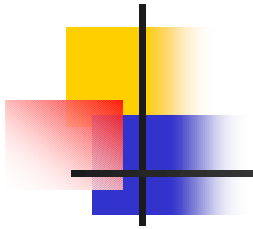
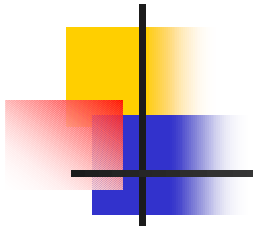


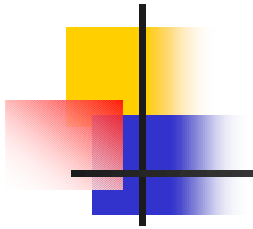
4. CODICI A RIVELAZIONE E CORREZIONE D'ERRORE

- **INTRODUZIONE**
- **STRATEGIE *ARQ* E *FEC***
- **TIPOLOGIE DI CODICI**
- **CODICI A BLOCCHI**
- **CODICI A RIPETIZIONE**
- **CODICI A CONTROLLO DI PARITÀ**
- **DISTANZA DI HAMMING**
- **CAPACITÀ RIVELATIVA E CORRETTIVA DI UN CODICE**



4. CODICI A RIVELAZIONE E CORREZIONE D'ERRORE

- **PROPRIETÀ DI UN CODICE A BLOCCHI**
- **CODICI SISTEMATICI**
- **CODICE DI HAMMING**
- **DECODIFICA A SINDROME**
- **CODICI CICLICI**
- **CODICI CONVOLUZIONALI**
- **DECODIFICA DI VITERBI**
- **DECODIFICA SEQUENZIALE**



INTRODUZIONE AI CODICI A CONTROLLO D'ERRORE

- Un sistema di telecomunicazione reale è soggetto ad errori di trasmissione dovuti alla presenza di rumore sul canale.
- La probabilità di errore P_{be} sul bit inviato sul canale è legata a diversi fattori:
  - alla potenza media del segnale in ricezione S_R;
  - alla bit rate di trasmissione r_b;
  - allo spettro di densità di potenza del rumore presente sul canale;
  - al tipo di canale (con rumore additivo, moltiplicativo, con caratteristiche di tempo-varianza, ecc.).

INTRODUZIONE AI CODICI A CONTROLLO D'ERRORE

- Nello studio della teoria dell'informazione, abbiamo visto che, se l'*information rate* R è inferiore alla capacità del canale, è possibile rendere arbitrariamente piccola la probabilità d'errore sui bit di informazione utilizzando opportune tecniche di codifica di canale.

- Il processo di codifica di canale consiste nell'aggiungere bit di ridondanza al messaggio che si vuole trasmettere. In fase di ricezione, la presenza di tali bit consente di rilevare o correggere eventuali errori introdotti nel messaggio dal rumore presente sul canale.

- Lo svantaggio della codifica di canale è che si devono trasmettere più bit di quanti ne siano effettivamente necessari per rappresentare il messaggio ⇨ a parità di altre condizioni, cresce il tempo richiesto per la trasmissione.

INTRODUZIONE AI CODICI A CONTROLLO D'ERRORE

- La codifica di canale, a parità di condizioni, non cambia la probabilità d'errore dei bit che viaggiano "fisicamente" sul canale.

- L'impiego della codifica di canale riduce la probabilità d'errore dei bit di informazione (ovvero dei bit del messaggio).

- Nel seguito vedremo le principali strategie di codifica e i principali codici che si possono utilizzare per proteggere il messaggio dal rumore presente sul canale.

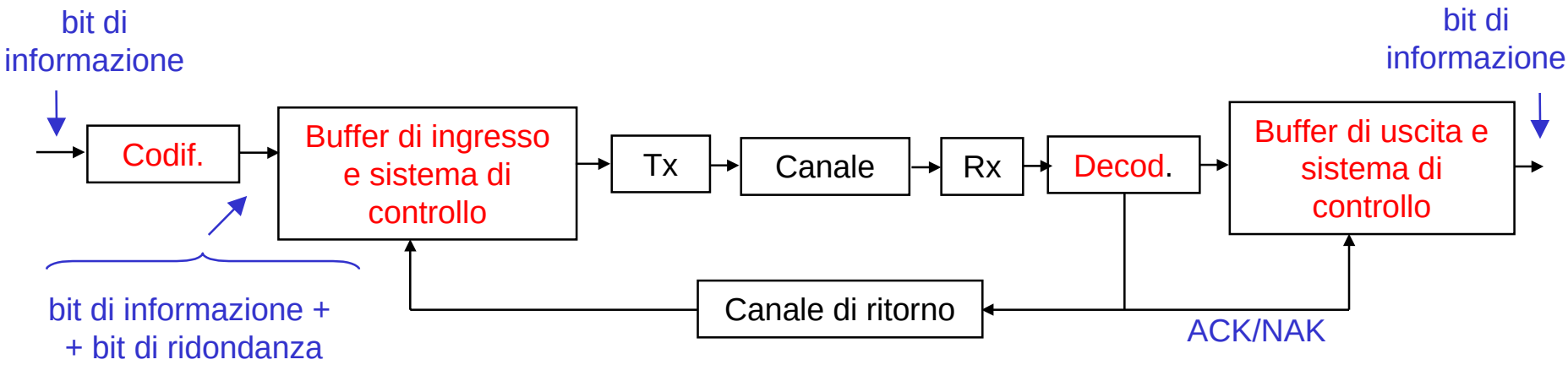
STRATEGIE DI IMPIEGO DEI CODICI A CONTROLLO D'ERRORE

- Esistono due possibili strategie per l'impiego dei codici a controllo d'errore:
  - si può usare un codice per rivelare errori (si riconosce la presenza di un errore, ma non lo si corregge);
  - si può usare un codice per correggere errori (si individua l'errore e lo si corregge).

- N.B.: Uno stesso tipo di codice può essere utilizzato per rivelare o per correggere errori a seconda dell'architettura del sistema di codifica/decodifica utilizzato.

STRATEGIA AUTOMATIC REPEAT AND REQUEST (ARQ)

- Strategia *Automatic Repeat and reQuest* (ARQ): utilizza i codici solo per rivelare la presenza di errori nel messaggio ricevuto. Quando si rivela la presenza di un errore, il ricevitore chiede la ritrasmissione del messaggio al trasmettitore (serve un canale di ritorno).

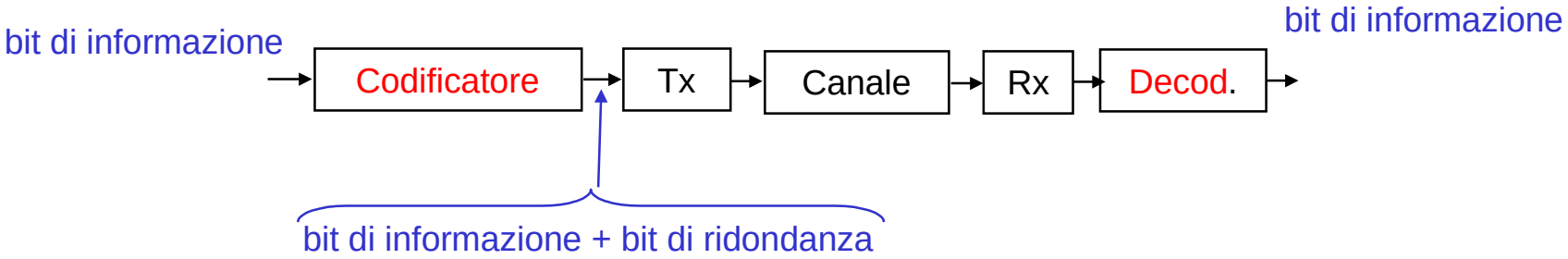


STRATEGIA AUTOMATIC REPEAT AND REQUEST (ARQ)

- Si possono utilizzare 3 diverse tecniche per implementare una strategia di tipo ARQ:
  - Stop and Wait;
  - Go Back N;
  - Selective Repeat.
- Tali tecniche sono studiate in maniera approfondita nel corso di Reti di Telecomunicazioni, a cui si rimanda per approfondimenti.

