbol{m}_2$ appartiene allo stesso insieme. Per quanto riguarda i q bit di protezione $\boldsymbol{c}$, osserviamo che

$$\boldsymbol{c}_3 = \boldsymbol{c}_1 \oplus \boldsymbol{c}_2 = \boldsymbol{m}_1 \boldsymbol{P} \oplus \boldsymbol{m}_2 \boldsymbol{P} = (\boldsymbol{m}_1 \oplus \boldsymbol{m}_2)\, \boldsymbol{P}$$

e dunque $\boldsymbol{c}_3$ corrisponde alla protezione di $\boldsymbol{m}_3$, ovvero $\mathbf{x}_3 = (\mathbf{m}_3 \mid \mathbf{c}_3)$ è una codeword esistente.

17.4.1.1 Codice di Hamming

E' un codice a blocco (n, k) sistematico e lineare che permette di conseguire un tasso di codifica elevato pur mantenendo un buon potere correttivo. Aggiunge $q \geq 3$ bit di controllo ai k bit informativi per formare codeword di lunghezza complessiva

$$n = 2^q - 1$$

ottenendo un tasso di codifica pari a

$$R_c = \frac{k}{n} = \frac{n-q}{n} = 1 - \frac{q}{2^q - 1}$$

generico vettore $\mathbf{m}$ è associata la codeword $\mathbf{x} = \sum_{i=1}^{k} m_i \mathbf{g}_i = \mathbf{m} \cdot \mathbf{G}$, in cui $\mathbf{G}$ è la nostra *matrice generatrice* di dimensione $k \times n$ le cui righe sono pari alle codeword $\mathbf{g}_i$ con $i = 1, 2, \cdots, k$ associate ai vettori della base $\mathbf{u}_i$.

che aumenta con il crescere di q, come mostrato in tabella. Le sue codeword si individuano ponendo *le k righe* della sottomatrice $\mathbf{P}$ pari a tutte le parole di q bit con *due o più* uni, in qualsiasi ordine. Ma la cosa *ancora più simpatica* è che per un codebook siffatto si ottiene una distanza di Hamming minima $d_m = 3$, indipendentemente dalla scelta di q.

q	n	k	R_c
3	7	4	0.57
4	15	11	0.73
5	31	26	0.84
6	63	57	0.9
7	127	120	0.94

Esempio: codice di Hamming $(7, 4)$. Corrisponde al caso più semplice di scegliere $q = 3$, e dunque $n = 2^3 - 1 = 7$ e $k = 7 - 3 = 4$. Una possibile matrice generatrice è pari a

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

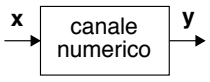
a cui corrispondono le seguenti $2^4 = 16$ codeword, per ognuna delle quali si è evidenziato il peso $w(\mathbf{x})$, confermando che $d_m = 3$.

$\mathbf{m}$	$\mathbf{c}$	$w(\mathbf{x})$
0 0 0 0	0 0 0	0
0 0 0 1	0 1 1	3
0 0 1 0	1 1 0	3
0 0 1 1	1 0 1	4
0 1 0 0	1 1 1	4
0 1 0 1	1 0 0	3
0 1 1 0	0 0 1	3
0 1 1 1	0 1 0	4

$\mathbf{m}$	$\mathbf{c}$	$w(\mathbf{x})$
1 0 0 0	1 0 1	3
1 0 0 1	1 1 0	4
1 0 1 0	0 1 1	4
1 0 1 1	0 0 0	3
1 1 0 0	0 1 0	3
1 1 0 1	0 0 1	4
1 1 1 0	1 0 0	4
1 1 1 1	1 1 1	7

Dato che sia $\mathbf{m}$ che $\mathbf{c}$ assumono tutte le configurazioni possibili, è verificata la linearità del codice (pag. 566), ossia che la somma di una qualunque coppia di codeword corrisponde ad una terza codeword. Notiamo infine che ciascuna della $2^q = 8$ triplette di protezione $\mathbf{c}$ viene usata nel codebook per due volte, ma associata a coppie di sequenze $\mathbf{m}$ da proteggere con valori di distanza di Hamming almeno pari a tre.

Correzione basata sulla distanza Indichiamo ora con $\mathbf{y}$ la parola di codice ricevuta; in presenza di errori, risulta $\mathbf{y} \neq \mathbf{x}$. Il metodo *diretto* per rivelare ed eventualmente correggere gli errori presenti è quello di confrontare gli n bit ricevuti con tutte le possibili 2^k codeword, e se nessuna di queste risulta uguale ad $\mathbf{y}$, scegliere la $\hat{\mathbf{x}}$ con la minima distanza di Hamming, ossia quella per la quale il peso $w(\mathbf{y} \oplus \hat{\mathbf{x}})$ è minimo, ovvero



$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \{w(\mathbf{y} \oplus \mathbf{x})\}$$

Correzione basata sulla sindrome Un metodo che non richiede una ricerca esaustiva si basa invece sul calcolo della cosiddetta *sindrome*, ottenuta mediante moltiplicazione del vettore $\mathbf{y}$ ricevuto per una matrice $n \times q$ di *controllo parità* $\mathbf{H}$, definita come

$\mathbf{H} = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_q \end{bmatrix}$ in cui $\mathbf{P}$ è la stessa matrice di parità utilizzata nella matrice generatrice $\mathbf{G}$, e $\mathbf{I}_q$ è una matrice identità di dimensioni $q \times q$. La matrice $\mathbf{H}$ esibisce la simpatica proprietà[25] che, se moltiplicata per una qualunque codeword valida, fornisce un vettore *nullo* di dimensione q, ossia

$$\mathbf{x} \cdot \mathbf{H} = (0 \quad 0 \quad \cdots \quad 0) \tag{17.25}$$

Al contrario, se moltiplicata per un vettore $\mathbf{y}$ non appartenente al codebook, fornisce un vettore detto *sindrome* $\mathbf{s} = \mathbf{y} \cdot \mathbf{H}$ *non nullo*, e quindi il suo calcolo permette la *rivelazione* (nei limiti consentiti da d_m) dell'occorrenza di errori.

Esempio considerando di nuovo il caso di $q = 3$, la corrispondente matrice di controllo parità $\mathbf{H} = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_q \end{bmatrix}$ è mostrata a lato. E' facile verificare che per tutte le possibili codeword $\mathbf{x}$ (ad es. $\mathbf{x} = [0100111]$) si ottiene $\mathbf{x} \cdot \mathbf{H} = [000]$. Poniamo ora che si verifichi una sequenza di errore $\mathbf{e} = [0010010]$, dando luogo alla ricezione della parola $\mathbf{y} = \mathbf{x} \oplus \mathbf{e} = [0110101]$: per essa si ottiene una sindrome $\mathbf{s} = \mathbf{y} \cdot \mathbf{H} = [100]$.

$$\mathbf{H} = \left[\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

Per quanto riguarda la *correzione*, iniziamo scrivendo il vettore ricevuto come $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$, dove $\mathbf{e}$ è un vettore di n bit le cui componenti sono diverse da zero in corrispondenza dei bit errati di $\mathbf{y}$. Il calcolo della sindrome fornisce allora

$$\mathbf{s} = \mathbf{y} \cdot \mathbf{H} = (\mathbf{x} \oplus \mathbf{e}) \cdot \mathbf{H} = \mathbf{x} \cdot \mathbf{H} \oplus \mathbf{e} \cdot \mathbf{H} = \mathbf{e} \cdot \mathbf{H}$$

visto che come espresso dalla (17.25), la sindrome delle codeword è nulla. Ma dato che la sindrome ha dimensione di q elementi, tutte le 2^n possibili sequenze di errore $\mathbf{e}$ danno luogo a sole 2^q diverse sindromi, e quindi la conoscenza della sindrome non consente di risalire direttamente ad $\mathbf{e}$. Osserviamo però che, riprendendo i risultati esposti al § 15.6.1.1, la probabilità $P(m, n)$ che si siano verificati m errori su n bit decresce al crescere di m, e pertanto il vettore $\hat{\mathbf{e}}$ che con maggior probabilità ha prodotto ognuna delle 2^q sindromi $\mathbf{s} \neq 0$, è quello (tra tutti quelli che producono la stessa $\mathbf{s}$) con il minor peso:

$$\hat{\mathbf{e}} = \underset{\mathbf{e} \,:\, \mathbf{e} \cdot \mathbf{H} = \mathbf{s}}{\operatorname{argmin}} \{w(\mathbf{e})\}$$

Osserviamo inoltre che il vettore di errore *più probabile* e con *peso minimo* è quello con *un solo* bit diverso da zero; indichiamo tali n possibili vettori di errore come $\mathbf{e}_i$ ($i = 1, 2, \cdots, n$), qualora solo l'$i-$esimo bit (di n) sia pari ad uno. Accade poi che (per costruzione) la sindrome $\mathbf{s}_i$ associata ad $\mathbf{e}_i$, calcolata come $\mathbf{s}_i = \mathbf{e}_i \cdot \mathbf{H}$, corrisponda esattamente all'$i-$esima riga tra le n righe di $\mathbf{H}$. Pertanto l'indice i della riga che corrisponde alla sindrome calcolata, identifica l'indice del bit che ha subito errore.

Notiamo infine che se si verificano più errori di quanti d_m permetta di correggere, è inutile (anzi dannoso) cercare di eseguire la correzione, perché il numero di errori com-

[25] In effetti la simpatia c'entra ben poco, ed $\mathbf{H}$ è costruita in modo che le sue q colonne siano *ortogonali* a tutte le k righe di $\mathbf{G}$ (oltre che tra loro), individuando così una base di rappresentazione per il *sottospazio* di dimensione 2^q *complemento ortogonale* di quello di dimensione 2^k descritto dalle codeword di lunghezza $n = k + q$. La dimostrazione che ho trovato (per provare che per codici sistematici il risultato è quello mostrato nel testo) contiene un errore, e non la cito.

plessivo può risultare ancora più elevato. Nel caso del codice di Hamming in cui $d_m = 3$ si può correggere un solo errore per codeword, in accordo alla procedura discussa. Se invece **e** contiene *due* errori, la sua moltiplicazione per **H** produce comunque una delle 2^q possibili sindromi, ed il tentativo di correzione produce un vettore $\hat{\mathbf{x}}$ contenente *tre* errori.

Esempio Un gruppo di bit 1001 è protetto dal codice di Hamming di pag. 568 producendo $\mathbf{x} = 1001110$, mentre in ricezione si osserva $\mathbf{y} = 1101110$. Il calcolo della sindrome $\mathbf{s} = \mathbf{y} \cdot \mathbf{H}$ fornisce il risultato $\mathbf{s} = 111$, che corrisponde alla seconda riga di **H**, ovvero al vettore di errore $\hat{\mathbf{e}} = 0100000$, cioè proprio quello che si è verificato. Se invece avvengono *due* errori, e si riceve ad es. $\mathbf{y} = 1111110$, si ottiene $\mathbf{s} = 101$, a cui corrisponde $\hat{\mathbf{e}} = 1000000$, e quindi il calcolo $\hat{\mathbf{x}} = \mathbf{y} \oplus \hat{\mathbf{e}}$ produce ora $\hat{\mathbf{x}} = 0111110$, che appunto contiene tre errori.

Esercizio Un flusso binario con codifica di Hamming $(31, 26)$ è affetto da $P_e = 10^{-4}$. Determinare la probabilità *residua* di errore *sul bit* (pag. 471) dopo decodifica. **Risposta** La presenza di un solo errore nella codeword viene corretta, mentre con due errori la correzione basata sulla sindrome ne sbaglia tre; il caso con più di due errori si considera improbabile, vedi § 15.6.1.1. La prob. di 2 errori su 31 è (eq. (15.27)) pari a $P(2, 31) \simeq \frac{31 \cdot 30}{2} \left(P_e\right)^2 = 4.65 \cdot 10^{-6}$, e l'evento di errore comporta 3 bit errati su 26 decodificati, dunque per la P_e^{bit} residua si ottiene come $\frac{3}{26} \cdot 4.65 \cdot 10^{-6} = 5.36 \cdot 10^{-7}$.

17.4.1.2 Codice ciclico

Anche questo appartiene alla famiglia dei codici lineari a blocco, con la condizione aggiuntiva che se $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ è una codeword lo sono anche tutti i suoi *scorrimenti ciclici*[26], ovvero le codeword

$$\mathbf{x}^{(1)} = (x_2, x_3, \cdots, x_1), \quad \mathbf{x}^{(2)} = (x_3, x_4, \cdots, x_2), \quad \cdots, \quad \mathbf{x}^{(n)} = (x_n, x_1, \cdots, x_{n-1})$$

In tal caso sussistono delle proprietà algebriche aggiuntive che si basano sulla teoria dei *campi di Galois*[27], i cui elementi sono polinomi

$$x(p) = x_1 p^{n-1} + x_2 p^{n-2} + \cdots + x_{n-1} p + x_n$$

nella variabile p, con coefficienti binari definiti a partire dagli elementi delle codeword **x**; per tale motivo, i codici ciclici sono detti anche codici *polinomiali*[28]: senza volerci addentrare nei particolari della teoria[29], citiamo direttamente i principali risultati.

Un codice ciclico (n, k) è completamente definito a partire da un *polinomio generatore*

$$g(p) = p^{n-k} + g_2 p^{n-k-1} + \cdots + g_{n-k} p + 1$$

di grado $n - k$ che deve essere un divisore di $p^n + 1$, ovvero tale che $p^n+1 / g(p) = q(p)$ con resto nullo[30]. Una volta definito $g(p)$ è possibile ottenere la matrice generatrice **G** del codice in forma *sistematica* $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$, calcolando le k righe di **P** come i coefficienti del polinomio *resto* della divisione tra p^i e $g(p)$, con $i = n-1, n-2, \cdots, n-k$.

[26] Ad esempio, il codice $[000, 110, 101, 011]$ è un codice ciclico, mentre $[000, 010, 101, 111]$ no. Notiamo che gli scorrimenti della codeword 000, sono la codeword stessa.

[27] `http://it.wikipedia.org/wiki/Campo_finito`

[28] Si veda `http://en.wikipedia.org/wiki/Polynomial_code`

[29] `http://en.wikipedia.org/wiki/Cyclic_code`

[30] Si applicano le regole di divisione tra polinomi `http://it.wikipedia.org/wiki/Divisione_dei_polinomi`

Esempio Troviamo la matrice generatrice in forma sistematica per un codice ciclico $(7,4)$. Osserviamo innanzitutto che p^n+1 si fattorizza come $p^7+1 = (p+1)\left(p^3+p^2+1\right)\left(p^3+p+1\right)$, e scegliamo $g(p) = p^3+p^2+1$ come polinomio generatore. Il calcolo di p^i/p^3+p^2+1 per $i = 6,5,4,3$ fornisce come resti i polinomi p^2+p, $p+1$, p^2+p+1, p^2+1, pertanto la matrice generatrice risulta $\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$. Osserviamo che, a parte una permutazione, le righe di $\mathbf{P}$ sono le stesse di quelle che definiscono il codice di Hamming di pag. 568: infatti, Hamming rientra anche nella classe dei codici ciclici.