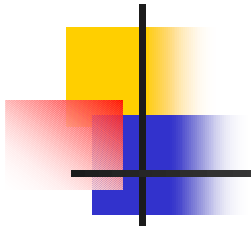
STRATEGIA FORWARD ERROR CORRECTION (FEC)

- Strategia *Forward Error Correction* (FEC): utilizza i codici sia per rivelare la presenza di errori, sia per correggerli (è particolarmente indicata nei casi in cui non sia possibile effettuare la ritrasmissione del messaggio).



CODICI: DEFINIZIONI

- Si definiscono:
  - capacità di rivelazione di un codice il numero massimo di errori che esso riesce a rivelare in una parola di codice.
  - capacità di correzione di un codice il numero massimo di errori che esso riesce a correggere in una parola di codice.
- Le prestazioni di un codice si misurano sulla base della capacità di rivelazione e/o di correzione degli errori, dell'efficienza $R_c=k/n$ ($k \rightarrow$ bit di messaggio, $n \rightarrow$ lunghezza della parola di codice) e della complessità realizzativa.

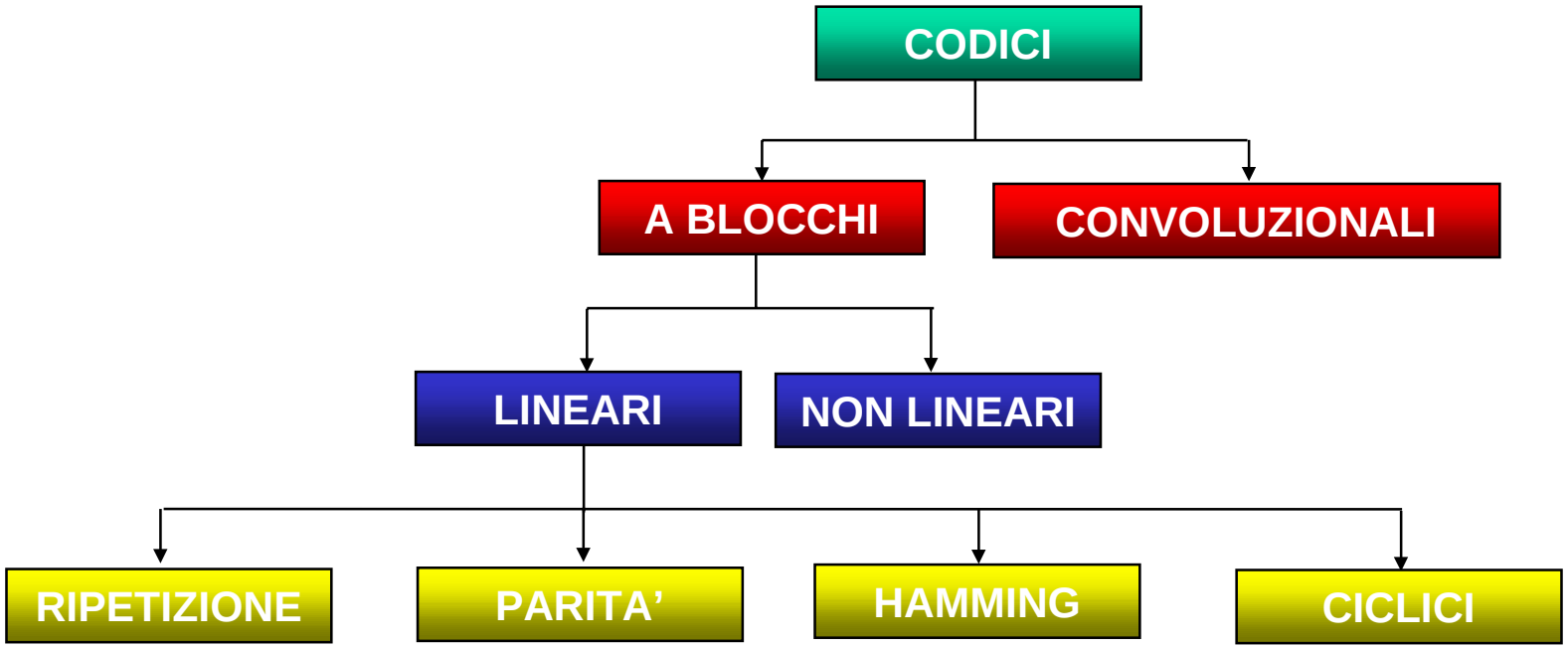


TIPOLOGIE DI CODICI

- Esistono diverse tipologie di codici che si distinguono per il modo con cui introducono ridondanza nel messaggio.

- In particolare, i codici possono essere divisi in 2 grandi classi: la classe dei codici a blocchi e quella dei codici convoluzionali. La classe dei codici a blocchi può essere ulteriormente suddivisa in sottoclassi.

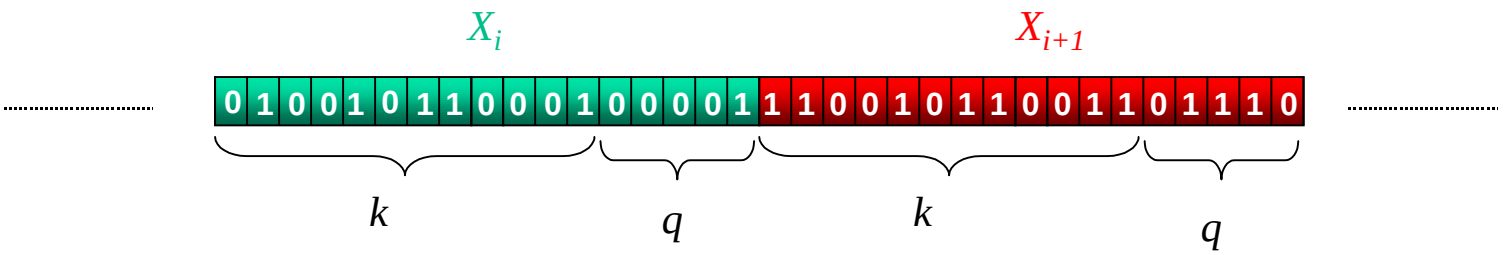
TIPOLOGIE DI CODICI



TIPOLOGIE DI CODICI

CODICI A BLOCCHI *(n,k)*

- Il messaggio è diviso in blocchi di k simboli, a cui vengono associate parole di codice formate da $n=k+q$ simboli. Si introducono quindi q bit di ridondanza.
- Nei codici a blocchi non c'è correlazione tra i simboli di una parola di codice e quelli della successiva.

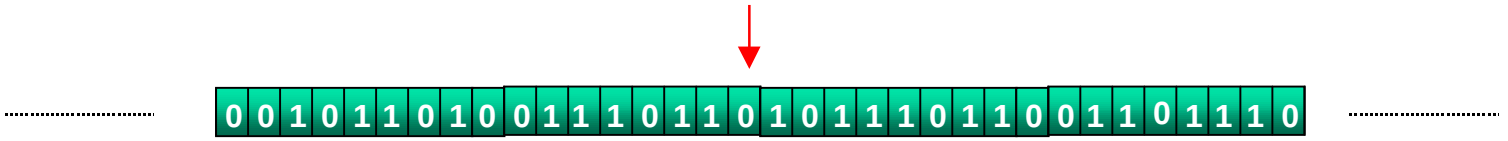


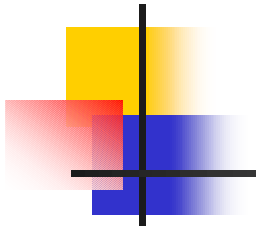
TIPOLOGIE DI CODICI

CODICI CONVOLUZIONALI

- La struttura non è costituita da singole parole di codice: un simbolo del messaggio influenza la generazione di molti simboli della sequenza trasmessa.

Non è possibile associare ciascun bit della sequenza codificata ad un particolare messaggio. Il valore di ogni bit è influenzato da più messaggi.





CODICI A BLOCCHI (n,k)

- Focalizziamo l'attenzione sui codici a blocchi (n,k).
- Innanzitutto, studiamo 2 semplici esempi introduttivi di codici a blocchi:
  - codici a ripetizione;
  - codici a controllo di parità.

CODICI A RIPETIZIONE

- I codici a ripetizione fanno parte della categoria dei codici a blocchi, lineari e sistematici (vedremo in seguito la definizione di codice lineare e sistematico).
- La strategia di codifica consiste nel costruire parole di codice che ripetano n volte il bit di informazione che si vuole trasmettere:

$$0 \rightarrow \overbrace{000\ldots00}^{n}$$

$$1 \rightarrow 111...11$$

- Nei codici a ripetizione si ha: $k=1$, $q=n-1$. Pertanto siamo in presenza di codici $(n,1)$.

CODICI A RIPETIZIONE

- Vediamo di calcolare quanto vale la probabilità di errore che si ottiene con un codice a ripetizione.
- Se gli errori di trasmissione avvengono in modo casuale e indipendente con probabilità $P_{be}=\alpha$ ($\alpha<<1$), la probabilità $P(i,n)$ che occorrano *i* errori in una parola di codice di n bit è pari a:

$$P(i,n)=\binom{n}{i}\alpha^{i}(1-\alpha)^{n-i}\cong\binom{n}{i}\alpha^{i}$$

- I codici a ripetizione sono efficienti se α è sufficientemente piccolo da verificare la condizione $P(i+1,n) << P(i,n)$.

CODICI A RIPETIZIONE: ESEMPIO

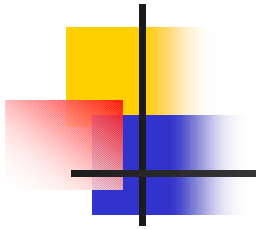
- Consideriamo un codice a ripetizione (3,1) (ovvero con $n=3$ e $k=1$). Tale codice è caratterizzato dalle seguenti parole:

$$0 \rightarrow 000 \quad \text{e} \quad 1 \rightarrow 111$$

- Tutte le volte che in ricezione si ottengono parole diverse da 000 e 111 (per esempio 001 o 101) si è chiaramente in presenza di errori.
- In ricezione, a seconda della strategia di codifica/decodifica adottata, possiamo:
  - rivelare la presenza di errori e chiedere la ritrasmissione dell'informazione (strategia ARQ);
  - rivelare la presenza di errori e correggerli (strategia FEC).

CODICI A RIPETIZIONE: ESEMPIO

Messaggio	Parola di codice	Singolo errore	Doppio errore	Triplo errore
0	000	(001 010 100)	(011 110 101)	111
1	111	(011 101 110)	(100 010 001)	000



CODICI A RIPETIZIONE: ESEMPIO

- Con n=3, nel caso in cui usiamo il codice per rivelare l'errore, siamo in grado di identificare 1 o 2 errori (errori singoli o doppi).

- Al contrario, non siamo in grado di rivelare errori tripli (3 errori trasformano una parola di codice in un'altra parola di codice).

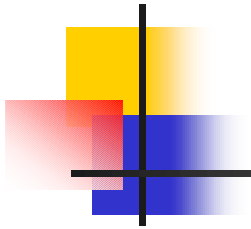
- Pertanto la probabilità P_{we} di non rivelare la presenza di errori in una parola ricevuta è data da:

$$P_{we} = P(3,3) = \alpha^3$$

CODICI A RIPETIZIONE: ESEMPIO

- Supponiamo adesso di usare lo stesso codice di prima per correggere errori sfruttando una regola di decisione di maggioranza: se una parola non appartiene al codice, la trasformiamo nella parola di codice ad essa più vicina assumendo che almeno $(n+1)/2$ bit siano corretti.
- Se assumiamo che 2 bit siano ricevuti correttamente (e 1 sia sbagliato) possiamo correggere l'errore.
- Se, invece, abbiamo 2 o 3 errori non è possibile effettuare la loro correzione.
- Pertanto, la probabilità P_{we} di sbagliare la correzione è data da:

$$P_{we} = P(2,3)+P(3,3) = 3\alpha^2-2\alpha^3$$



CODICI A CONTROLLO DI PARITÀ

- Un'altra tipologia di codici a blocchi (n,k) lineari e sistematici molto semplice è quella dei codici a controllo di parità.
- Nei codici a controllo di parità:
  - una parola è detta con parità pari (*even parity*) quando contiene un numero pari di 1;
  - una parola è detta con parità dispari (*odd parity*) quando contiene un numero dispari di 1.
- Le parole di un codice a controllo di parità $(n,n\text{-}1)$ contengono n-1 bit di informazione e 1 bit di controllo che viene scelto in modo da ottenere parole tutte caratterizzate dalla stessa parità.

CODICI A CONTROLLO DI PARITÀ: ESEMPIO ($n=3$, $k=2$)

Non rilevati perché hanno parità pari

Messaggio	Parola di codice	Singolo errore	Doppio errore	Triplo errore
00	000	(001 010 100)	(011 110 101)	111
01	011	(010 001 111)	(000 101 110)	100
10	101	(100 111 001)	(110 011 000)	010
11	110	(111 100 010)	(101 000 011)	001

Rilevati perché hanno parità dispari

CODICI A CONTROLLO DI PARITÀ

- Un codice a controllo di parità è in grado di rivelare un numero dispari di errori.

- La probabilità P_{we} di non rivelare la presenza di errori in una parola ricevuta è quindi data da:

$$P_{we} = \sum_{i\ pari}^{n} P(i,n) \cong P(2,n) = \binom{n}{2} \alpha^2 (1-\alpha)^{n-2} \cong \frac{n(n-1)}{2} \alpha^2$$

$\alpha << 1$

- Un codice a controllo di parità non ha capacità correttiva.

CODICI A CONTROLLO DI PARITÀ: ESEMPIO ($n=10$, $k=9$)

- Supponiamo di essere nel caso in cui n=10, k=9 e $\alpha=10^{-3}$.

- La probabilità P_{we} di non rivelare la presenza di errori in una parola ricevuta è data da:

$$P_{we} \cong \frac{n(n-1)}{2}\alpha^2 = 45 \cdot 10^{-6}$$

- Se non avessimo usato la codifica avremmo avuto una probabilità P_{uwe} di sbagliare una parola di messaggio pari a:

$$P_{uwe} = 1 - P(0, n-1) = 1 - \binom{n-1}{0}\alpha^0 (1-\alpha)^{n-1} \cong$$

$$= 1 - (1-\alpha)^{n-1} \cong 1 - [1-(n-1)\alpha] = (n-1)\alpha = 9 \cdot 10^{-3}$$

CODICE A BLOCCHI (n,k)

- Abbiamo visto 2 semplici esempi di codici a blocchi (n,k).
- In generale, dato un codice a blocchi (n,k):
  - Il codice è costituito da 2^k parole;
  - l'efficienza del codice vale:

$$R_c = k/n < 1$$

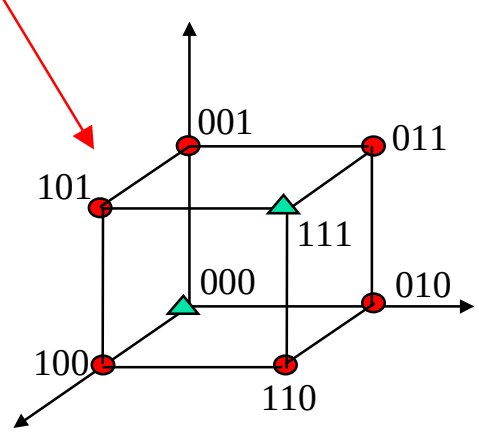
  - le proprietà del codice (capacità di correzione e di rivelazione) dipendono da come si scelgono le parole di codice e da come tali parole sono separate nello spazio ad n dimensioni ad esse associato.

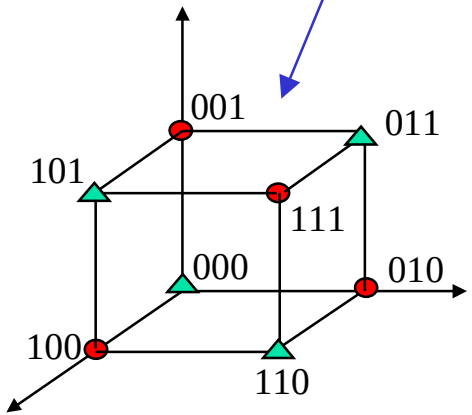
SPAZIO DEI CODICI

- Vediamo come vengono rappresentate le parole di codice ottenute con il codice a ripetizione (3,1) ed il codice a controllo di parità (3,2) nello spazio dei codici di dimensione n=3:

Parola di codice

Parola non appartenente al codice





- Nel primo caso le parole di codice sono maggiormente distanziate tra di loro ⇨ si possono rivelare un numero maggiore di errori.

DISTANZA DI HAMMING

- Per quantificare matematicamente il concetto della separazione tra le parole di codice, definiamo la distanza di Hamming e la distanza minima del codice.