D'altra parte nel caso di un codice ciclico le codeword $\mathbf{x}$ associate ai bit $\mathbf{m}$ da proteggere si possono ottenere non solo utilizzando $\mathbf{G}$ come descritto dalla (17.24), ma anche in base alla seguente proprietà: indicando con $m(p) = m_1 p^{k-1} + m_2 p^{k-2} + \cdots + m_{k-1} p + m_k$ il polinomio associato ai k bit da codificare, il polinomio associato alla corrispondente codeword $\mathbf{x}$ si ottiene come *il prodotto* $x(p) = m(p) \cdot g(p)$, e quindi gli elementi x_i della codeword sono individuati calcolando la *convoluzione discreta*[31] tra i coefficienti di $m(p)$ e $g(p)$: $x_i = \sum_{j=1}^{k} m_j g_{i-j+1}$ per $i = 1, 2, \cdots, n$, operazione che può essere realizzata mediante un filtro FIR con coefficienti dati dai valori g_i.

Notiamo infine che nel testo si è già incontrato un codice polinomiale (e quindi ciclico) al § 15.6.3.3 a proposito del CRC: i q bit di protezione possono quindi essere anche ottenuti mediante l'utilizzo di un registro a scorrimento controreazionato[32], e lo stesso può essere usato anche per il calcolo della sindrome dal lato ricevente.

17.4.1.3 Codice BCH

Prende il nome dalle iniziali dei rispettivi inventori, e rappresenta una sottoclasse dei codici ciclici, in grado di correggere fino a $t < n$ errori: per ottenere questo risultato occorre scegliere un numero di bit di protezione $q \leq mt$ in cui $m \geq 3$ è un intero, una lunghezza delle codeword pari a $n = 2^m - 1$, ed un polinomio generatore $g(p)$ di grado massimo mt le cui radici sono potenze dell'elemento primitivo α del campo di Galois $GF(2^m)$[33]. In tal caso si ottiene $d_m \geq 2t+1$, e la capacità di correggere fino a t errori. Nel caso in cui $t = 1$, si ricade nel caso dei codici di Hamming. La fase di correzione degli errori può essere realizzata mediante il calcolo di una sindrome, per mezzo di una matrice di controllo $\mathbf{H}$ realizzata a partire dalle potenze dell'elemento primitivo α, o sfruttando nuovamente proprietà algebriche, che eventualmente si traducono in

[31] Il valore dei coefficienti del polinomio prodotto è indicato anche come *prodotto di Cauchy* (vedi `http://it.wikipedia.org/wiki/Prodotto_di_Cauchy`), e che si ottenga come una convoluzione è facilmente verificabile: dati ad es. $a(p) = a_0 + a_1 p + a_2 p^2$ e $b(p) = b_0 + b_1 p$, si ottiene $c(p) = a_0 b_0 + (a_0 b_1 + a_1 b_0) p + a_2 p^2 = \sum_{i=0}^{2} p^i \sum_{j=0}^{2} a_j b_{i-j}$. La notazione lievemente diversa del testo, è dovuta al diverso modo di indicizzare i coefficienti.

[32] Forse ho trovato dove si spiega come faccia un registro a scorrimento controreazionato a svolgere questo genere di compiti: penso che in una prossima edizione sarà interessante aggiungerlo.

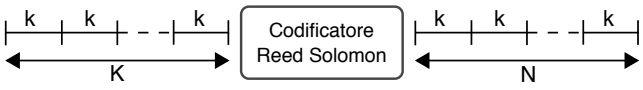
[33] `http://en.wikipedia.org/wiki/BCH_code`

soluzioni circuitali basate su registri a scorrimento controreazionati. Inoltre, è anche possibile applicare l'algoritmo iterativo di *Berlekamp-Massey*[34].

17.4.1.4 Codice di Reed-Solomon

Si tratta di un sottoinsieme dei codici BCH, caratterizzato[35] da parole di codice ad elementi *non binari* ma bensì L-ari, con L che deve essere una *potenza prima*[36]; scegliendo $L = 2^k$ ogni elemento (o simbolo) del codice corrisponde a k bit.

Un codice di *Reed Solomon* (N, K) produce codeword di N simboli $L-$ari ogni K simboli (ognuno di k bit) di informazione prelevati da $\mathbf{m}$; esistono formulazioni in grado di produrre codebook sia di natura sistematica che non. La minima distanza del codice è pari a $d_m = N-K+1$, ed i metodi di decodifica (su cui non ci addentriamo[37]) permettono di correggere fino a $t = \frac{d_m-1}{2} = \frac{N-K}{2}$ simboli $L-$ari per codeword; qualora *la posizione* dei simboli errati sia nota[38] può correggere il doppio dei simboli, ossia $2t$. Dato che opera a livello di simbolo e non di singolo bit, non soffre del problema legato alle sequenze di errore come invece avviene per il codice di Hamming, almeno finché il numero di errori sul bit consecutivi non supera il valore $t \cdot k$, ovvero non coinvolge più di t simboli.



Esempio Scegliendo $k = 8$, $N = 2^k - 1 = 255$, $t = 16$ e $K = N - 2t = 223$ definiamo il codice $RS\,(255, 223)$ che protegge $K \cdot k = 1784$ bit inserendoli in codeword di $N \cdot k = 2040$ bit, di cui $2t \cdot k = 256$ di ridondanza. Tale codice è in grado di correggere fino a 16 simboli ad 8 bit: qualora gli errori sul bit avvengano tutti in simboli differenti si ha il caso *peggiore*, in cui sono correggibili solamente 16 bit su 2040; viceversa nel caso *migliore* in cui gli errori sono consecutivi ed iniziano all'inizio di un simbolo, ne corregge fino a $t \cdot k = 128$. Il tasso di codifica risulta $R_c = {}^{K}/_{N} = 0.937$.

Codice accorciato Può accadere che i bit che costituiscono il flusso f_b di dati da proteggere non sia *omogeneo*, ma presenti strutture sintattiche e quindi delimitazioni che si desidera riflettere nel flusso codificato[39]; oppure, è il sistema di trasmissione ad imporre strutture di trama a dimensione fissa, e non si desidera frammentare una singola codeword a cavallo di time-slot differenti. In tali casi si omette la codifica (e la trasmissione) di alcuni dei K simboli di informazione, idealmente posti a zero, riducendo così la dimensione N della codeword pur mantenendo la stessa quantità di ridondanza $N - K$, accettando di ridurre il tasso di codifica R_c.

[34] `http://en.wikipedia.org/wiki/Berlekamp-Massey_algorithm`

[35] `http://en.wikipedia.org/wiki/Reed-Solomon_error_correction`

[36] Ovvero deve essere della forma $L = \alpha^k$ con α numero primo e k intero positivo. Ciò è necessario affinché gli L simboli corrispondano agli elementi di un campo di Galois $GF\,(L)$.

[37] Ma si veda ad es. `http://scienze-como.uninsubria.it/previtali/Bellini-TeoriaInfoCodiciNote.pdf`

[38] Come nel caso dei codici a cancellazione, vedi ad es. `https://en.wikipedia.org/wiki/Erasure_code`

[39] Come ad esempio avviene nella codifica di sorgenti multimediali (cap. 10)

Esempio Il codice accorciato $RS\,(204, 188)$[40] viene ricavato dal $RS\,(255, 239)$ (con 16 byte di ridondanza) ponendo a zero (e non trasmettendo) 51 dei 239 byte di informazione. Il tasso di codifica passa da 0.937 a 0.921.

In questi casi la codeword viene calcolata, in formato sistematico, a partire da una sequenza informativa fittizia che contiene anche i simboli posti a zero e poi non trasmessi; qualora in fase di decodifica questi risultino diversi da zero[41], la codeword viene segnalata come errata agli stadi di elaborazione successivi.

17.4.1.5 Codifica concatenata

Come abbiamo fatto notare i codici Reed Solomon riescono a correggere efficacemente errori ripetuti, al contrario di quelli di Hamming, che invece hanno un buon comportamento in presenza di errori singoli. E dato che l'unione fa la forza, è possibile usarli assieme adottando lo schema mostrato in figura 17.3, in cui la prima codifica (di *RS*) è detta *esterna* perché più lontana dal canale, mentre per converso la seconda (di Hamming) è detta *interna*.

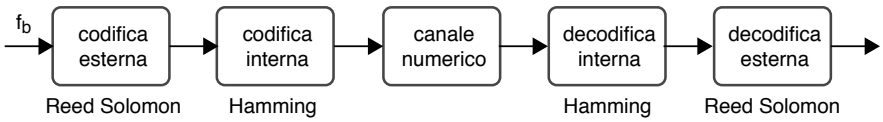
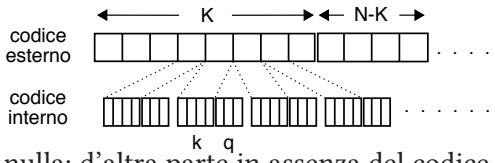


Figura 17.3: Schema di codifica concatenata

Ad ogni simbolo di k bit della codeword esterna vengono aggiunti q bit di ridondanza da parte del codice interno, come mostrato a lato. Gli errori *isolati* introdotti da parte del canale sono corretti dal decodificatore interno, e se distanti tra loro per più di $k+q$ bit, il codice esterno neanche si accorge di nulla; d'altra parte in assenza del codice interno, il solo codice di *RS* non sarebbe stato in grado di correggere più di $\frac{N-K}{2}$ errori per codeword, qualora avvenuti ognuno in un simbolo diverso. Al contrario, in presenza di errori ravvicinati è il codice interno di Hamming a non poter fare nulla, anzi ne introduce di ulteriori, ma nello stesso simbolo: in tal caso il codice esterno di *RS* trova gli errori tutti concentrati in pochi simboli, e provvede alla loro correzione.



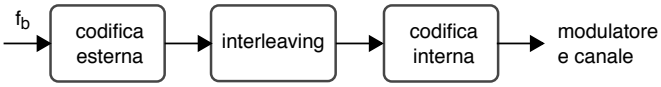
Codice prodotto E' un nome alternativo dato a questo schema di funzionamento[42], e riflette il fatto che il tasso di codifica R_c complessivo è il prodotto dei tassi dei due stadi, dando luogo ad una velocità binaria in ingresso al canale pari a $f_b' = \frac{f_b}{R_c^i \cdot R_c^e}$.

Interleaving Qualora si manifestino sequenze di errore più lunghe di $\frac{N-K}{2} \cdot k$ bit, il numero di simboli del codice esterno che risultano errati diviene maggiore di quanto

[40] Il valore di 188 byte deriva dalla dimensione di una cella ATM (48 byte di dati e 5 di intestazione, § 23.2): infatti $48 \cdot 4 = 192$, ed una codeword si suddivide su 4 celle.

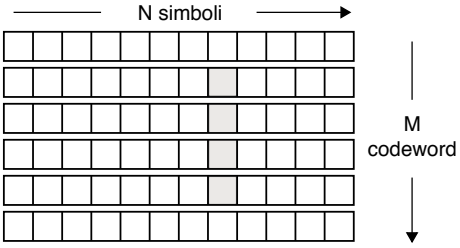
[41] Come osservato a pag. 569 in presenza di un numero di errori superiori al massimo correggibile, se ne verificano ancora di più.

[42] O meglio, ad una versione dello schema in cui la ridondanza interna viene calcolata *sulla totalità* dei simboli nella stessa posizione di M codeword esterne, come avviene per la tecnica esposta al § 15.6.3.2.



esso non possa correggere, e lo schema precedente non funziona più. Per evitare questa situazione, tra i due stadi di codifica (e di decodifica) si frappone un blocco di *interleaving* (ed il suo inverso) (vedi § 15.6.2.3), come mostrato sopra.

La matrice di interleaving viene scritta per righe, inserendovi per intero M codeword del codice esterno, ed è letta per colonne dalla codifica interna, che vi aggiunge ulteriore ridondanza; dal lato ricevente la stessa matrice viene scritta *per colonne* con l'uscita della decodifica interna, e letta *per righe* da parte della decodifica esterna.

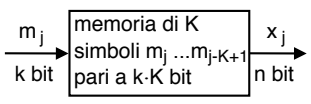


Le aree grigie in figura rappresentano una lunga sequenza di errore, che il codice interno non riesce a correggere: finché M (detto *fattore di interleaving*) è maggiore della più lunga sequenza di simboli errati previsti, questi ultimi vanno a finire in codeword (esterne) differenti, e possono dunque essere ancora corretti. Mentre in assenza di interleaver, i simboli errati sarebbero potuti finire tutti dentro la stessa codeword esterna.

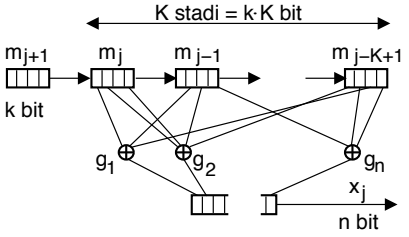
Resta una ultima *sventura* possibile, ovvero la presenza di un disturbo *periodico* che si presenta alla stessa cadenza (o con un suo multiplo) con cui sono scritte le colonne da parte della decodifica interna. Ciò determina che tutti i simboli di una medesima codeword esterna siano errati, rendendone impossibile la correzione. Per rimediare anche a questa evenienza occorre adottare un metodo di interlevaing detto *convoluzionale*, il cui funzionamento non viene approfondito[43], precisando solamente che quello discusso sopra è invece detto interleaver *a blocco*.

17.4.2 Codifica convoluzionale

A differenza della codifica a blocco, questa tecnica produce una sequenza binaria i cui valori dipendono da gruppi di bit di ingresso *temporalmente sovrapposti*, in analogia formale a quanto avviene con l'integrale di convoluzione, che calcola valori di uscita che dipendono da *intervalli* di quelli in ingresso, pesati dai valori della risposta impulsiva. Un generico codice convoluzionale è indicato con la notazione $CC\,(n, k, K)$, che lo descrive come capace di generare gruppi di n bit di uscita (sequenza $\{x\}$) in base alla conoscenza di K simboli di ingresso (sequenza $\{m\}$), ognuno composto da k bit.



Lo schema strutturale del codificatore, raffigurato a lato, ospita i K simboli della sequenza di ingresso (di k bit ciascuno) in un registro a scorrimento in cui per ogni nuovo simbolo m_j che entra da sinistra, i precedenti scorrono a destra, ed il più "vecchio" viene dimenticato. Ognuno



[43] Ma vedi ad es. https://en.wikipedia.org/wiki/Burst_error-correcting_code#Convolutional_interleaver

degli n bit di uscita $x_j\,(i)$, $i = 1, 2, ..., n$ è calcolato eseguendo una somma modulo 2 tra alcuni dei $k \cdot K$ bit di ingresso, individuati da un vettore *generatore* g_i $(i = 1, 2, ..., n)$ costituito da una parola binaria di $k \cdot K$ bit, zero od uno a seconda se l'i*-esimo* sommatore modulo due sia connesso (o meno) al corrispondente bit della *finestra* di ingresso.