- La distanza di Hamming $d(X,Y)$ tra due vettori binari X e Y di uguale lunghezza è il numero di bit differenti che compaiono nei due vettori in posizioni corrispondenti.

- Esempio:

$$X: 1\,0\,1$$
$$Y: 1\,1\,0$$
$$\Rightarrow d(X,Y) = 2$$

- La distanza minima del codice d_{min} è la minima distanza di Hamming fra tutte le possibili coppie di parole di codice.

CAPACITÀ RIVELATIVA DI UN CODICE

- Se si è verificato un numero di errori minore di d_{min}, la parola ricevuta non è una parola prevista dal codice e quindi è possibile rivelare gli errori.
- Se si è verificato un numero di errori maggiore o uguale a d_{min}, la parola ricevuta potrebbe corrispondere ad una parola di codice. In tal caso non sarebbe possibile rivelare gli errori.
- La regola da utilizzare per valutare la capacità rivelativa di un codice è la seguente:

Capacità rivelativa pari a ℓ errori per parola $\Leftrightarrow$ $d_{min} \geq \ell + 1$

CAPACITÀ CORRETTIVA DI UN CODICE

- Se si è verificato un numero di errori minore di $d_{min}/2$, possiamo correggere la parola ricevuta trasformandola nella parola di codice più vicina.
- Se si è verificato un numero di errori maggiore o uguale a $d_{min}/2$, non possiamo correggere la parola ricevuta (la parola di codice ad essa più vicina potrebbe non essere la parola che era stata trasmessa).
- La regola da utilizzare per valutare la capacità correttiva di un codice è la seguente:

Capacità correttiva pari a t errori per parola $\Leftrightarrow$ $d_{min} \geq 2t+1$

CAPACITÀ RIVELATIVA E CORRETTIVA DI UN CODICE

- ESEMPIO: Codice a ripetizione (3,1)
  - Parole di codice: $X_1=(000)$, $X_2=(111)$;
  - Distanze di Hamming: $d(X_1,X_2)=3$;
  - Distanza minima del codice: $d_{min}=3$;

$$d_{min} \geq \ell + 1 \Rightarrow \ell \leq 3 - 1 = 2$$

$$d_{min} \geq 2t + 1 \Rightarrow 2t \leq 3 - 1 = 2 \Rightarrow t \leq 1$$

- Quindi, come avevamo già osservato, un codice a ripetizione (3,1) può rivelare sino a 2 errori o correggere solo 1 errore.

CAPACITÀ RIVELATIVA E CORRETTIVA DI UN CODICE

- ESEMPIO: Codice a controllo di parità (3,2)
  - Parole di codice: $X_1=(000)$, $X_2=(011)$, $X_3=(101)$, $X_4=(110)$;
  - Distanze di Hamming:

    $d(X_1,X_2)=2$, $d(X_1,X_3)=2$, $d(X_1,X_4)=2$, $d(X_2,X_3)=2$, $d(X_2,X_4)=2$, $d(X_3,X_4)=2$;
  - Distanza minima del codice: $d_{min}=2$;

$$d_{min} \geq \ell + 1 \Rightarrow \ell \leq 2 - 1 = 1$$

$$d_{min} \geq 2t + 1 \Rightarrow 2t \leq 2 - 1 = 1 \Rightarrow t \leq 1/2$$

- Quindi, come avevamo già osservato, un codice a controllo di parità (3,2) può rivelare 1 errore, mentre non può essere utilizzato per la correzione di errori.

CAPACITÀ RIVELATIVA E CORRETTIVA DI UN CODICE

- La distanza minima di un codice (e quindi la sua capacità rivelativa e/o correttiva) dipende dal numero di bit di ridondanza che si aggiungono.

- In particolare, nel caso di un codice a blocchi (n,k) vale la seguente relazione:

$$d_{min} \leq n - k + 1$$

- Sfortunatamente, il limite superiore viene raggiunto solo nel caso di codici a ripetizione che però (avendo k=1) hanno un'efficienza molto bassa:

$$R_c = k/n = 1/n$$

PROPRIETÀ DI UN CODICE A BLOCCHI

- Abbiamo visto che uno dei passi fondamentali nella costruzione di un codice è scegliere le parole in modo tale da massimizzare d_{min}.

- Esistono altre proprietà che è bene che un codice a blocchi soddisfi al fine di facilitare le operazioni di codifica e decodifica. Tali proprietà sono:
  - linearità;
  - sistematicità.

CODICE LINEARE

- Consideriamo due generiche parole di codice rappresentabili come 2 vettori $X=(x_1\ x_2 \ldots x_n)$ e $Z=(z_1\ z_2 \ldots z_n)$, dove gli elementi x_i e z_i sono dei bit.

- Si definisce l'operazione di somma tra le due parole di codice come:

$$X+Z=(x_1 \oplus z_1\ \ x_2 \oplus z_2 \ldots x_n \oplus z_n)$$

- Un codice si definisce lineare se contiene il vettore nullo e se la somma tra due generiche parole ad esso appartenenti produce sempre un'altra parola di codice.

DISTANZA MINIMA IN UN CODICE LINEARE

- Definiamo peso $w(X)$ il numero di elementi non nulli del vettore X.
- Si può verificare facilmente che la distanza di Hamming tra due parole di codice risulta essere data da:

$$d(X,Z)=w(X+Z)$$

- Sfruttando la proprietà di linearità, si può dimostrare che:

$$d_{min}=[w(X)]_{min} \qquad X\neq(0\ 0\ \ldots.\ 0)$$

- La distanza minima di un codice lineare è uguale al più piccolo dei pesi dei vettori delle parole codice (escluso il vettore nullo).

CODICE SISTEMATICO

- Si definisce codice sistematico un codice a blocchi in cui i primi bit x_i $(i=1,\ldots,k)$ corrispondono ai bit del messaggio, mentre i bit di controllo x_j $(j=k+1,\ldots,n)$ sono raggruppati al termine della parola di codice.

- Una parola di un un codice sistematico può essere indicata come:

$$X=(\underbrace{m_1\ m_2\ \ldots\ m_k}\ \underbrace{c_1\ c_2\ \ldots\ c_{n-k}})=$$

$$=(\ M\ |\ C\ )$$

RAPPRESENTAZIONE MATRICIALE DEI CODICI SISTEMATICI LINEARI

- La notazione matriciale è molto utile in quanto permette di determinare le parole di codice per mezzo di una semplice moltiplicazione righe per colonna tra matrici.
- Infatti, dato un messaggio M, la parola di codice ad esso relativa può essere scritta come:

$$X=MG$$

- La matrice G (di dimensioni kxn) è detta matrice generatrice del codice ed ha la seguente struttura:

$$G \overset{def}{=} \left[I_k \mid P \right] = \left[\begin{array}{cccc|ccc} 1 & 0 & \dots & 0 & p_{1,1} & \dots & p_{1,n-k} \\ 0 & 1 & \dots & 0 & p_{2,1} & \dots & p_{2,n-k} \\ \vdots & & \ddots & & & \ddots & \\ 0 & 0 & \dots & 1 & p_{k,1} & \dots & p_{k,n-k} \end{array}\right]$$

RAPPRESENTAZIONE MATRICIALE DEI CODICI SISTEMATICI LINEARI

- La moltiplicazione matriciale righe per colonna segue le regole classiche, con l'eccezione che si usa una somma modulo 2 invece della somma convenzionale.
- Pertanto i bit di controllo sono calcolati come:

$$c_j = m_1 p_{1,j} \oplus m_2 p_{2,j} \oplus \ldots \oplus m_k p_{k,j}$$

OSSERVAZIONE

- Non esiste un metodo analitico per determinare gli elementi della sottomatrice P e per metterli in relazione con d_{min}.
- La scoperta di buoni codici è per lo più dovuta a ispirazione e perseveranza.

CODICE DI HAMMING

- Un codice di Hamming è un codice lineare a blocchi per il quale sono verificate le seguenti relazioni:

$$q=n-k\geq 3; \qquad n=2^q-1;$$

- La *code rate* è quindi pari a:

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

CODICE DI HAMMING

- È possibile verificare che, indipendentemente da q, la distanza minima è fissata e vale:

$$d_{min}=3$$

- Pertanto un codice di Hamming può rilevare due errori o correggere un errore per ogni parola di codice.