Il numero $L = k \cdot K$ di bit che contribuiscono al calcolo della parola di uscita viene indicato come *lunghezza del vincolo*, mentre la quantità $\nu = (K - 1) \cdot k$ è indicata come *memoria* del codice per il motivo che vedremo presto. Resta poi valido il concetto di *coding rate* $R_c = \frac{k}{n}$ che rappresenta il rapporto tra quanti *nuovi* bit di informazione sono necessari in ingresso per ogni gruppo di n bit in uscita dal codificatore.

Automa e diagramma di transizione Il numero di possibili configurazioni dei bit contenuti nello shift register è finito e pari a $2^L = 2^{K \cdot k}$. In base alla scelta degli n vettori g_i, ad ogni configurazione corrisponde un unico valore dell'uscita, e ciò comporta che lo schema può essere descritto da una *tabella della verità*, ovvero da un *automa* a stati finiti, con annesso diagramma di transizione.

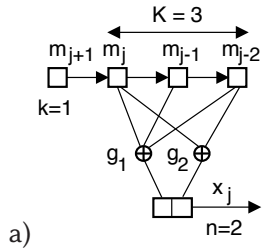
Osserviamo ora che il passaggio da uno stato all'altro non è qualsiasi, ma è stabilito dall'ultimo simbolo in ingresso m_j, e pertanto il *diagramma di transizione* si costruisce individuando i 2^ν *stati* S associati alle possibili combinazioni di bit dei precedenti $K - 1$ simboli di ingresso. Ad ogni stato competono 2^k transizioni, una per ogni possibile m_j di ingresso, diretta verso lo stato individuato dalla nuova configurazione di shift-register che si è determinata. Infine, ad ogni transizione è associato in modo univoco il gruppo di n bit $x_j\left(m_j, S_j\right)$ da emettere in uscita[44]. Ma prima che il discorso diventi troppo confuso, procediamo con un esempio pratico.

Codice CC(2,1,3) Consideriamo il codice convoluzionale il cui schema è mostrato in fig. 17.4-a), che produce due bit di uscita per ogni bit di ingresso in funzione degli ultimi tre, ovvero $CC\,(n, k, K) = CC\,(2, 1, 3)$, caratterizzato da un tasso di codifica $R_c = \frac{k}{n} = \frac{1}{2}$, da una lunghezza del vincolo $L = K \cdot k = 3$, una memoria $\nu = (K - 1) \cdot k = 2$ ed un numero di stati $2^\nu = 4$, da ognuno dei quali si dipartono $2^k = 2$ transizioni. Se scegliamo come vettori generatori le parole[45] $g_1 = (1\ 1\ 1)$ e $g_2 = (1\ 0\ 1)$ si ottiene la tabella della verità di fig. 17.4-b), che mostra i valori di uscita $x_j\,(i)$ ottenuti eseguendo gli XOR descritti dai vettori g_i sulla parola di tre bit costituita dall'ultimo ingresso m_j e dai due precedenti ingressi, indicati come S_j, e che rappresentano appunto *la memoria del passato* all'istante j. Infine la fig. 17.4-c) rappresenta l'automa ed il diagramma di transizione corrispondente[46], in cui le transizioni tra stati sono disegnate con linee tratteggiate o continue a seconda se il valore m_j dell'ultimo ingresso sia pari a zero o ad uno, e sono etichettate con la coppia di bit in uscita x_j riportati nella tabella della verità. Come si può verificare, ogni stato ha *solo due* transizioni, e dunque per ogni

[44] Lo stesso valore di x_j potrebbe essere prodotto da più di una delle $2^{k \cdot K}$ diverse memorie del codificatore.

[45] Il valore di un vettore generatore viene spesso espresso in notazione ottale, dunque nel nostro caso avremmo $g_1 = 7_8$ e $g_2 = 5_8$.

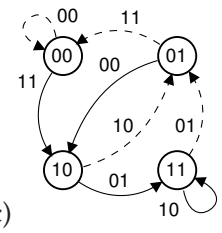
[46] In accordo allo schema `https://it.wikipedia.org/wiki/Macchina_di_Mealy`



a)

b)

m_j	S_j	$x_j(1)$	$x_j(2)$
0	00	0	0
0	01	1	1
0	10	1	0
0	11	0	1
1	00	1	1
1	01	0	0
1	10	0	1
1	11	1	0



c)

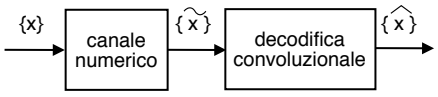
Figura 17.4: a) - architettura del codice convoluzionale $CC(2,1,3)$; b) - tabella della verità; c) - diagramma di transizione; le linee tratteggiate indicano uno zero in ingresso

ingresso sono possibili solo *due valori* dei quattro 4 che sarebbero possibili con due bit; inoltre questi valori differiscono in *entrambi i bit*[47].

Esempio Con una sequenza di ingresso $\{m\} = \{1010\}$ si osserva che, seguendo il diagramma di transizione a partire dallo stato iniziale 00, la sequenza di stati risulta $\{S\} = \{00, 10, 01, 10, 01\}$, mentre quella di uscita è $\{x\} = \{11, 10, 00, 10\}$. E' d'altra parte possibile anche il procedimento inverso, ossia conoscendo $\{x\}$ e quindi $\{S\}$ si può risalire ad $\{m\}$, sempre percorrendo le transizioni etichettate con i simboli x_j. In definitiva, osserviamo come ad ogni coppia di sequenze $(\{m\}, \{x\})$ sia biunivocamente associata una sequenza di stati $\{S\}$.

Diagramma a traliccio Per meglio visualizzare le possibili sequenze di stati si adotta una rappresentazione detta *diagramma a traliccio* (TRELLIS) del codificatore, ottenuta a partire dalla *riorganizzazione* grafica del diagramma di transizione (fig. 17.4-c)) come mostrato a sin. in fig. 17.5-a). Il traliccio è quello al centro, costituito da *nodi* disposti su tante righe quanti sono gli stati dell'automa, e tante colonne quanti sono gli istanti temporali che vogliamo considerare. I collegamenti tra colonne del traliccio corrispondono alle transizioni dell'automa: ponendo uno stato iniziale $S_{j=0} = 00$, si costruisce la colonna $j = 1$ riportando le due possibili transizioni, tratteggiata o continua a seconda che sia entrato uno zero od un uno, e si etichettano le transizioni con la parola x_j emessa in uscita. Il processo si ripete per tutti gli istanti temporali. In basso in figura è riportata una possibile sequenza codificata $\{x\}$ *trasmessa*, in corrispondenza della quale *l'effettivo* percorso attraversato nel traliccio, e dunque la successione di stati $\{S\}$ associata, è rappresentato dalle linee *blu* e più spesse.

17.4.2.1 Criterio di decodifica



Come già osservato non tutte le sequenze di stati (e di uscite) sono ammissibili; indichiamo con $\mathcal{X}$ il loro insieme. Consideriamo quindi la sequenza $\{\tilde{x}\}$ osservata all'uscita di un canale rumoroso, e contenente *errori*. In virtù dei vincoli, nel caso di errori sufficientemente isolati non esiste una sequenza di stati

[47] La dipendenza di x_j da (m_j, S_j) è legata alla scelta dei generatori g_i. Nel caso in cui un valore $x_j(i)$ sia sempre uguale ad uno dei k bit di m_j, il codice diviene *sistematico*.

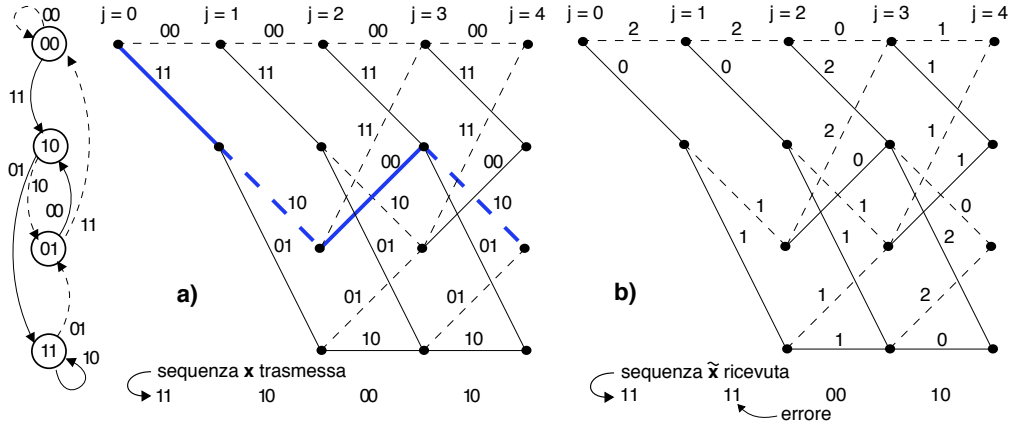


Figura 17.5: a) - diagramma a traliccio e sequenza $\boldsymbol{x}$ trasmessa; b) - costi d_H per la sequenza ricevuta; le linee tratteggiate indicano uno zero in ingresso

$\{S\}$ in grado di produrre *esattamente* la sequenza ricevuta $\{\tilde{x}\}$, e la correzione degli errori consiste nel trovare la sequenza $\{\hat{S}\}$ *ammissibile* e tale che la corrispondente uscita $\{\hat{x}\}$ sia la più *simile* a $\{\tilde{x}\}$. La metrica adottata è la *distanza di Hamming*[48] $d_H(\widehat{x}, \tilde{x})$ tra le due sequenze, ossia il numero di bit diversi tra le due, ed il criterio di decodifica viene definito come

$$\{\hat{x}\} = \arg\min_{\{x\}\in\mathcal{X}} \{d_H(x, \tilde{x})\} \tag{17.26}$$

Decodifica di Viterbi E' il nome della tecnica basata sui principi della programmazione dinamica[49] che permette di risolvere il problema (17.26) senza dover enumerare[50] tutte le sequenze ammissibili $\{x\} \in \mathcal{X}$.

A tale scopo gli archi del traliccio vengono etichettati come in fig. 17.5-b), con le $d_H(x_j, \tilde{x}_j)$ tra il simbolo x_j associato alla transizione e quello $\tilde{x}_j$ ricevuto all'istante j. Per ogni particolare sequenza di stati $\{S\}$ il *costo* $d_H(x, \tilde{x})$ tra la sequenza ricevuta $\tilde{x}$ e quella x associata ad S è pari alla somma delle $d_H(x_j, \tilde{x}_j)$ che etichettano gli archi attraversati, ossia[51]

$$d_H(x, \tilde{x}) = \textstyle\sum_j d_H(x(S_j/S_{j-1}), \tilde{x}_j) \tag{17.27}$$

Il criterio (17.26) è dunque equivalente a quello di trovare il percorso *di minimo costo*[52] per l'attraversamento di un grafo valutato. Ciò avviene esaminando il traliccio

[48] Questo caso viene indicato con il termine *hard-decision decoding* in quanto il ricevitore *ha già* operato una decisione (quantizzazione) rispetto a $\tilde{x}$. Al contrario, se i valori ricevuti sono passati *come sono* al decodificatore di Viterbi, questo può correttamente valutare le probabilità $p(\tilde{x}/\widehat{x})$ ed operare in modalità *soft decoding*, conseguendo prestazioni migliori.

[49] Vedi as es. `https://it.wikipedia.org/wiki/Programmazione_dinamica`

[50] Dato che da ogni stato si dipartono 2^k archi, ad ogni istante j il numero di percorsi alternativi aumenta di un fattore 2^k, crescendo molto velocemente all'aumentare di j.

[51] Ad esempio, con riferimento alla fig. 17.5, la $\{S\} = \{00, 10, 11, 01, 10\}$ ha un *costo* pari a 3.

[52] Qualora la distanza tra $\tilde{x}_j$ ed un possibile x_j sia invece espressa da una probabilità condizionata $p(\tilde{x}_j/x_j)$, il processo di decodifica è detto *di massima verosimiglianza* e la decodifica è detta *soffice* (soft), vedi § 17.4.2.3.

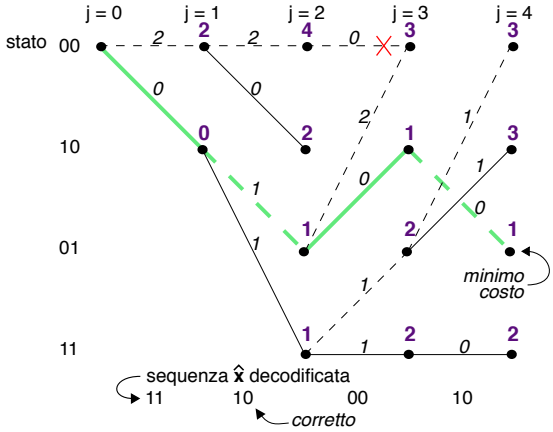


Figura 17.6: Decodifica di Viterbi come percorso di minimo costo

per colonne da sinistra a destra, e valutando per ciascun nodo (riga) il *miglior* costo *parziale*[53] tra i diversi percorsi che lo raggiungono.

Esempio Applichiamo l'algoritmo descritto al caso in questione con l'aiuto della figura 17.6 (ottenuta a partire dalla 17.5-b)) che mostra sopra ad ogni nodo il risultato del calcolo del *costo parziale* del *miglior* percorso che lo raggiunge, ottenuto sommando i costi di attraversamento degli archi (fig. 17.5-b)) che lo compongono, mostrati in corsivo. Ad esempio, lo stato 00 all'istante $j = 3$ potrebbe essere raggiunto tramite il percorso tutto *orizzontale* che rimane nello stato 00, e che assomma un costo parziale di 4. A questo si preferisce il percorso proveniente dallo stato 01, che ha accumulato (per $j = 2$) un costo parziale di 1, a cui sommare $d_H(x(S_{00}/S_{01}), \tilde{x}_3) = d_H(11, 00) = 2$, per un nuovo costo parziale di 3.

All'estremità destra di figura 17.6 viene indicato il percorso di minimo costo, e la corrispondente $d_H(x, \tilde{x})$ minima. In effetti l'algoritmo calcola solamente la d_H minima, e per risalire alla sequenza di stati (e dunque di uscite) che l'ha prodotta, occorre (per ogni nodo del traliccio) memorizzare l'indice del nodo nella colonna precedente da cui parte l'arco che ha determinato il minor costo locale. Una volta individuato (all'ultima colonna) il nodo in cui termina il percorso di minor costo, svolgendo all'indietro la catena dei puntatori si trova la sequenza di stati $\{S\}$ ottima (percorso *verde* a tratto spesso), e da questa in base alla fig. 17.5-a) sia la $\{m\}$ che la $\{\tilde{x}\}$, che come si vede *è quella esatta.*

In fig. 17.7 è riportato un esempio di *pseudo-codice* che implementa l'algoritmo, a partire da tre tabelle $m(p, q)$, $x(p, q)$ e $A(p, q)$, di dimensione $2^\nu \times 2^\nu$, che contengono rispettivamente i valori di ingresso m e di uscita x in corrispondenza ad una transizione da p a q, ed un valore pari a 1 o 0 a seconda se la transizione esiste o meno. Il codice opera su di una sequenza di ingresso $\tilde{\boldsymbol{x}}$ di J elementi, e ne produce una $\boldsymbol{m}$ di uguale lunghezza,

[53] La scelta del miglior percorso parziale è l'applicazione del principio di programmazione dinamica, secondo il quale "quando due percorsi con costi diversi si incontrano in uno stesso nodo, quello di costo *maggiore* sicuramente *non è* la parte iniziale del percorso di minimo costo, e quindi può essere eliminato".

```
TABELLE:
  m(p,q)                          # ingresso m sull'arco da S_p a S_q
  x(p,q)                          # uscita x sull'arco da S_p a S_q
  A(p,q)                          # presenza di transizione da S_p a S_q
INIZIALIZZA:
per tutti gli stati p = 1,2,··· 2^ν
  CJ(p) = 0
ITERAZIONE:
per tutti gli istanti j = 1,2,··· ,J
  per tutti gli stati q = 1,2,··· 2^ν
    CM(q) = CJ(q)                 # miglior costo parziale al tempo j − 1
    CJ(q) = ∞                     # miglior costo fino a (q,j)
    per tutti gli stati p = 1,2,··· ,2^υ
      se A(p,q) == 1              # esiste transizione da p a q
        c = d_H(x(p,q), x̃_j) + CM(p)  # ipotesi da valutare
        se c < CJ(q)
          CJ(q) = c
          BP(q,j) = p             # backpointer al vincitore
DECODIFICA:
best = ∞
per tutti gli stati q = 1,2,··· 2^ν
  se CJ(q) < best
    best = CJ(q)
    w = q                         # indice stato terminale
per tutti gli istanti j = J,··· ,2,1
  p = BP(w,j)                     # miglior predecessore
  emette m(p,w)                   # messaggio m scritto al contrario
  w = p
```

Figura 17.7: Pseudo codice dell'algoritmo di Viterbi

ma temporalmente invertita. Ovviamente si possono pensare implementazioni molto più efficienti, ma l'esempio ha il solo scopo dimostrativo.

Considerazioni

- l'esempio adottato si mostra in grado di correggere un errore pur impiegando un coding rate pari ad $\frac{1}{2}$, migliore (ad es.) di quello $\left(\frac{1}{3}\right)$ del codice a ripetizione;
- la d_H del miglior percorso corrisponde al numero di bit errati (nel caso in cui siano stati corretti) nella $\tilde{x}$ *ricevuta*, il che permette al decodificatore di *stimare* la qualità del collegamento;
- si verifica errore se esiste una $\hat{x} \neq x$ *ammissibile* e tale che $d_H(\widehat{x}, \tilde{x})$ è *minore* di $d_H(x, \tilde{x})$. In tal caso la sequenza erroneamente decodificata $\hat{x}$ contiene errori *a pacchetto*;
- le capacità di correzione del codice migliorano aumentando la d_H tra le possibili sequenze $\{x\}$. La minima distanza d_m tra sequenze codificate è denominata *distanza libera*, e può essere trovata come la d_H tra una $\{x^0\}$ tutta nulla ($\{x^0\} = \{000000000\}$) e quella con il minor numero di uni, che si diparte e ritorna (nel traliccio) dallo/allo stato 00: nell'esempio di fig. 17.5a) si ottiene $d_m = 5$. Il

decodificatore è in grado di correggere fino a $\frac{d_m-1}{2}$ errori che si presentano in un intervallo pari al vincolo del codice L. Il codice del nostro esempio può dunque correggere 2 bit errati su 5;

- la distanza libera d_m aumenta con il rapporto $\frac{K}{k}$, in quanto la matrice di transizione tra stati diviene più sparsa, ed i valori di $\{x\}$ sono più interdipendenti;
- se il miglioramento di cui sopra è ottenuto aumentando K, ciò equivale ad estendere nel tempo la memoria del codificatore, ma senza per questo alterare il tasso di codifica $R_c = \frac{k}{n}$;
- in presenza di un flusso di dati continuo non ha senso attendere un istante finale (che non esiste) per poter individuare il percorso di minimo costo. In tal caso la decodifica parziale avviene con riferimento ad una colonna del traliccio temporalmente abbastanza lontana dall'ultimo istante j di ingresso, ad es. cinque volte la lunghezza del vincolo L, liberando di conseguenza la memoria occupata dai puntatori precedenti;
- dato che ad ogni istante-colonna sono scartati i percorsi che si incontrano con uno migliore, il numero di percorsi *sopravvissuti* è sempre pari al numero di stati 2^ν;
- all'aumentare del numero 2^ν di stati aumenta la complessità e l'occupazione di memoria dell'algoritmo di Viterbi, che può essere sostituito da tecniche di ricerca a fascio (*beam search*) che estendono solo i percorsi parziali più promettenti, ossia con minor costo parziale.

17.4.2.2 Tail biting

Letteralmente traducibile come *mordendo la coda*, è un metodo per rendere un codice convoluzionale simile ad uno *a blocchi*. Il bitstream di ingresso viene suddiviso in segmenti di m bit, gli ultimi L dei quali sono utilizzati (in ordine inverso) per *inizializzare* lo shift-register del codificatore, che quindi inizia ad elaborare, uno alla volta e dall'inizio, i bit del segmento. Al suo termine, ovvero quando sarà entrato l'$m-$esimo bit del segmento, lo stato del codificatore sarà identico a quello con cui ha iniziato: ciò significa che se venisse di nuovo inserito lo stesso segmento (ovvero in forma periodica), si otterrebbe la stessa uscita, che può dunque essere riguardata nel suo insieme come *la codeword* associata al segmento.

Una variante della tecnica (detta *zerotail*), anziché inizializzare il codificatore ne azzera lo stato, ed aggiunge L bit pari a zero in coda al segmento, ottenendo lo stesso risultato, a spese di un peggioramento di R_c. Con la tecnica *zerotail* la decodifica *sa* che il percorso nel traliccio deve iniziare e terminare con lo stato nullo, e dunque al termine della *digestione* della codeword associata al segmento si può iniziare il backtraking dei puntatori da li, oppure segnalare la presenza di eccessivi errori, qualora non sia quello il percorso di minimo costo.

Con l'inizializzazione dello stato, invece, il traliccio diviene periodico (alcuni lo definiscono *circolare*), e l'algoritmo di Viterbi viene fatto lavorare su di una periodiz-

zazione della codeword ricevuta; dopo alcuni periodi si individua lo stato di minimo costo, e recuperati i puntatori (per un periodo-codeword) a partire da quello.

17.4.2.3 Decodifica a decisione soffice

Fino ad ora la decodifica di canale è stata applicata *a valle* del processo di decisione[54], in modo da individuare le codeword come quelle con la minima *distanza di Hamming* rispetto al risultato prodotto dal decisore. Se invece la decodifica di linea non esegue nessuna decisione, ma inoltra interi blocchi di n campioni $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ prelevati dal segnale ricevuto, la decodifica di canale può individuare le codeword trasmesse come quelle con la minima *distanza euclidea* d_E rispetto a quanto ricevuto[55], ottenendo prestazioni migliori a spese di un maggior onere di calcolo. Questa tecnica valuta la distanza come

$$d_E(\mathbf{y}, \mathbf{x}^i) = \sum_{j=1}^{n} (y_j - x_j^i)^2 \tag{17.28}$$

tra i valori (continui) y_j ricevuti e quelli (binari) che si sarebbero dovuti ricevere nell'ipotesi che fosse stata trasmessa la $i-$esima delle possibili codeword $\mathbf{x}^i = (x_1^i, x_2^i, \cdots, x_n^i)$, decidendo quindi per la codeword $\hat{\mathbf{x}}$ che rende $d_E(\mathbf{y}, \hat{\mathbf{x}})$ minima.

Decodifica di massima verosimiglianza Di fatto minimizzare la distanza (17.28) equivale a massimizzare la verosimiglianza logaritmica $\ln(p_n(\mathbf{y}/\mathbf{x}^i))$ dell'ipotesi che sia stata trasmessa la codeword $\mathbf{x}^i$ avendo ricevuto la v.a. $n-$dimensionale $\mathbf{y}$ affetta da rumore gaussiano n di varianza σ^2. Infatti considerando i bit incorrelati[56], $p_n(\mathbf{y}/\mathbf{x}^i)$ è pari al prodotto delle $p_n(y_j/x_j^i)$ dei singoli bit, e $\ln(p_n(\mathbf{y}/\mathbf{x}^i))$ diviene una somma, i cui termini[57]

$$\ln(p_n(y_j/x_j^i)) = -\ln(\sqrt{2\pi}\sigma) - \frac{(y_j - x_j^i)^2}{2\sigma^2} \tag{17.29}$$

sono legati a quelli (cambiati di segno) che compaiono nella (17.28), dato che ne il primo termine di (17.29) ne il denominatore del secondo intervengono nella minimizzazione di (17.28).

Mentre per un codice a blocchi appare problematico valutare la (17.28) per tutte le possibili $\mathbf{x}^i$, le considerazioni ora svolte sono invece particolarmente adatte al contesto della codifica convoluzionale, dato che basta sostituire nella (17.27) la d_H con la d_E per il calcolo dei costi associati ai percorsi nel traliccio esplorato dall'algoritmo di Viterbi. In tal caso si assiste ad un miglioramento di prestazioni (rispetto all'approccio *hard)* equivalente ad un aumento di E_b/N_0 di circa 2 dB.

Prestazioni La figura che segue riporta i valori della P_e^{bit} *residua* per la decodifica di Viterbi di un CC con $R_c = 1/2$ al variare della lunghezza del vincolo L, il cui bitstream viene trasmesso mediante modulazione QPSK, in funzione del rapporto E_b/N_0 in ricezione.

[54] Per questo detta *hard decision decoding* in quanto opera su decisioni già prese.

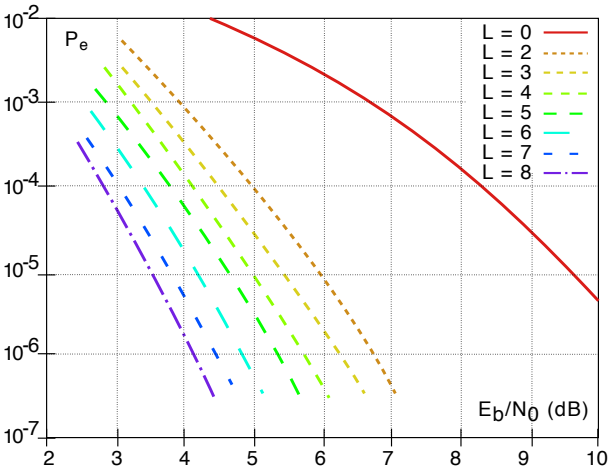
[55] Tale possibilità è indicata come *soft* in quanto richiede operazioni in virgola mobile; si veda ad es. la spiegazione fornita presso
`http://www.gaussianwaves.com/2009/12/hard-and-soft-decision-decoding-2/`.

[56] Anche se la codifica introduce correlazione, l'ipotesi è troppo comoda per poter arrivare al risultato!

[57] Vedere come si è proceduto a pag. 631.

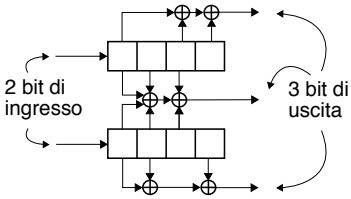
Osserviamo il conseguimento di un guadagno di codifica (pag. 565) che per una $P_e = 10^{-5}$ va da circa 3.5 a 6 dB per le diverse scelte di L, che corrispondono a tralicci da 4 a 256 stati.

Oppure, visto nell'altro senso, per un E_b/N_0 di 5 dB otteniamo una P_e migliore tra 100 e più di 10000 volte rispetto al caso non codificato.

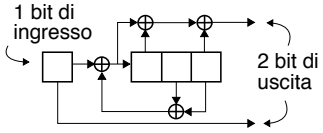


17.4.2.4 Altri schemi di codifica convoluzionale

Lo schema riportato nell'esempio di fig. 17.4 non è l'unico possibile. Anche se permette di variare il tasso di codifica $R_c = \frac{k}{n}$ variando i valori di k ed n, questi impattano anche su complessità del traliccio, lunghezza del vincolo L e distanza libera d_m; per garantirsi la massima libertà di azione sui parametri della codifica, sono state definite anche architetture differenti. Lo schema riportato a lato ad esempio dispone i $k = 2$ bit di ingresso su due diversi rami, ed impiegando tre generatori, consegue un tasso $R_c = \frac{k}{n} = \frac{2}{3}$ con 64 stati, $K = 8$ ed $L = 8$.



Anche se non è stata affrontata la parte di teoria che descrive i *CC* in termini di risposta impulsiva e risposta in frequenza, non può sfuggire la similitudine tra le architetture illustrate e quelle dei filtri FIR, pur se in logica modulo due. Ma esiste anche la possibilità di realizzare un *CC ricorsivo*, in cui cioè sono presenti operazioni XOR *all'indietro* e non solo in avanti, come nello schema mostrato in figura[58] che consegue $R_c = 1/2$, e che è anche *sistematico* in quanto un bit di uscita su due replica quello di ingresso.



17.4.2.5 Codice perforato

Modificare completamente l'architettura del codificatore non sembra la scelta migliore per poter variare R_c, dato che ciò comporta la totale modifica del traliccio su cui è basata la decodifica. Una diversa opzione è quella di *omettere* del tutto la trasmissione di alcuni dei bit presenti nella sequenza codificata, operazione nota come perforazione (*puncturing*) del codice: ad esempio eliminare un bit ogni tre dall'uscita di un *CC* con $R_c = 1/2$ determina un nuovo $R_c = 3/4$, dato che servono ora tre bit di ingresso per produrne quatto di uscita, anziché sei. Ovviamente in questo

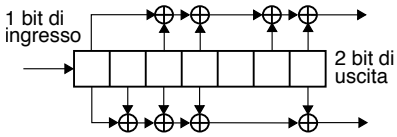
[58] In particolare, lo schema illustrato viene impiegato nella telefonia LTE (ETSI TS 136 212) nell'ambito della codifica *turbo*, vedi § 17.5.1.

caso le capacità correttive peggiorano, ma il processo di decodifica può aver luogo comunque dato che il ricevitore conosce le posizioni non trasmesse, e per esse considera una distanza *neutra* pari a 0.5, come esemplificato alla tabella a lato, dove con δ_i si indica la distanza (hard o soft) tra l'$i-$esimo bit nella sequenza ricostruita, e le corrispondenti ipotesi espresse dal traliccio.

sequenza di ingresso	1	0	1	1	0	0	0
sequenza codificata, $R_c = 0.5$	11	10	00	01	01	11	00
perforazione al tasso $R_c = 0.75$	11	$_1$0	0_0	01	$_0$1	1_1	00
bit trasmessi dal modulatore	11	00		01	11		00
ricostruzione per la decodifica	11	x0	0x	01	x1	1x	00
metrica	$\delta_1\delta_2$	½δ_4	δ_5½	$\delta_7\delta_8$	½δ_{10}	δ_{11}½	$\delta_{13}\delta_{14}$

La percentuale di perforatura può essere resa variabile nel corso della trasmissione in funzione del tasso di errore rilevato, in modo da ridurre la ridondanza nel caso di una buona qualità di ricezione, o mantenerla elevata in caso di peggioramento.