- Per costruire un codice di Hamming sistematico basta scegliere una sottomatrice P costituita da righe di q bit aventi tutte due o più bit a “1”.

CODICE DI HAMMING: ESEMPIO

- Consideriamo un codice di Hamming sistematico con q=3.
- La matrice generatrice G può, ad esempio, essere la seguente:

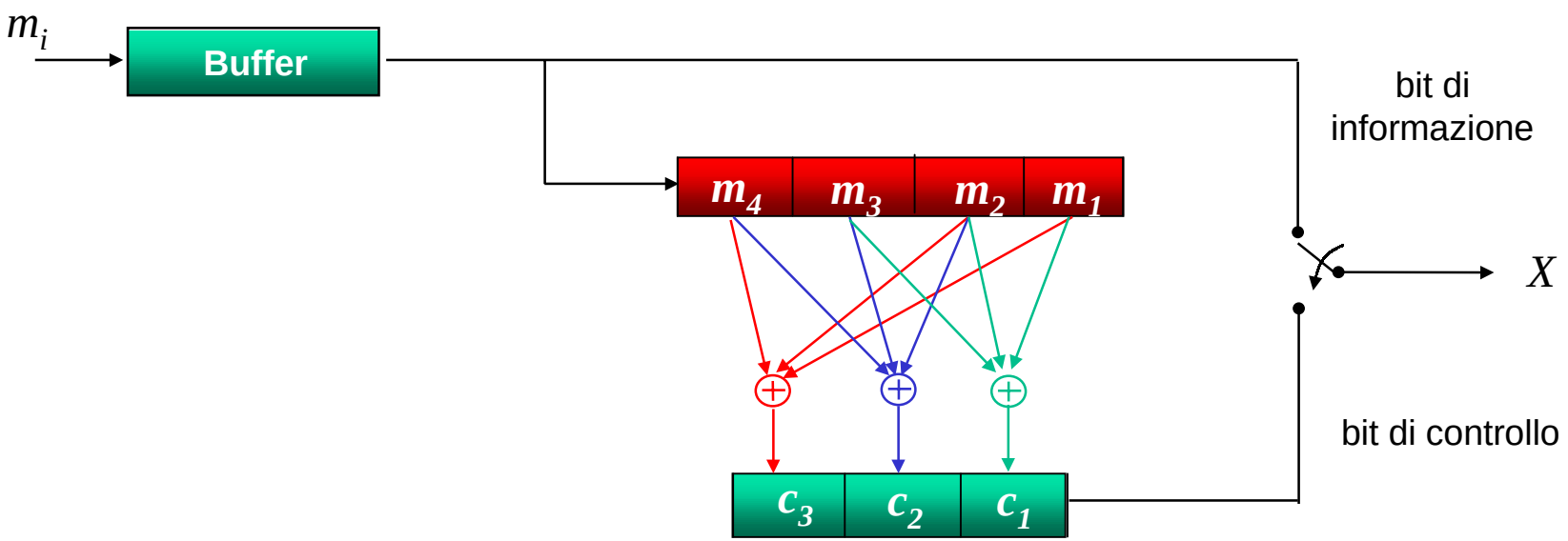
$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

- I bit di controllo risultano pertanto essere:

$$c_1 = m_1 \oplus m_2 \oplus m_3 \oplus 0$$
$$c_2 = 0 \oplus m_2 \oplus m_3 \oplus m_4$$
$$c_3 = m_1 \oplus m_2 \oplus 0 \oplus m_4$$

CODICE DI HAMMING: ESEMPIO

Il codificatore può quindi essere costruito nel modo seguente:



CODICE DI HAMMING: ESEMPIO

- Le 2^4 parole di codice risultanti sono riportate nella seguente tabella:

M	C	$w(X)$	M	C	$w(X)$
0 0 0 0	0 0 0	0	1 0 0 0	1 0 1	3
0 0 0 1	0 1 1	3	1 0 0 1	1 1 0	4
0 0 1 0	1 1 0	3	1 0 1 0	0 1 1	4
0 0 1 1	1 0 1	4	1 0 1 1	0 0 0	3
0 1 0 0	1 1 1	4	1 1 0 0	0 1 0	3
0 1 0 1	1 0 0	3	1 1 0 1	0 0 1	4
0 1 1 0	0 0 1	3	1 1 1 0	1 0 0	4
0 1 1 1	0 1 0	4	1 1 1 1	1 1 1	7

- Si può notare che il più piccolo peso $w(X)$ non nullo vale 3. Ciò a conferma che d_{min}=3.

DECODIFICA DI UN CODICE LINEARE SISTEMATICO A BLOCCHI

- Supponiamo di trasmettere la parola di codice X e di ricevere un vettore Y. Se sono avvenuti degli errori di trasmissione, $Y \neq X$.
- Un modo per effettuare la rivelazione degli errori è confrontare Y con ognuna delle parole di codice ⇒ è necessario memorizzare tutte le 2^k parole di codice.

ESEMPIO

- Se usiamo un codice di Hamming con q=5, si ottiene n=31 e k=26 ⇒ è necessario memorizzare 2^{26} parole!

DECODIFICA: MATRICE DI CONTROLLO DI PARITÀ

- Un metodo più efficiente per effettuare la decodifica e rivelare gli errori consiste nell'utilizzare la sottomatrice P.
- Si definisce la matrice di controllo di parità H come:

$$H \overset{def}{=} \left[P^T \mid I_q \right]$$

Matrice identità di dimensioni $q \times q$

- Tale matrice ha la seguente proprietà:

$$XH^T = (\,0\;0\;0\;\ldots\;0\,)$$

dove X è una parola di codice.

- Se Y non è una parola di codice, il prodotto YH^T contiene almeno un elemento non nullo.

SINDROME: RIVELAZIONE DI ERRORI

- La quantità $S=YH^T$ viene definita sindrome e consente di rivelare gli errori presenti nel vettore ricevuto Y.

- Se S è un vettore nullo, possiamo essere in uno tra 2 possibili casi:
  - non si sono verificati errori;
  - si sono verificati errori e Y è uguale ad una parola di codice differente da quella trasmessa (gli errori non sono rivelabili).

- Se la sindrome S contiene almeno un elemento non nullo si è in presenza di errori.

SINDROME: CORREZIONE DI ERRORI

- La correzione degli errori è più complessa, ma può ugualmente sfruttare le proprietà della sindrome S.
- Definiamo un vettore di errore E costituito da n bit. Tale vettore indica la posizione degli errori di trasmissione contenuti in Y.

ESEMPIO

$$\left.\begin{array}{l} X = (\,1\,0\,1\,1\,0\,) \\ Y = (\,1\,0\,0\,1\,1\,) \end{array}\right\} \Rightarrow E = (\,0\,0\,1\,0\,1\,)$$

DECODIFICA: SINDROME

- In generale si può scrivere:

$$Y = X+E$$

- Si può anche scrivere:

$$X = Y+E$$

- Inoltre, la sindrome S può essere riscritta come:

$$S=YH^T=(X+E)H^T = X\,H^T +E\,H^T= E\,H^T$$

La sindrome dipende quindi solo dagli errori occorsi e non dalla specifica parola di codice X trasmessa.

SINDROME: CORREZIONE DI ERRORI

PROBLEMA

- I possibili errori su una parola codificata con n bit sono 2^n-1.
- La sindrome, essendo costituita da n-k bit, può rappresentare solo 2^{n-k} -1 errori.

Ad ogni vettore di sindrome S corrispondono più vettori di errore E. Si deve quindi identificare l'errore che ha la maggiore probabilità di essere associato ad ogni specifico valore della sindrome.