Esempio Lo standard di trasmissione DVB utilizza il *CC* con due generatori mostrato in figura, con 64 stati, vincolo di lunghezza L=7, $d_m = 10$ e $R_c = 1/2$. Le uscite di entrambi i rami possono essere perforate in accordo allo schema in tabella, che mostra anche la variazione del tasso di codifica e della distanza minima: la posizione *degli zeri* nella matrice di perforazione indica i bit di cui è omessa la trasmissione.

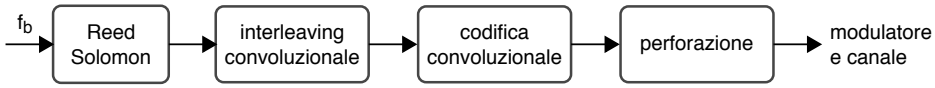


R_c	1/2	2/3	3/4	5/6	7/8
matrice di	1	10	101	10101	1000101
perforazione	1	11	110	11010	1111010
d_m	10	6	5	4	3

17.4.2.6 Concatenazione Solomon-Viterbi

La strategia illustrata al § 17.4.1.5 di collegare in cascata un codice *esterno* in grado di correggere errori a pacchetto (come il Reed-Solomon) con uno *interno* particolarmente idoneo a correggere errori isolati viene impiegata con vantaggio utilizzando come secondo un codice convoluzionale, che come abbiamo illustrato produce errori a pacchetto qualora si eccedano le sue capacità correttive.

E' precisamente la soluzione adottata per la codifica del segnale DVB[59], in cui il *CC* dell'esempio precedente è affiancato (tramite un interleaver convoluzionale di profondità 12) da un codice *RS* accorciato $(214, 188)$ derivato dal $(255, 239)$ (pag. 572) che, con la sua capacità di poter correggere 8 simboli per codeword, porta il tasso di errore *residuo* ad una P_e di 10^{-11} pur in presenza di un E_b/N_0 di 3.2 dB.



17.4.2.7 Viterbi con uscite soffici

Quando la decodifica di canale viene fattorizzata su due blocchi in cascata diviene globalmente vantaggioso che il primo dei due possa alimentare il secondo con valori *continui*, in modo da esprimere *l'affidabilità* della decisione presa.

[59] Vedi le specifiche ufficiali presso
https://www.etsi.org/deliver/etsi_en/300400_300499/300421/01.01.02_60/en_300421v010102p.pdf

Al § 17.4.2.3 abbiamo notato come l'algoritmo di Viterbi alimentato con ingressi soffici di n campioni $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ pervenga ad una decisione di *massima verosimiglianza* a riguardo di una codeword $\hat{\boldsymbol{x}}$ ovvero tale che $\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\mathbf{y}/\mathbf{x})$, da cui risalire al messaggio $\hat{\boldsymbol{m}} = (m_1, m_2, \cdots, m_k)$ associato alla codeword $\hat{\boldsymbol{x}}$. Analizziamo ora una sua *variante* capace di produrre anche una *probabilità a posteriori* $p\left(m_j/\boldsymbol{y}\right)$ per ognuno dei bit m_j $j = 1, 2 \cdots, k$ decodificati, o quantomeno una misura di affidabilità della decisione. Tale variante è denominata SOVA[60] (*soft output Viterbi algorithm*), ed in associazione ad un ingresso soffice, realizza una decodifica detta SISO (*soft input, soft output*).

L'uscita soffice si ottiene associando ad ogni $m_j = \pm 1$ decodificato anche un valore p_j che misura la probabilità che la decisione sia *errata*, in modo che l'affidabilità possa essere espressa come

$$L_j = \ln \frac{1 - p_j}{p_j} \geq 0 \tag{17.30}$$

e l'uscita soffice come $\Lambda_j = m_j L_j$ in cui il segno è la decisione *hard*, e la verosimiglianza logaritmica (17.30) ne esprime il modulo. Vediamo quindi come determinare p_j.

Valutazione della prob. di decisione errata Adottiamo per semplicità un CC sui cui nodi del traliccio convergono due sole transizioni, e prendiamo in considerazione un istante i in cui per ognuno dei 2^ν stati[61] s_i occorre scegliere tra *due* ipotesi di percorso parziale, che indicizziamo con $h = 1, 2$. Indicando con δ un istante del passato per il quale tutti i 2^ν percorsi superstiti condividono con elevata probabilità un medesimo *progenitore* (vedi fig. 17.8) la decisione all'istante i avviene confrontando i costi parziali

$$C_h = \sum_{j=i-\delta}^{i} d_E\left(\mathbf{y}_j, \mathbf{x}_j^{(h)}\right) \quad h = 1, 2 \tag{17.31}$$

in cui $\mathbf{y}_j$ è la parola (soft) di n bit ricevuta all'istante j ed $\mathbf{x}_j^{(h)}$ è la parola (con valori ± 1) emessa allo stesso istante dal codice in corrispondenza del $h-$esimo percorso. Ricordando quanto discusso al § 17.4.2.3, la (17.31) è legata in modo diretto alla verosimiglianza logaritmica $-\ln\left(p\left(\mathbf{y}/\mathbf{x}^{(h)}\right)\right)$ delle ipotesi $\boldsymbol{x}^{(h)}$ avendo osservato $\boldsymbol{y}$; pertanto è

[60] Illustrato per esteso in J.Hagenauer, P.Hoeher, *A Viterbi algorithm with soft-decision outputs and its applications*, 1989 IEEE, a cui si ispira questa parte, e reperibile ad es. presso
`http://shannon.ece.ufl.edu/eel6550/lit/SOVA.pdf`

[61] Si è volutamente mantenuta la notazione introdotta al § 17.4.2.

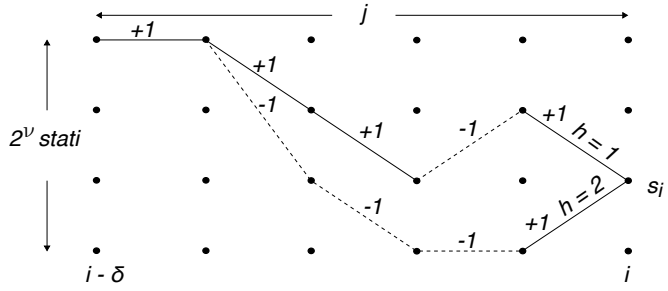


Figura 17.8: Traliccio con cui illustrare il funzionamento di SOVA. Due percorsi convergono su uno stesso stato s_i, e le sequenze informative associate differiscono in due istanti

lecito assumere che

$$Prob\{percorso\ h\} \approx e^{-C_h} \quad h = 1, 2$$

indicando con $\approx$ una qualunque relazione monotona crescente. Assumendo ora $h = 1$ l'indice del miglior percorso e $h = 2$ di quello scartato (e dunque $C_1 < C_2$), la probabilità di aver scelto il percorso *sbagliato* è pari a

$$p_{s_i} = \frac{e^{-C_2}}{e^{-C_1} + e^{-C_2}} = \frac{1}{1 + e^{C_2 - C_1}} = \frac{1}{1 + e^{\Delta}} \tag{17.32}$$

con $\Delta = C_2 - C_1 \geq 0$. Dunque p_{s_i} è pari a 0.5 quando $C_2 = C_1$ mentre tende a zero qualora $C_2 \gg C_1$; in altre parole quanto più i due costi sono simili, tanto più la decisione potrebbe essere errata.

La misura di affidabilità (17.32) ci avverte che, qualora la decisione sia errata, allora (con probabilità p_{s_i}) sono errate anche le decisioni nei q istanti in cui i bit di informazione m nei due percorsi differiscono, ossia

$$m_j^{(1)} \neq m_j^{(2)} \quad j = j_1, \cdots, j_q \tag{17.33}$$

evidenziati in fig. 17.8 da un differente tratteggio, mentre gli istanti j in cui $m_j^{(1)} = m_j^{(2)}$ non sono danneggiati dalla decisione errata. Se abbiamo preventivamente salvato le prob. di errore $\hat{p}_j$ per gli indici espressi dalla (17.33), queste vengono aggiornate come

$$\hat{p}_j = \hat{p}_j \left(1 - p_{s_i}\right) + \left(1 - \hat{p}_j\right) p_{s_i} \quad j = j_1, \cdots, j_q \tag{17.34}$$

ovvero rimangono uguali con prob. $1 - p_{s_i}$, o si aggiornano al nuovo valore p_{s_i} con prob. $1 - \hat{p}_j$ di non aver sbagliato all'istante j.

Differenze rispetto a Viterbi classico In sostanza l'algoritmo ricalca quello in § 17.4.2.1 e produce le medesime decisioni *hard*, ma ognuna corredata dalla verosimiglianza logaritmica $L_j = \ln \frac{1-\hat{p}_j}{\hat{p}_j}$ calcolata a partire dai valori ottenuti tramite la (17.34). Per arrivare a questo risultato occorre memorizzare per ogni istante e per ogni stato, oltre al puntatore al miglior predecessore, anche le differenze Δ tra i costi, da cui ottenere i valori $\hat{p}_j$ (eqq. (17.32) - (17.34)) e L_j (17.30) in fase di decodifica.

Applicazioni Il metodo esposto trova impiego nella demodulazione di segnali con memoria come la modulazione *a traliccio* e CPM (§ 16.10), nella equalizzazione di canali con memoria (§ 18.4.5), e come stadio di decodifica interno di una codifica concatenata (§§ 17.4.1.5 e 17.4.2.6). Quest'ultimo caso si presta particolarmente bene all'utilizzo di una tecnica convoluzionale anche per il codice esterno, operante ovviamente in modalità soft, e separato da quello interno da un adeguato stadio di interleaving. Lo stadio esterno potrà quindi eseguire eseguire una decodifica di massima verosimiglianza a partire dalla sequenza $\Lambda_j = \hat{m}_j L_j$ ($\hat{m}_j = \pm 1$, $L_j \geq 0$) proveniente dallo stadio SOVA interno, decidendo per il messaggio $\boldsymbol{m}^{dec}$ tale che

$$\boldsymbol{m}^{dec} = \arg\max_{\boldsymbol{m}} \left\{ \sum_j m_j \Lambda_j \right\} \tag{17.35}$$

Mentre il caso di un codice esterno a blocchi non permette di risolvere agevolmente la (17.35), consideriamo il caso di un semplice controllo di parità a bit singolo (§ 15.6.3.1), e poniamoci nella condizione che lo stesso rilevi la presenza di un bit errato. Anziché invocare una ritrasmissione, può procedere modificando la decisione per il bit a cui corrisponde la massima probabilità di errore, ovvero a cui corrisponde il valore L_j più piccolo.

17.5 Verso il limite di Shannon

I codici a blocco e convoluzionali (e loro combinazioni) esaminati nella precedente sezione sono utilizzati in innumerevoli sistemi[62] anche grazie ai progressi avvenuti nel frattempo dal punto di vista delle capacità di calcolo e memorizzazione, arrivando a conseguire prestazioni che si discostano di circa 3 dB da quelle *limite* previste dalla teoria di Shannon (§ 17.3.2). Sembrava che non si riuscisse a fare di meglio, quando nei primi anni '90 sono stati definiti i TURBO-CODICI[63] (*Berrou, Glavieux*), e poco dopo rivalutati i codici LDPC[64], inizialmente proposti nel '62 da *Gallager*. Mentre il primo metodo prende le mosse da un nuovo modo di applicare la codifica convoluzionale, il secondo è un codice lineare a blocchi *non sistematico*, con n molto grande (decine di migliaia). In entrambi i casi è determinante l'adozione di una decodifica *soffice*, a cui si affianca la novità dell'adozione di un algoritmo *iterativo*, in modo da arrivare *per gradi* alla decodifica del messaggio. Il risultato è che (sempre grazie all'evoluzione della tecnologia) si è riusciti ad approssimare ancor più da vicino il limite dei -1.6 dB per E_b/N_0 (§ 17.3.4) rispetto al quale mancano solo da 0.7 a 0.5 dB per le due tecniche, determinando la loro adozione da parte dei sistemi più recenti[65]. Vediamo dunque di che si tratta.

17.5.1 Codifica turbo

L'aggettivo *turbo* deve la sua origine al funzionamento iterativo dell'algoritmo di decodifica, che fa uso dei risultati parziali ottenuti al passo precedente[66].

Codifica La versione di turbo codice più studiata è il *codice convoluzionale concatenato parallelo* (PCCC) che consiste nel codificatore *sistematico* mostrato in figura, in cui

[62] In particolare sono stati adottati nell'ambito della telefonia GSM, GPRS, EDGE, e 3G e 3GPP, della diffusione televisiva DVB-S e DVB-T, del WIFI (802.11a-g) e delle missioni spaziali, per non parlare dei supporti di memorizzazione come CD audio, DVD, unità RAID. Per una narrazione di questa evoluzione, oltre che degli argomenti che stiamo trattando, si veda `http://www.crit.rai.it/eletel/LeMiniSerie/MS1.pdf`

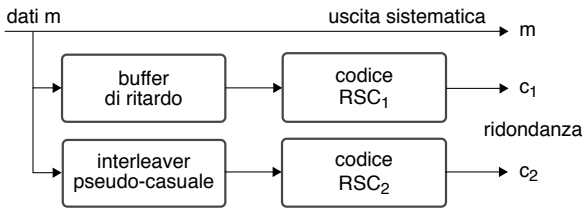
[63] Vedi ad es. `https://en.wikipedia.org/wiki/Turbo_code`

[64] Vedi ad es. `https://en.wikipedia.org/wiki/Low-density_parity-check_code`

[65] Come ad esempio DVB-2, telefonia UMTS ed LTE, 10GBASE-T Ethernet, WIFI 802.11*n* e *ab*, WiMAX 802.16, nonché le missioni spaziali più recenti.

[66] Evidenziamo tra breve la presenza di un vero e proprio percorso di *retroazione*, ma l'aggettivo *turbo* è nato in analogia a quanto avviene in campo automobilistico con i *motori turbo*, una novità tecnica introdotta negli stessi anni in cui è stato definito questo metodo di decodifica.

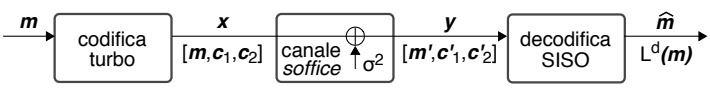
due *CC* ricorrenti (RSC[67]) uguali[68] elaborano il medesimo bitstream $\boldsymbol{m}$ di ingresso producendo i bit di protezione $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$, tranne che il secondo encoder riceve una copia *rimescolata* da parte dell'interleaver, la cui definizione[69] è fondamentale per la riuscita di un turbo codice: grazie ad esso infatti gli ingressi ai due rami divengono istante per istante *incorrelati*, così come le rispettive uscite.



Codificatore turbo parallelo

La fase di codifica suddivide il bitstream di ingresso $\boldsymbol{m}$ in segmenti di dimensione k uguale a quella dell'interleaver, ed applica la tecnica del *tail biting* (§ 17.4.2.2) in modo da ottenere per ogni ramo una sequenza $\boldsymbol{c} = (c_1, \cdots, c_k)$ di k bit *di protezione* del segmento. Alla fine viene quindi trasmessa una parola $\boldsymbol{x} = (\boldsymbol{m}, \boldsymbol{c}_1, \boldsymbol{c}_2)$ di $n = 3k$ bit ottenendo un tasso complessivo $R_c = 1/3$, eventualmente *aumentato* mediante una successiva operazione di perforazione.

Trasmissione e decodifica soft La figura sotto ha il duplice scopo di riepilogare la notazione adottata e di evidenziare la presenza di un canale *soffice* che, pur includendo codifica di linea, modem e canale AWGN (§ 17.3), non prevede la presenza di un decisore, e dunque fornisce al decodificatore la sequenza $\boldsymbol{y}$ di valori analogici (*soft*) necessaria per la decodifica turbo, con elementi $y_j = x_j + \varepsilon$ in cui ε è una v.a. gaussiana a media nulla, varianza σ^2 e valori indipendenti. Il processo di decodifica è anch'esso di tipo soft e per questo detto SISO, ed il suo esito viene espresso come un valore di *verosimiglianza logaritmica a posteriori* L^p per ognuno dei k bit m_i del messaggio informativo, in modo da poter applicare un criterio di massima prob. a posteriori o MAP (§ 17.1.2) ossia decidere che $\hat{m}_j = 1$ o 0 a seconda se $L^p(m_j) \gtrless 0$.



Verosimiglianza logaritmica ed informazione estrinseca Finché non si attua la decodifica, l'osservazione del valore y_j in uscita dal canale permette il calcolo della verosimiglianza logaritmica a posteriori (nel seguito LLR ossia *log likelihood ratio*) per

[67] L'acronimo sta per *Recursive Sistematic Convolutional*, ed al § 17.4.2.4 ne è raffigurato un possibile schema architetturale. Il motivo di questa scelta è triplice: da un lato un RSC è simile ad uno *scrambler* pseudo random, e la teoria di Shannon basa la sua dimostrazione (vedi nota 14 a pag. 560) su codeword casuali; inoltre, un RSC ha prestazioni migliori dei *CC* classici per bassi valori di E_b/N_0. Infine, solo poche sequenze di lunghezza finita in ingresso ne producono di lunghezza finita in uscita, indice di una *elevata ridondanza*.

[68] In realtà possono anche essere diversi e con un diverso tasso R_c, ma non si desidera appesantire la trattazione.

[69] Ad esempio, l'interleaver può essere implementato mediante una sequenza di numeri pseudo casuali da utilizzare ciclicamente come indice di scrittura in un array dove si memorizzano gli elementi della sequenza di ingresso, e la cui lettura avviene poi in modo sequenziale.

ciascun bit x_j in ingresso al canale soffice come

$$L\left(x_j\right) = \ln \frac{Pr\left(x_j=1/y_j\right)}{Pr\left(x_j=0/y_j\right)} = \ln \frac{Pr\left(y_j/x_j=1\right)Pr\left(x_j=1\right)}{Pr\left(y_j/x_j=0\right)Pr\left(x_j=0\right)} = \ln \frac{Pr\left(y_j/x_j=1\right)}{Pr\left(y_j/x_j=0\right)} + \ln \frac{Pr\left(x_j=1\right)}{Pr\left(x_j=0\right)}$$
$$= L^c\left(y_j\right) + L^a\left(x_j\right) \tag{17.36}$$

in cui alla seconda eguaglianza si applica il teorema di Bayes, ed il risultato si interpreta osservando che $L\left(x_j\right)$ è somma di due termini: il primo $L^c\left(y_j\right)$ è dovuto al canale e dipende solo dal valore y_j ricevuto, mentre il secondo $L^a\left(x_j\right)$ è legato alla prob. *a priori* di x_j ed è nullo se 0 ed 1 sono equiprobabili.

A differenza della (17.36), il valore $L^p\left(x_j\right) = \ln \frac{Pr\left(x_j=1/\boldsymbol{y}\right)}{Pr\left(x_j=0/\boldsymbol{y}\right)}$ della LLR a posteriori in uscita dal decoder è ottenuto a partire da *tutti* i bit ricevuti $\boldsymbol{y}$, compresi quelli di ridondanza $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$, e per questo la decisione MAP presa in base ai valori di $L^p\left(x_j\right)$ contiene *meno errori*. La variazione della LLR $L^p\left(x_j\right)$ rispetto alla (17.36) vale

$$L^e\left(x_j\right) = L^p\left(x_j\right) - L\left(x_j\right) = L^p\left(x_j\right) - L^c\left(y_j\right) - L^a\left(x_j\right) \tag{17.37}$$

e prende il nome di *informazione estrinseca*[70] in quanto aggiunta da parte del decoder; si può dimostrare che questa *non dipende* dai valori di $\boldsymbol{m}$, ma solo *da quelli della ridondanza* $\boldsymbol{c}_1$ e $\boldsymbol{c}_2$.

Soft input soft output a quattro porte L'algoritmo SISO adottato nella decodifica turbo può essere schematizzato come nella figura a fianco, che lo mostra accettare in ingresso le componenti della LLR (17.36) ovvero i contributi delle singole osservazioni $L^c\left(y_j\right)$ e quelli a priori $L^a\left(x_j\right)$, e ottenere in uscita sia il valore di LLR a posteriori $L^p\left(x_j\right)$ (ottenuta applicando il metodo di decodifica) sia quello della informazione estrinseca ottenuta applicando la (17.37).

Valutazione della LLR di ingresso e di uscita al SISO Consideriamo di effettuare una trasmissione binaria antipodale, in cui i valori del segnale agli istanti di simbolo sono espressi come $x_j = \left(2m_j - 1\right)$ ossia assumono i valori ± 1 quando $m_i = 1$ oppure 0, e lo stesso dicasi per i bit di protezione c_{1j} e c_{2j}: in tal caso il termine $L^c\left(y_j\right) = \ln \frac{Pr\left(y_j/x_j=1\right)}{Pr\left(y_j/x_j=0\right)}$ (eq. (17.36)) assume il valore[71]

$$L^c\left(y_j\right) = \frac{2}{\sigma^2} y_j \tag{17.38}$$

Per quanto riguarda invece $L^p\left(x_i\right) = \ln \frac{Pr\left(x_j=1/\boldsymbol{y}\right)}{Pr\left(x_j=0/\boldsymbol{y}\right)}$ diciamo che nel contesto dei codici RSC usati nel caso in esame il relativo decodificatore opera su di un traliccio analogo

[70] Questo è il nome attribuito a tale quantità dalla comunità che ha lavorato alla definizione dei turbocodici. In effetti, essendo la LLR un logaritmo di probabilità può a tutto diritto essere chiamata *informazione*, ma espressa in *nat* anziché in bit, avendo adottato un logaritmo in base e.

[71] Infatti ora y_i è una v.a. gaussiana con media $x_i = \pm 1$ e varianza σ^2, e dunque

$$\ln \frac{Pr(y/x=1)}{Pr(y/x=0)} = \ln \frac{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y+1)^2}{2\sigma^2}\right)} = -\frac{(y-1)^2}{2\sigma^2} + \frac{(y+1)^2}{2\sigma^2} = \frac{-y^2+2y-1+y^2+2y+1}{2\sigma^2} = \frac{4y}{2\sigma^2} = \frac{2}{\sigma^2} y$$

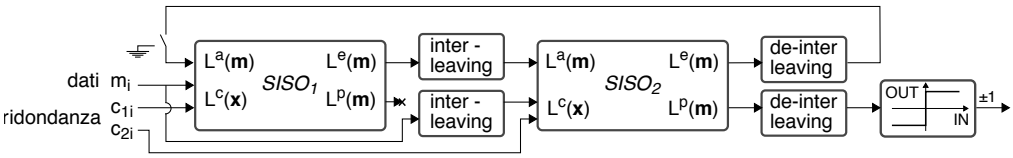


Figura 17.9: Schema di decodifica turbo per un codificatore RSC parallelo

a quello visto nella decodifica di Viterbi, ed il valore $L^p\left(x_j\right)$ può essere *approssimato* ricorrendo ad una decodifica SOVA (§ 17.4.2.7)[72], ma per ottenere il suo valore esatto (e dunque le migliori prestazioni) occorre ricorrere ad una modifica dell'algoritmo BCJR[73], le cui problematiche numeriche spingono però ad utilizzare metodi sub-ottimi e che operano direttamente nel dominio logaritmico[74].

Decodifica Siamo finalmente in grado di illustrare l'operatività della tecnica, con l'aiuto di fig. 17.9, in cui (come nel seguito) si adotta la notazione $L\left(\boldsymbol{m}\right)$ per indicare tutti i valori $L\left(m_j\right)$ per $j = 1, 2, \cdots, k$:

1. una prima decodifica SISO_1 considera nulla la verosimiglianza *a priori* ovvero $L^a(\boldsymbol{m}) = 0$ (switch *a massa*), e in base a L_c in uscita dal canale, relativa ai bit di $\boldsymbol{m}$ e di $\boldsymbol{c}_1$ (prodotti dal primo RSC di codifica) ottiene la verosimiglianza *a posteriori* $L_1^p(\boldsymbol{m})$ e da questa l'informazione estrinseca $L_1^e(\boldsymbol{m}) = L_1^p - L^c$ che *dipende solo* dai valori di $\boldsymbol{c}_1$;

2. il blocco SISO_2 adotta $L_1^e(\boldsymbol{m})$ come valore della LLR *a priori* $L_2^a(\boldsymbol{m})$ per i bit di messaggio $\boldsymbol{m}$, dopo che l'interleaver ne ha posto i valori nello stesso ordine con cui si sono presentati a RSC_2. Dato che L_1^e dipende dai valori di $\boldsymbol{c}_1$ a cui SISO_2 non ha accesso, rappresenta effettivamente qualcosa in *più*. SISO_2 esegue l'algoritmo di decodifica ottenendo $L_2^p(\boldsymbol{m})$ a partire da $L^c(\boldsymbol{m})$, $L^c(\boldsymbol{c}_2)$ e $L_2^a(\boldsymbol{m})$, e valuta $L_2^e = L_2^p - L^c - L_2^a$, che dipende solamente da $\boldsymbol{c}_2$;

3. dopo essere stata di nuovo riordinata temporalmente, l'informazione $L_2^e(\boldsymbol{m})$ viene fornita al blocco SISO_1 sull'ingresso *a priori* $L_1^a(\boldsymbol{m})$, in quanto anch'essa rappresenta qualcosa che SISO_1 non può calcolare per suo conto. Ecco così attuato il principio di *controreazione!* Ciò consente di ottenere dei nuovi valori per L_1^p e $L_1^e = L_1^p - L^c - L_1^a$;

[72]Per poter utilizzare anche le prob. a priori in SOVA occorre che nella (17.29) venga sommato anche un termine $\ln\left(p\left(x_j\right)\right)$.

[73]L. BAHL; J. COCKE; F. JELINEK; J. RAVIV, *Optimal decoding of linear codes for minimizing symbol error rate*, in IEEE Trans. on Inf. Theory, March 1974, per come modificato in C BERROU, A GLAVIEUX, *Near optimum error correcting coding and decoding: Turbo-codes*, IEEE Trans. on Comm., Oct. 1996.

In parole povere, il traliccio è esaminato oltre che da sinistra a destra, anche da destra a sinistra, permettendo il calcolo per ogni istante i della probabilità *congiunta* di trovarsi in uno stato, e che sia stato trasmesso un valore x_i, da cui saturando sugli stati ottenere i valori $Pr\left(x_i = 1/\boldsymbol{y}\right)$ e $Pr\left(x_i = 0\right)/\boldsymbol{y})$. Tale procedura fu poi adottata nel contesto della stima di parametro dei *modelli di Markov nascosti* (HMM) utilizzati per il riconoscimento del parlato, ma quella *è un'altra storia.*

[74]Vedi ad es. P. ROBERTSON; E. VILLEBRUN; P. HOEHER, *A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain*, Proc. IEEE ICC '95

4. se i valori di L_1^p e L_2^p sono abbastanza simili per tutti gli indici $j = 1, 2, \cdots, k$ i due rami sono collaborativamente addivenuti alla stessa conclusione, e la decodifica finale per tutti gli $\hat{m}_j$ si ottiene dall'uscita L^p di uno dei due SISO valutando se $L^p \gtrless 0$; altrimenti, si torna al passo 2. La tecnica descritta si è mostrata capace di convergere nel giro di una decina di iterazioni.

Utilizzi I turbo codici sono utilizzati, oltre che nei sistemi UMTS ed LTE, dagli standard DVB-RCS, WiMax, e da missioni spaziali. Il principio della codifica turbo si applica non solo al caso accennato degli RSC paralleli, ma può essere adottato anche per schemi seriali, e per *codici prodotto*. Il blocco di codifica interno può altresì essere sostituito da un modulatore-demodulatore *con memoria*, come ad es. il TCM (pag. 537). In base alla stessa logica anche l'equalizzazione MLSD di un canale con memoria (§ 18.4.5) può beneficiare di uno schema turbo, in cui equalizzatore e decodificatore si scambiano iterativamente informazione estrinseca per addivenire ad una decisione condivisa[75].

17.5.2 Codifica a bassa densità di controllo parità

Questo approccio si basa su di un codice lineare *a blocchi* caratterizzato da una una *matrice di controllo* $\boldsymbol{H}$ con una *bassa densità di uni*, da cui l'acronimo LDPC (*low-density parity-check*).

Riprendendo i concetti espressi al § 17.4.1, la moltiplicazione con somma modulo due $\oplus$ (pag. 567) $\boldsymbol{x} = \boldsymbol{m} \cdot \boldsymbol{G}$ tra il vettore *riga* $\boldsymbol{m}$ dei k bit di messaggio e la matrice binaria $\boldsymbol{G}$ (detta *generatrice*) con k righe ed n colonne produce una codeword $\boldsymbol{x}$ di n elementi. Se $\boldsymbol{G}$ è posta nella forma $\boldsymbol{G} = [\boldsymbol{I}_k \,|\boldsymbol{P}]$ con $\boldsymbol{I}_k$ matrice identità con k righe e colonne e $\boldsymbol{P}$ matrice *di parità* di k righe per $n-k$ colonne, il codice è detto *sistematico* e le codeword possono esser scritte come $\boldsymbol{x} = \begin{bmatrix} m_1 \cdots m_k & c_1 \cdots c_{n-k} \end{bmatrix}$ in cui[76] $c_j = \sum_{i=1}^{k}{}_{\oplus} m_i p_{ij}$ valuta la parità sui bit m_i per i quali $p_{ij} = 1$.