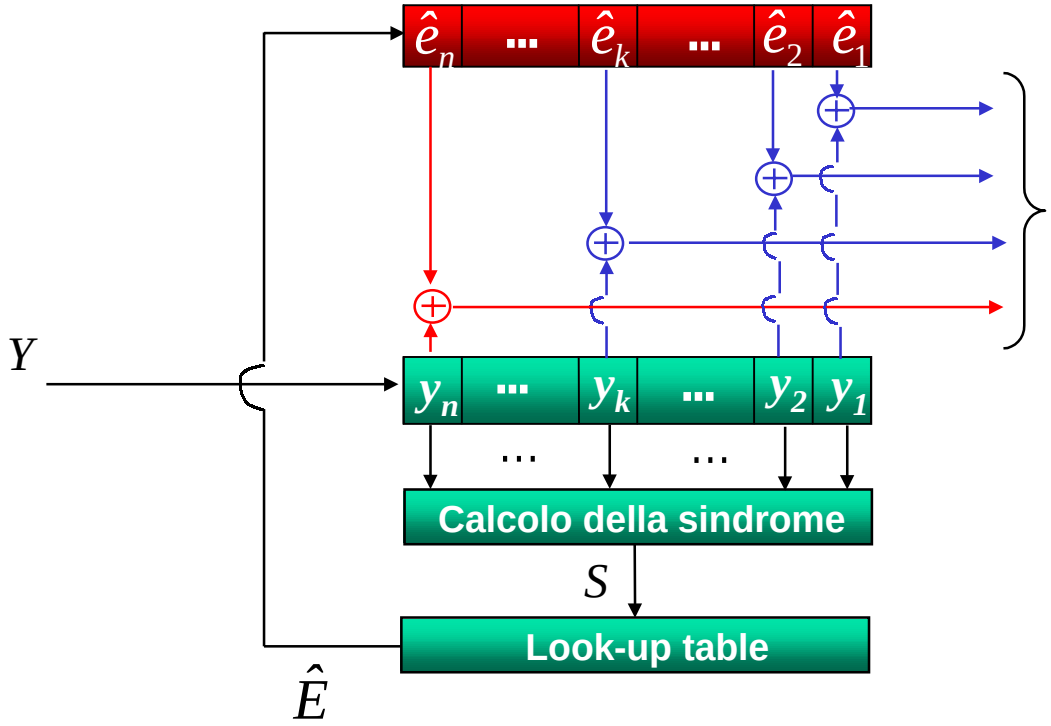
DECODIFICA A MASSIMA VEROSIMIGLIANZA

- La strategia nota con il nome di decodifica a massima verosimiglianza consiste nello scegliere, tra tutti i possibili vettori E che generano la stessa sindrome $S=EH^T$, il vettore errore $\hat{E}$ più probabile.

- Si può dimostrare che tale vettore è quello che contiene il numero minore di bit sbagliati. Questa scelta corrisponde a determinare la parola di codice $\hat{X}$ che ha la minore distanza di Hamming dal vettore ricevuto Y.

- La decodifica a massima verosimiglianza è ottima nel senso che minimizza la probabilità di errore per parola P_{we}.

DECODIFICA A MASSIMA VEROSIMIGLIANZA



$$Y + \hat{E} = \hat{X}$$

In realtà ci interessano solo i primi k bit in quanto sono quelli che contengono il messaggio

CODICI CICLICI

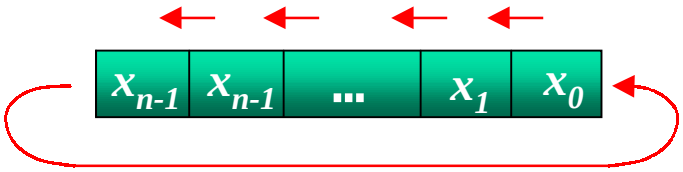
- È possibile dimostrare che, affinché un codice utilizzato per effettuare correzione di errori abbia capacità correttiva t≥1 e $R_c \cong 1$, è necessario avere $n>>1$, $k>>1$.
- Purtroppo però, in generale, la complessità hardware necessaria per codificare e decodificare parole di codice con elevata lunghezza è molto elevata.
- Al fine di superare questa limitazione, si può ricorrere all'impiego dei codici ciclici.
- I codici ciclici sono una sottoclasse dei codici lineari a blocchi la cui struttura consente una riduzione della complessità del codificatore e del decodificatore anche in presenza di parole di codice caratterizzate da elevata lunghezza.

CODICI CICLICI

- Consideriamo una generica parola di codice costituita da n bit:

$$X=(x_{n-1}\ x_{n-2}\ \ldots\ x_1\ x_0)$$

- Un codice è detto ciclico se:
  - soddisfa la proprietà di linearità;
  - ogni scorrimento ciclico dei bit della parola di codice X genera un'altra parola di codice



Ad esempio, $X'=(x_{n-2}\ \ldots\ x_1\ x_0\ x_{n-1})$ è una parola di codice.

CODICI CICLICI

- I codici ciclici possono essere trattati matematicamente associando alla parola di codice X un polinomio $X(p)$:

$$X(p) = x_{n-1}\, p^{n-1} + x_{n-2}\, p^{n-2} \ldots + x_1 p + x_0$$

con p variabile reale arbitraria.

- La potenza cui viene elevata p indica la posizione nella parola di codice del bit rappresentato dal coefficiente associato a p.

- Per quanto visto sopra, i codici ciclici vengono anche definiti codici polinomiali.

CODICI CICLICI: FORMULAZIONE MATEMATICA

- La formulazione matematica che fornisce le basi teoriche per la trattazione dei codici ciclici è costituita dai campi di Galois.
- Senza scendere nei dettagli, assumiamo quanto segue:
  - la somma di due polinomi è ottenuta mediante la somma modulo 2 dei rispettivi coefficienti;
  - poiché tali coefficienti possono assumere solo il valore “1” o “0” e $1 \oplus 1 = 0$ ⇨ l’operazione di sottrazione è uguale all’operazione di somma modulo 2:

$$X(p)+Z(p)=0 \quad \Rightarrow \quad X(p)=Z(p)$$

CODICI CICLICI: FORMULAZIONE MATEMATICA

- Vediamo che relazione esiste tra 2 parole di codice in un codice ciclico. Consideriamo i due polinomi seguenti:

$$pX(p) = x_{n-1}\,p^n + x_{n-2}\,p^{n-1} + \ldots + x_1\,p^2 + x_0\,p$$

$$X'(p) = x_{n-2}\,p^{n-1} + \ldots + x_1\,p^2 + x_0\,p + x_{n-1}$$

- Sommando $pX(p)$ e $X'(p)$ si ottiene:

$$pX(p)+X'(p) = x_{n-1}\,p^n + \underbrace{(x_{n-2} \oplus x_{n-2})}_{0}\,p^{n-1} + \ldots + \underbrace{(x_1 \oplus x_1)}_{0}\,p^2 + \underbrace{(x_0 \oplus x_0)}_{0}\,p + x_{n-1}$$

$$pX(p)+X'(p) = x_{n-1}\,p^n + x_{n-1} \quad \Rightarrow \quad X'(p) = pX(p) + x_{n-1}\,(p^n + 1)$$

CODICI CICLICI: POLINOMIO GENERATORE

- Effettuando scorrimenti multipli, si ottengono risultati analoghi a quello visto precedentemente per uno scorrimento singolo.
- Il polinomio $(p^n + 1)$ e il suo coefficiente x_{n-1} giocano un ruolo molto importante nei codici ciclici.
- In particolare, un codice ciclico (n,k) è definito per mezzo di un polinomio generatore $G(p)$ avente la seguente forma:

$$G(p) = p^q + g_{q-1} p^{q-1} + \ldots + g_1 p + 1$$

dove i coefficienti g_i sono tali che $G(p)$ sia un fattore del polinomio $(p^n + 1)$.

CODICI CICLICI: POLINOMIO GENERATORE

- Si può dimostrare che ogni parola di codice può essere scritta come:

$$X(p) = Q_M(p) \cdot G(p)$$

dove $Q_M(p)$ è il polinomio che rappresenta un blocco di k bit di messaggio.

- Ogni fattore di $(p^n + 1)$ che ha grado q può essere utilizzato come polinomio generatore, ma non è detto che fornisca un buon codice.

- Tra i codici polinomiali più usati citiamo i codici ciclici di Hamming, il codice di Golay e i codici di Bose, Chaudhuri e Hocquenghem (BCH).

CODICI CICLICI: ESEMPI

- Nella tabella seguente sono riportati i polinomi generatori di alcuni codici ciclici tra i più usati.

Codice	n	k	R_c	d_{min}	$G(p)$
Hamming	7	4	0.57	3	1 011
	15	11	0.73	3	10 011
	31	26	0.84	3	100 101
BCH	15	7	0.46	5	111 010 001
	31	21	0.68	5	11 101 101 001
	63	45	0.71	7	1 111 000 001 011 001 111
Golay	23	12	0.52	7	101 011 100 011

CODICI CICLICI SISTEMATICI E NON SISTEMATICI

- In generale, un codice ciclico può essere sistematico o non sistematico. Ciò dipende dalla struttura del termine $Q_M(p)$.