Un codice LDPC *non è* definito a partire da $\boldsymbol{G}$ bensì dalla matrice $\boldsymbol{H}$ *di controllo* di dimensione $(n-k) \times n$ e che nel caso sistematico ha la forma[77] $\boldsymbol{H} = \left[\boldsymbol{P}^T \,|\boldsymbol{I}_{n-k}\right]$, ed in generale soddisfa la condizione (valida anche per un codice *non* sistematico)

$$\boldsymbol{G} \cdot \boldsymbol{H}^T = \boldsymbol{0}_{k \times (n-k)}$$

in quanto ciascuna riga di $\boldsymbol{H}$ *è ortogonale*[78] ad ogni riga di $\boldsymbol{G}$; pertanto risulta[79] $\boldsymbol{H} \cdot \boldsymbol{x}^T = \boldsymbol{0}_{n-k}^T$ se e solo se $\boldsymbol{x}$ è una codeword; mentre in presenza di errori il vettore ricevuto è $\boldsymbol{y} \neq \boldsymbol{x}$, e se il codice è sistematico il prodotto $\boldsymbol{H} \cdot \boldsymbol{y}^T \neq \boldsymbol{0}$ è detto *sindrome* e viene usato per individuare i bit errati.

Precisiamo ora che le codeword $\boldsymbol{x}$ di un codice LDPC non fanno distinzione tra bit di messaggio m e di parità c, e sebbene il codice possa essere di tipo sistematico,

[75] Vedi ad es. `https://en.wikipedia.org/wiki/Turbo_equalizer`

[76] Si adotta il simbolo $\sum_{\oplus}$ per intendere una somma *modulo due*.

[77] Ci si discosta dalla notazione adottata a pag. 568 in quanto la $\boldsymbol{H}$ definita qui è *la trasposta* di quella definita in tale sede.

[78] Vedi ad es. S.Lin, D.J.Costello, *Error control coding*, Prentice-Hall 1983

[79] Infatti $\boldsymbol{H} \cdot \boldsymbol{x}^T = \boldsymbol{H} \cdot (\boldsymbol{m} \cdot \boldsymbol{G})^T = \boldsymbol{H} \cdot \boldsymbol{G}^T \cdot \boldsymbol{m}^T = (\boldsymbol{G} \cdot \boldsymbol{H}^T)^T \cdot \boldsymbol{m}^T = \boldsymbol{0}_{(n-k)\times k} \cdot \boldsymbol{m}^T = \boldsymbol{0}_{n-k}^T$

è di gran lunga preferibile che non lo sia, per i motivi presto illustrati; questo fa si che la decodifica basata sulla sindrome non sia applicabile. Un modo per descrivere il funzionamento di un LDPC è pensare che ogni riga i di $\boldsymbol{H}$ rappresenti il vincolo imposto sulle codeword da una tra $n-k$ *equazioni* di parità del tipo

$$\sum_{\oplus}^{n} x_j h_{ij} = 0$$

equivalente riga per riga dell'espressione $\boldsymbol{H} \cdot \boldsymbol{x}^T = \boldsymbol{0}$.

Esempio La matrice di controllo $\boldsymbol{H}$ riportata sotto corrisponde alle quattro equazioni di vincolo scritte a fianco, che devono essere soddisfatte dai bit x_j delle codeword esenti da errore. Il codice risultante è descritto dai parametri $n, k = 8, 4$ e da un tasso $R_c = 1/2$.

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{cases} x_2 \oplus x_4 \oplus x_5 \oplus x_8 = 0 \\ x_1 \oplus x_2 \oplus x_3 \oplus x_6 = 0 \\ x_3 \oplus x_6 \oplus x_7 \oplus x_8 = 0 \\ x_1 \oplus x_4 \oplus x_5 \oplus x_7 = 0 \end{cases}$$

A parte *il piccolo dettaglio* di come poter effettuare la codifica[80] $\boldsymbol{x} = \boldsymbol{m} \cdot \boldsymbol{G}$, per la matrice dell'esempio possiamo osservare che ogni bit x_i compare in due equazioni, ed ogni equazione si applica a quattro bit. In generale un codice LDPC si dice *regolare* se presenta esattamente $w_c \ll n-k$ elementi pari ad uno per ogni colonna, e $w_r = w_c \frac{n}{n-k}$ uni per ogni riga, a cui corrisponde un tasso $R_c = k/n = 1 - w_c/w_r$.

Grafo di Tanner E' il nome dato al grafo di cui $\boldsymbol{H}$ è la matrice *di adiacenza*, e che risulta essere tipo *bipartito* ovvero i cui vertici si dividono in due insiemi, tra gli elementi dei quali non sono presenti archi. Si traccia (fig. 17.10) riportando *sotto* i nodi (detti *variabile*) associati agli n bit ricevuti[81] x_j, e *sopra* quelli (*di controllo*) c_i che verificano il rispetto delle equazioni di vincolo; tra questi nodi si traccia un arco tra x_j e c_i se è presente un *uno* tra la riga i e la colonna j di $\boldsymbol{H}$, ovvero se $h_{ij} = 1$.

In altre parole, i w_c *uni* nelle n colonne rappresentano le connessioni dei nodi x_j (ed infatti ne troviamo due) mentre i w_r uni sulle $n-k$ righe indicano le connessioni dei nodi c_i (e ne troviamo quattro per ciascuno).

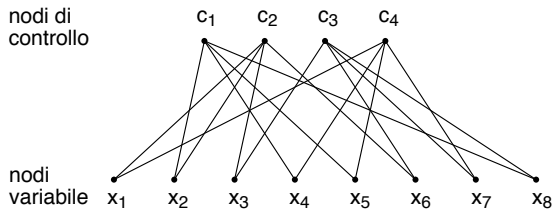


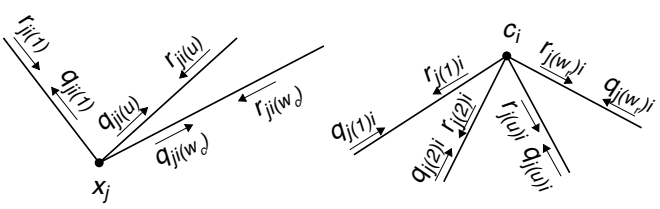
Figura 17.10: Grafo di Tanner per la matrice dell'esempio precedente

[80] In linea di principio per trovare una matrice generatrice $\boldsymbol{G}_{k \times n}$ tale che $\boldsymbol{G} \cdot \boldsymbol{H}^T = \boldsymbol{0}$ si può procedere trasformando prima $\boldsymbol{H}$ nella forma *canonica* di un codice sistematico, modificandone le righe applicando il metodo di Gauss; ciò determina però una $\boldsymbol{G}$ per nulla sparsa, ed una eccessiva complessità di codifica per n elevato. Fortunatamente hanno escogitato metodi più efficienti, anche ricorrendo a codici LDPC *non regolari*; per un approfondimento si può vedere W.E.RAYAN, *An introduction to LDPC code*, Univ. of Arizona 2003, ed es. presso `http://tuk88.free.fr/LDPC/ldpcchap.pdf`.

[81] La nomenclatura adottata in letteratura indica i nodi-variabile come *v-nodes* e li rappresenta con la lettera c, mentre quelli di controllo (*check-nodes* o *nodi-fattore*) sono rappresentati dalla lettera f. Preferisco qui attenermi alla notazione dell'attuale contesto espositivo.

17.5.2.1 Decodifica iterativa

La particolarità più rilevante di un LDPC è quella di svolgere la decodifica in modo *iterativo* basandosi su di un ripetuto scambio di messaggi di natura probabilistica tra nodi-variabile e nodi di controllo, realizzando una applicazione di *propagazione della credenza*[82] nota anche come *algoritmo somma-prodotto*. Lo scopo dell'algoritmo è individuare la codeword $\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}/\boldsymbol{y})$ che rende massima la prob. a posteriori (PAP) una volta noto il vettore $\boldsymbol{y}$ in uscita dal canale. La ricerca è condotta attraverso un *raffinamento successivo* di ipotesi, a partire dalla conoscenza dei valori y_j indipendenti che fornisce una $p(\boldsymbol{x}/\boldsymbol{y}) = \prod_{j=1}^{n} p(x_j/y_j)$ *di partenza*.



Ad ogni iterazione ciascun nodo-variabile j invia a tutti i w_c nodi di controllo $i(u)$, $u = 1, \cdots, w_c$ a cui è connesso un messaggio q_{ji} (pensiamo stia per *query*) in cui comunica la *sua percezione* della probabilità p_j di essere *pari ad uno*. Ricevuti i messaggi q_{ji}, ogni nodo di controllo c_i invia a ciascun nodo variabile $j(u)$, $u = 1, \cdots, w_r$ a cui è connesso un messaggio r_{ji} in cui *risponde* con nuove stime di p_j ottenute combinando le opinioni q_{ji} ricevute da tutti i nodi *tranne quello a cui è diretta la risposta*. A questo punto i nodi-variabile generano nuovi messaggi q_{ji} integrando la propria opinione di partenza con quella r_{ji} ricevuta dai nodi di controllo, omettendo di includere l'informazione ricevuta da quello verso cui è diretto il messaggio. Il senso di omettere l'informazione proveniente dal destinatario è quello di attingere unicamente all'informazione *estrinseca*, ossia non ricavabile autonomamente a destinazione, come avviene per i codici turbo.

L'opinione *di partenza* sulla PAP p_j che il bit x_j sia pari ad 1 è ottenuta (per ogni istante $j = 1, \cdots, n$) a partire dal valore y_j in uscita dal canale come[83]

$$p_j = p(x = 1/y) = \frac{p(y/x = 1)\, p(x = 1)}{p(y)} = K \cdot p(y/x = 1)$$

in cui $K = \frac{p(x=1)}{p(y)} = \frac{1}{p(y/x=1)+p(y/x=0)}$[84]. A seconda se in presenza di un BSC (§ 17.1.1) oppure di un canale *soffice* (o AWGN, pag. 587)

- BSC: il canale compie una decisione *hard* ed emette un valore y pari a zero od uno, con prob. condizionata *in avanti* $p(y/x)$ di valore $p(1/1) = p(0/0) = 1 - p_e$, $p(0/1) = p(1/0) = p_e$. Dunque $K = 1$[85] sia per $y = 1$ che per $y = 0$, e $p(x = 1/y) = p_e$ se $y = 0$ oppure $1 - p_e$ quando $y = 1$;

[82] Dall'inglese *belief propagation*, vedi ad es. `https://en.wikipedia.org/wiki/Belief_propagation`

[83] Applicando il solito teorema di Bayes, ed omettendo il pedice j per estetica e spazio.

[84] Questo perché $p(x = 1)$ è la prob. *a priori* considerata pari a $1/2$, e quindi $\frac{p(x=1)}{p(y)} = \frac{p(x=0)}{p(y)}$. Imponendo ora $p(x = 1/y) + p(x = 0/y) = 1$ si ottiene $(p(y/1) + p(y/1))\, K = 1$, e dunque il risultato.

[85] In quanto $p(1/1) + p(1/0) = 1 - p_e + p_e = 1$

- AWGN: in funzione del valore binario di $x \in \{0,1\}$ il canale emette il valore continuo $y = 2x - 1 + \varepsilon$ che è una v.a. gaussiana a valori indipendenti, media ± 1 a seconda se $x = 1$ o 0, e varianza σ^2; si ha quindi $p(y/x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(y\pm 1)^2}{2\sigma^2}\right\}$.

In entrambi i casi, ogni nodo variabile j pone il messaggio iniziale $q_{ji} = p_j$ uguale per tutti gli i.

Calcolo ai nodi di controllo Consideriamo ora un nodo c_i che riceve più di un q_{ji}, e che *sa* che tra i nodi-variabile a lui collegati ci deve essere un numero *pari* di uni. Per ottenere il valore del messaggio r_{ji} da inviare indietro, c_i *somma* le stime di probabilità ricevute.

Esempio Il nodo c_1 dell'esempio di fig. 17.10 deve far valere il vincolo $x_2 \oplus x_4 \oplus x_5 \oplus x_8 = 0$ ovvero nei quattro bit ci devono essere 4 uni, oppure due, oppure nessuno. Genera quindi il messaggio r_{21} diretto a x_2 tenendo conto delle probabilità q_{j1} ricevute da x_4, x_5 e x_6 e, considerandole *statisticamente indipendenti*, stima

$$r_{21} = \hat{p}_2 = q_{41}(1-q_{51})(1-q_{61}) + (1-q_{41})q_{51}(1-q_{61}) + (1-q_{41})(1-q_{51})q_{61} + q_{41}q_{51}q_{61}$$

ossia pari a quella che ci sia un altro uno, oppure tre. Calcola quindi in modo analogo i messaggi r_{41} , r_{51} e r_{61} omettendo ogni volta di considerare l'informazione originata dal nodo destinazione.

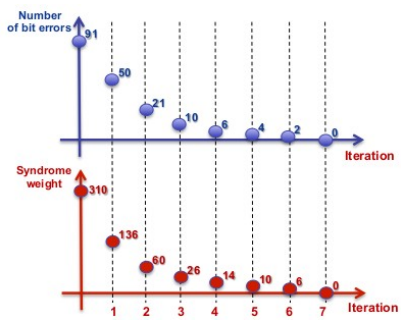
Calcolo ai nodi-variabile A questo punto ogni nodo-variabile j ha ricevuto w_c messaggi r_{ji}, da cui ne calcola altrettanti da rispedire indietro, considerando oltre all'informazione $p_j^{(0)}$ proveniente dal canale anche quella r_{ji} proveniente dai nodi c_i, tranne quello di destinazione. Questa volta il calcolo di q_{ji} comporta *il prodotto* dei valori di probabilità ricevuti.

Esempio Il nodo x_1 ha ricevuto r_{12} e r_{14}, e valuta $q_{12} = p_1^{(1)} = k_2 p_1^{(0)} r_{14}$ da inviare a c_2 in cui[86] $k_2 = \frac{1}{\left(1-p_1^{(0)}\right)\left(1-r_{14}\right)+p_1^{(0)} r_{14}}$ serve per *normalizzare* la stima, dato che se k_2 non fosse presente $p_2^{(1)}$ risulterebbe *più piccolo* di tutti i valori ricevuti. In modo simile, il nodo x_1 calcola poi q_{14} da inviare a c_4 omettendo di usare r_{14}.

Arresto Ad ogni ciclo $v = 1, 2, \cdots$ si perviene ad una stima di probabilità $\hat{p}_j^{(v)}$ che ogni bit x_j sia pari ad uno utilizzando *tutte* le fonti informative[87], e da questa si ottiene una *ipotesi* di codeword $\tilde{\boldsymbol{x}}$ operando per ogni bit una decisione *hard* mediante una soglia di probabilità pari a 1/2. Se $\tilde{\boldsymbol{x}}$ soddisfa la condizione $\boldsymbol{H} \cdot \tilde{\boldsymbol{x}}^T = \boldsymbol{0}$ allora è una codeword ammissibile, e la decodifica è terminata: in figura si mostra l'andamento

[86] Il risultato si ottiene imponendo che la stessa normalizzazione valga anche per l'evento complementare, ovvero $1 - q_{21} = k_2\left(1 - p_1^{(0)}\right)\left(1 - r_{14}\right)$, ma dall'equazione *sopra* abbiamo anche $1 - q_{21} = 1 - k_2 p_1^{(0)} r_{14}$, ed eguagliando le due espressioni si consegue lo scopo.