- Nel seguito incentreremo la nostra attenzione sui codici ciclici sistematici.

CODICI CICLICI SISTEMATICI

- Consideriamo un generico codice ciclico sistematico e definiamo il polinomio $M(p)$ dei bit di informazione e il polinomio $C(p)$ dei bit di controllo:

$$M(p) = m_{k-1}\, p^{k-1} + \ldots + m_1 p + m_0$$

$$C(p) = c_{q-1}\, p^{q-1} + \ldots + c_1 p + c_0$$

- Affinché il codice sia sistematico, la parola di codice $X(p)$ relativa al messaggio $M(p)$ deve essere espressa come:

$$X(p) = p^q M(p) + C(p)$$

CODICI CICLICI SISTEMATICI

- Per quanto visto prima, deve valere la relazione:

$$X(p) = Q_M(p) \cdot G(p) = p^q M(p) + C(p)$$

$$\frac{p^q M(p)}{G(p)} = Q_M(p) + \frac{C(p)}{G(p)}$$

- Il polinomio dei bit di controllo può quindi essere calcolato come il resto della divisione di $p^q M(p)$ con $G(p)$:

$$C(p) = \left[p^q M(p) \right] \bmod \left[G(p) \right]$$

Funzione che restituisce il resto della divisione

CODICI CICLICI: SINDROME

- Anche per i codici ciclici si può effettuare la decodifica utilizzando la sindrome.
- Il calcolo della sindrome $S(p)$ in ricezione è molto semplice. Infatti si può scrivere:

$$S(p) = [Y(p)] \, mod \, [G(p)]$$

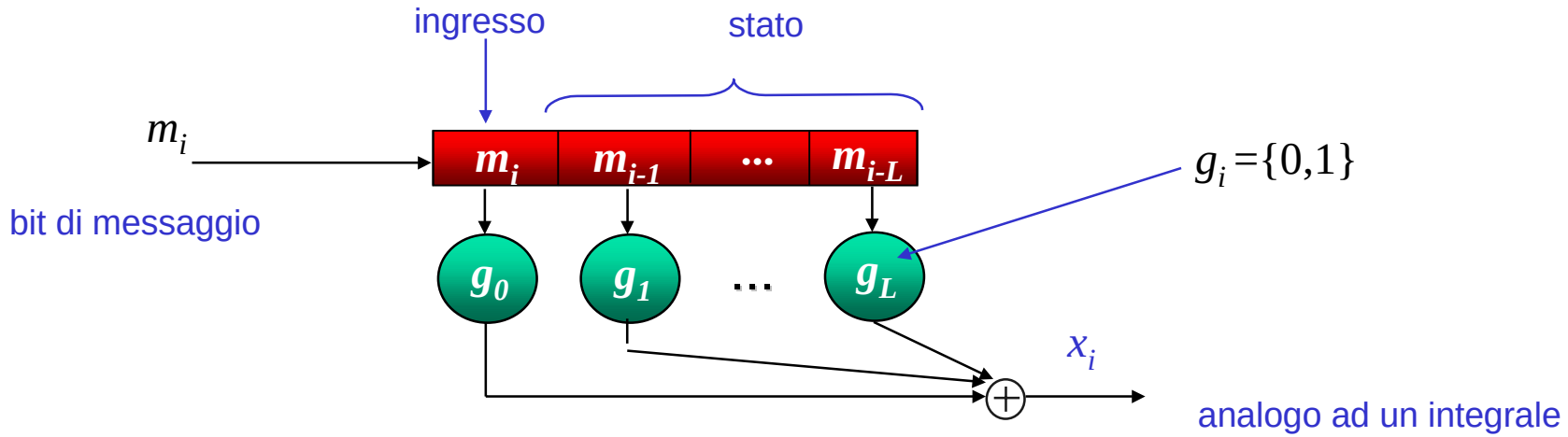
- Se il polinomio $Y(p)$ associato al vettore ricevuto è una parola di codice, allora $S(p) = 0$ in quanto $G(p)$ è un fattore di $Y(p)$. Se invece $S(p) \neq 0$, significa che il vettore ricevuto è corrotto dalla presenza di errori.

CODICI CICLICI: VANTAGGI

- I principali vantaggi dei codici ciclici sono:
  - semplicità di codifica;
  - semplicità nel calcolo della sindrome;
  - struttura matematica ben definita che permette lo sviluppo di metodi di decodifica molto efficienti. Tali metodi riducono drasticamente la necessità di memorizzare informazione e quindi rendono possibile l'impiego di codici con $n>>1$, $k>>1$ ⇨ $R_c \cong 1$.

CODICI CONVOLUZIONALI

- I codici convoluzionali, al contrario di quelli a blocchi, non generano sequenze organizzate in parole di codice ben definite.
- Per capire il principio di funzionamento dei codici convoluzionali analizziamo lo schema seguente:



$$x_i = m_{i-L}\, g_L \oplus \ldots \oplus m_{i-1}\, g_1 \oplus m_i\, g_0 = \sum_{j=0}^{L} m_{i-j}\, g_j$$

CODICI CONVOLUZIONALI

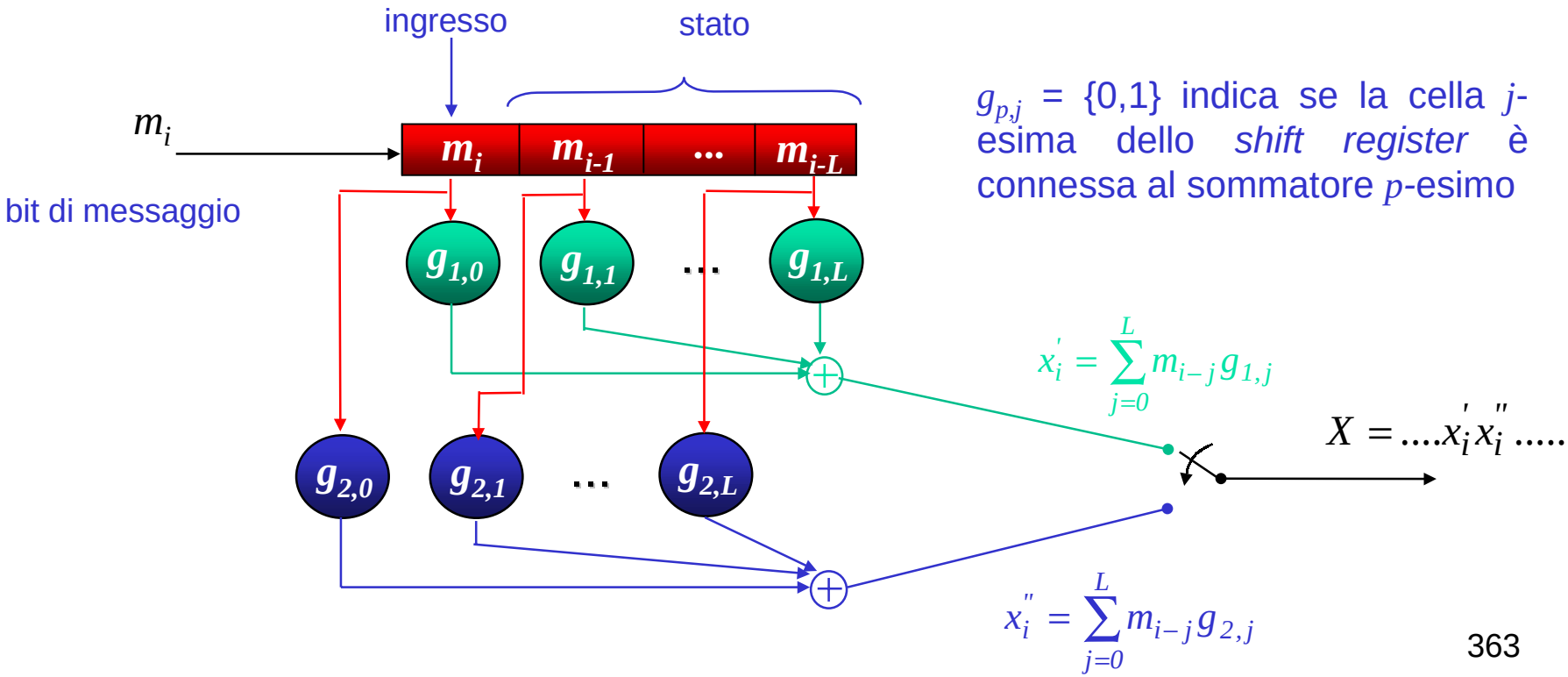
- Il bit codificato x_i dipende dall'ingresso m_i e dallo stato del registro (ovvero dagli L bit di messaggio precedenti).