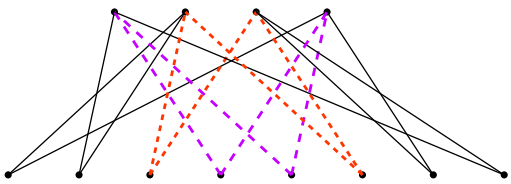
[87] Infatti questa stima non deve essere re-inviata a nessuno, per cui nel caso dell'esempio il nodo x_1 calcola $\tilde{p}_1^{(v)} = k_2 p_1^{(0)} r_{12} r_{14}$ con $k_2 = \frac{1}{\left(1-p_1^{(0)}\right)\left(1-r_{12}\right)\left(1-r_{14}\right)+p_1^{(v)} r_{12} r_{14}}$.



del numero di errori sul bit e del *peso* della sindrome al progredire delle iterazioni. Se invece anche dopo un loro ragionevole numero[88] la condizione non è mai verificata, significa che una eccessiva quantità di errori impedisce la decodifica corretta, e tale evenienza può essere segnalata agli stadi di elaborazione seguenti (particolare che non avviene per la decodifica *turbo*).

17.5.2.2 Attenti a quel ciclo

Il calcolo svolto sia dai nodi di controllo che da quelli -variabile implica che gli eventi a cui si riferiscono i messaggi ricevuti siano statisticamente indipendenti. Ma se il grafo associato alla matrice $\boldsymbol{H}$ presenta cicli di lunghezza ν, dopo l'iterazione numero $\nu/2$ l'ipotesi perde di validità, in quanto le stime di probabilità divengono dipendenti anche da quelle inviate dal nodo che le riceve.



In figura si evidenziano due cicli di lunghezza 4 presenti nel grafo di fig. 17.10, associati a quattro *uni* disposti agli angoli di una sottomatrice rettangolare di $\boldsymbol{H}$. In fase di progetto della matrice di controllo tale circostanza va evitata, e dato che non è possibile non avere cicli, è bene vincolarne il numero e la lunghezza minima ad valore ritenuto adeguato a non degradare le prestazioni.

17.5.2.3 Implementazione Min-Sum

Il metodo esposto al § 17.5.2.1 comporta difficoltà legate a dover moltiplicare molti valori di probabilità, determinando instabilità numerica per dimensioni n anche di decine di migliaia. Si preferisce allora lavorare nel dominio della verosimiglianza logaritmica (LLR), con un algoritmo del tutto simile a quello esposto, ma con alcune particolarità. Iniziamo con il definire la LLR della PAP di un bit x_j in perfetta analogia con la (17.36), ovvero

$$L\left(x_j/y_j\right) = L^c\left(y_j\right) + L\left(x_j\right)$$

La decodifica iterativa ha ora l'obiettivo di aumentare *il modulo* dalla LLR *a priori* $L\left(x_j\right) = \ln \frac{p(x_j=1)}{p(x_j=0)}$, inizialmente nullo, grazie all'apporto dell'informazione estrinseca proveniente dagli altri nodi.

Per quanto riguarda il contributo del canale $L^c\left(y_j\right) = \ln \frac{p(y_j/x_j=1)}{p(y_j/x_j=0)}$

- nel caso AWGN con mapping $y = 2x - 1 + \varepsilon$, in cui $x \in \{0, 1\}$ e $\varepsilon \in N(0, \sigma)$, si ha (vedi eq. (17.38)) $L^c\left(y_j\right) = \frac{2}{\sigma^2} y_j$;

[88] Tipicamente, tra dieci e trenta. Una simpatica animazione dell'evoluzione della decodifica può essere trovata presso `http://www.inference.org.uk/mackay/codes/gifs/demo2.html` sia per il caso BSC che AWGN.

- nel caso BSC con $P_e = p$ risulta $L^c\left(y_j\right) = \ln\frac{1-p}{p}$ se $y_j = 1$ ed $L^c\left(y_j\right) = \ln\frac{p}{1-p}$ quando $y_j = 0$.

In entrambi i casi si usa $L^c\left(y_j\right)$ per inizializzare i messaggi verso i nodi di controllo, che ora non valgono più q_{ji} ma $L\left(q_{ji}\right) = \ln\frac{Pr\left\{x_j=1\right\}}{1-Pr\left\{x_j=1\right\}}$.

Min Per il calcolo della LLR dei messaggi *di risposta* r_{ji}, il nodo c_i si auspica che la probabilità $p_j = r_{ji} = Pr\left\{x_j = 1\right\}$ sia uguale a quella che gli altri bit che partecipano al controllo svolto da c_i presentino un numero *dispari* di uni, ossia

$$L(r_{ji}) = L\left(Pr\left\{\sum_{\substack{j'=1\\ \oplus,\, j'\neq j}}^{n} x_{j'}h_{ij'} = 1\right\}\right) = \ln\frac{Pr\left\{r_{ji}=1\right\}}{Pr\left\{r_{ji}=0\right\}}$$

Dopo una serie di sviluppi analitici di cui tralasciamo l'approfondimento[89], sotto l'ipotesi di indipendenza statistica si arriva ad esprimere $L(r_{ji})$ in funzione approssimata della LLR $L\left(q_{ji}\right)$ dei q_{ji} da cui dipende, come

$$L(r_{ji}) \approx (-1)\left\{\prod_{j'\neq j}\mathrm{sgn}\left(L\left(q_{j'i}\right)\right)\right\}\cdot\min_{j'\neq j}\left\{\left|L\left(q_{j'i}\right)\right|\right\}$$

Il risultato si interpreta notando che il modulo (l'affidabilità) della LLR risultante $L(r_{ji})$ è determinato dal *più piccolo* dei moduli dei contributi $\left|L\left(q_{j'i}\right)\right|$ (il meno affidabile), da cui l'appellativo di *Min* a questo passaggio. Il segno positivo di $L(r_{ji})$ indica poi che $Pr\left\{r_{ji}=1\right\} > Pr\left\{r_{ji}=0\right\}$ (o viceversa se negativo), e si ottiene come prodotto dei segni di $L\left(q_{j'i}\right)$, indicando così una parità dispari o pari.

Sum I nodi-variabile aggiornano le stime di $L\left(x_j\right)$ come

$$L\left(x_j\right) = L^c\left(y_j\right) + \textstyle\sum_i L\left(r_{ji}\right)$$

da cui ottenere una ipotesi di codeword $\tilde{\boldsymbol{x}}$ con elementi 1 o 0 a seconda se $L\left(x_j\right) \gtrless 0$. Qualora $\boldsymbol{H}\cdot\tilde{\boldsymbol{x}}^T = \boldsymbol{0}$ la decodifica è terminata; altrimenti si calcolano le

$$L\left(q_{ji}\right) = L^c\left(y_j\right) + \textstyle\sum_{i'\neq i} L\left(r_{ji'}\right)$$

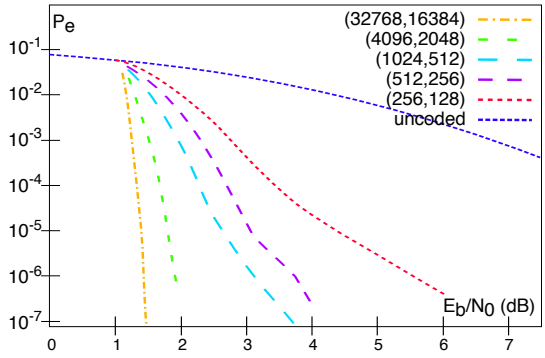
(passo *Sum*) e si torna al passo *Min*.

17.5.2.4 Prestazioni

La natura probabilistica del metodo di decodifica non consente di ottenere una espressione in forma chiusa della P_e in funzione di E_b/N_0, il cui grafico deve essere ottenuto mediante simulazione al computer ottenuta mediando su un gran numero di vettori ricevuti $\boldsymbol{y}$: accade infatti che, sebbene il codice si comporti generalmente bene, per alcune configurazioni di partenza l'algoritmo di decodifica non riesca a convergere.

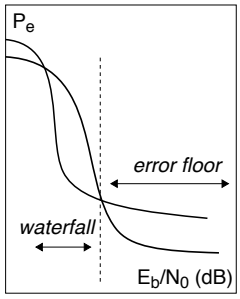
Di seguito sono riportate le prestazioni ottenute da un codice LDPC regolare con

[89] Che può essere svolto incrociando le infomazioni presenti oltre che nel già citato W.E.Rayan, *An introduction to LDPC code*, anche in T.Strutz, *Low-Density Parity-Check codes - An introduction* presso `http://www1.hft-leipzig.de/strutz/Kanalcodierung/ldpc_introduction.pdf`, con la modifica di B.Sklar, *A Primer on Turbo Code Concepts*, IEEE Comm. Mag. 1998 ad es. presso `http://wireless.ece.ufl.edu/eel6550/lit/sklar_primer.pdf`



$w_c = 3$ e $w_r = 6$, $R_c = 1/2$, per una segnalazione antipodale su canale AWGN, con diverse scelte[90] della lunghezza del blocco n da 256 a 32768. A parte l'evidente guadagno rispetto alle prestazioni in assenza di codifica, i risultati possono essere confrontati con quelli di pag. 582 relativi al codice di Viterbi, e con il *limite di Shannon* (§ 17.3.4) che per un tasso $R_c = 1/2$ fissa un requisito minimo pari a $E_b/N_0 \geq 0.188$ dB, *mancato* dal miglior codice esaminato solamente per un dB e mezzo.

Una alternativa per la matrice $\boldsymbol{H}$ è quella che da luogo ad un codice *irregolare*, contraddistinto da un numero di uni per riga $w_r\,(i)$ e per colonna $w_c\,(j)$ non costanti, e che in generale consegue prestazioni *migliori* di un codice regolare. In questo caso i bit x_j con $w_c\,(j)$ più grande sono coinvolti in un maggior numero di vincoli e dunque la stima della loro probabilità diviene più affidabile; il miglioramento di verosimiglianza è quindi *distribuito* in modo più *diffuso* da parte dei nodi c_i con $w_r\,(i)$ maggiore, in quanto collegati ad un maggior numero di nodi-variable. Tra i codici irregolari si menzionano quelli *quasi-ciclici* e quelli *protografici*[91], le cui matrici $\boldsymbol{H}$ presentano una qualche struttura interna, che riduce la complessità del processo di co-decodifica.



Nel grafico di $P_e\,(E_b/N_0)$ ottenuto dalle simulazioni si può distinguere la presenza di due regioni, la prima cosidetta di *waterfall* (cascata) in cui oltre un certo valore di E_b/N_0 la P_e decade piuttosto bruscamente, a cui fa seguito una regione *piattaforma* (error floor) in cui la riduzione di P_e è molto più graduale, se non nulla. Questo comportamento è attribuibile a *quasi-codeword* ovvero sequenze $\tilde{\boldsymbol{x}}$ la cui sindrome $\boldsymbol{H} \cdot \tilde{\boldsymbol{x}}^T$ presenta un numero ridotto di uni, e che determina una situazione di *minimo locale* per il processo di decodifica. In genere l'error floor si manifesta *prima* per i codici con andamento *più ripido* nella regione di waterfall (come quelli irregolari), sussistendo una situazione di compromesso tra le due esigenze.

Il numero di iterazioni necessario per arrivare alla decodifica corretta diminuisce all'aumentare di E_b/N_0, della dimensione del blocco n, e del tasso R_c, ed è possibile tenerne conto nel fissare il numero massimo di iterazioni prima di dichiarare un fallimento.

Rispetto ai turbo codici gli LDPC hanno il vantaggio che

- la decodifica può essere parallelizzata;
- sono più adatti alle velocità di trasmissione elevate;

[90] Figura tratta dal già citato T.Strutz, ottenuta con il software di R. M. Neal disponibile presso `https://www.cs.toronto.edu/~radford/ftp/LDPC-2012-02-11/index.html`

[91] Il cui grafo corrispondente è costruito a partire da *prototipi* di sottografo.

- l'error floor si presenta per valori di P_e inferiori;
- resistono meglio agli errori a pacchetto;
- non è necessario alcun interleaver;
- uno stesso codice LDPC è adatto per diversi tipi di canale.

Tra gli svantaggi si citano

- una maggior complessità del codificatore;
- la realizzazione hardware può essere grande e ingombrante;
- un turbo codice si comporta meglio per lunghezze di blocco n più brevi e per tassi R_c minori.

17.5.2.5 Adozione

Il successo della codifica LDPC ha portato alla sua adozione nelle ultime generazioni di standard: dopo la tv satellitare DVB-S2 (2005) viene adottata anche per la diffusione terrestre (DVB-T2) e via cavo (DVB-C), secondo un schema concatenato con LDPC come codice interno e BCH esterno in modo da poter gestire il fenomeno dell'error floor.

E' inoltre adottata per i collegamenti a microonde Wi-MAX 802.16, per le reti wireless WiFi 802.11n, per collegamenti Ethernet 10GBase-T su cavo ritorto, per reti domestiche G.hn con distribuzione su linee elettriche, telefoniche e coassiali fino a 1 Gbit/s (ITU G.9960, 2009), nonchè nel sistema televisivo terrestre DTMB della repubblica popolare cinese, e nella telefonia 5G.[92]

[92] Vedi ad es. *An overview of channel coding for 5G NR cellular communications* presso doi:10.1017/ATSIP.2019.10

L'opera

Trasmissione dei Segnali e Sistemi di Telecomunicazione

è il risultato di un progetto ventennale di cultura libera, aggiornato di continuo ed evolutosi fino alla forma attuale. La sua disponibilità pubblica è regolata dalle norme di licenza Creative Commons

Attribuzione - Non commerciale - Condividi allo stesso modo

`https://creativecommons.org/licenses/by-nc-sa/4.0/deed.it`

e tutte le risorse relative al testo sono accessibili presso

`https://teoriadeisegnali.it/libro/`

Puoi contribuire al suo successo promuovendone la diffusione e supportarne lo sviluppo attraverso una donazione, in buona parte devoluta ai progetti *open source*[1] che ne hanno resa possibile realizzazione e divulgazione. Ai donatori viene accordato un accesso *vitalizio* al formato PDF *navigabile* di tutte le edizioni presenti *e future*.

[1]
- Lyx - `http://www.lyx.org/`
- LaTeX- `https://www.latex-project.org/`
- TeX Users Group - `https://tug.org/`
- Inkscape - `http://www.inkscape.org/`
- Gnuplot - `http://www.gnuplot.info/`
- Octave - `http://www.gnu.org/software/octave/`
- Geany - `https://www.geany.org/`
- Linux - `https://www.linux.it/`
- Free Software Foundation - `https://shop.fsf.org/`
- GNOME Foundation - `https://www.gnome.org/`
- Mozilla Foundation - `https://www.mozilla.org/it/`
- Wikipedia - `https://it.wikipedia.org`
- Internet Archive - `https://archive.org/about/`
- Creative Commons - `https://creativecommons.it/chapterIT/`
- WordPress - `https://it.wordpress.org/`
- Phplist - `https://www.phplist.org/`