- Il bit di messaggio m_i influenza i successivi $L+1$ bit.

- Il meccanismo fin qui illustrato permette di trasformare una sequenza di bit in una sequenza differente, ma non aggiunge ridondanza. Come si può utilizzare quanto visto per introdurre ridondanza?

CODICI CONVOLUZIONALI

- Per aggiungere i bit di ridondanza necessari al controllo degli errori, è necessario utilizzare due o più sommatori modulo 2:



CODICI CONVOLUZIONALI

- In generale, se si usano n sommatori modulo 2, si ottiene una *code rate* $R_c=1/n$.
- Il bit di messaggio m_i influenza i successivi $n(L+1)$ bit. La quantità $n(L+1)$ viene definita *constraint length.*
- Si definisce invece memoria la quantità L (ovvero la lunghezza dello *shift register*).
- Analogamente a quanto fatto per i codici a blocchi, tipicamente si parla di codici convoluzionali (n,k,L).

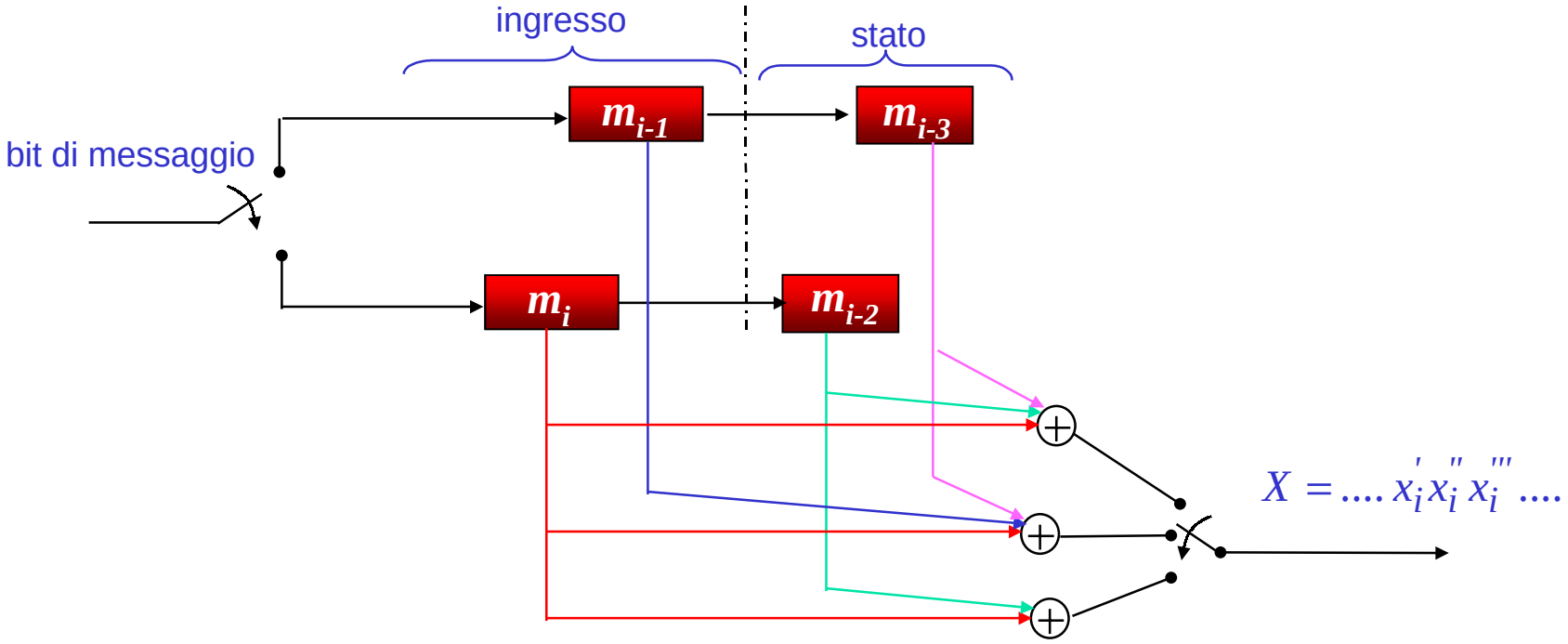
CODICI CONVOLUZIONALI

- Per ottenere una *code rate* $R_c>1/n$ è necessario utilizzare $k\geq 2$ *shift register.* In particolare utilizzando *k shift register* è possibile ottenere una *code rate* $R_c=k/n$.

- Nella *slide* successiva è riportato un esempio di come si può costruire un codificatore per ottenere un codice convoluzionale caratterizzato da $R_c=2/3$.

CODICI CONVOLUZIONALI: ESEMPIO

- Codificatore per codice convoluzionale (3,2,1) (ovvero caratterizzato da e $n=3$, $k=2$ e $L=1$).



CODICI CONVOLUZIONALI: OSSERVAZIONI

- Per ottenere una code rate $R_c > 1/n$, in alternativa ad utilizzare k registri, è possibile utilizzare un solo registro di lunghezza $kL+1$ e fare scorrere i bit a gruppi di k.

- I codici convoluzionali tipicamente usati in sistemi FEC reali sono caratterizzati da valori di k e di n piccoli e da una *constraint length* compresa tra 10 e 30.

- È possibile ottenere codici convoluzionali in forma sistematica, ovvero caratterizzati dal seguente comportamento:

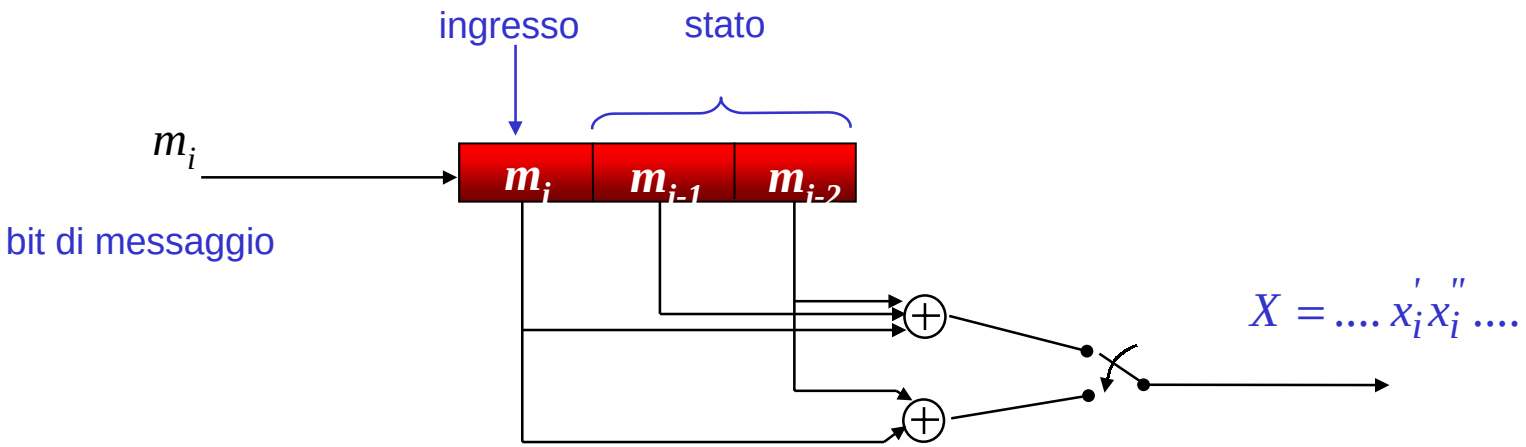
$$X = m_1 x_1'' x_2''' \ldots\ldots\ldots m_i x_i'' x_i''' \ldots\ldots$$

RAPPRESENTAZIONI GRAFICHE DEI CODICI CONVOLUZIONALI

- Per studiare i codici convoluzionali si possono usare tre diversi tipi di rappresentazioni:
  - rappresentazione ad albero;
  - rappresentazione a traliccio;
  - diagramma degli stati.

CODICI CONVOLUZIONALI: ESEMPIO

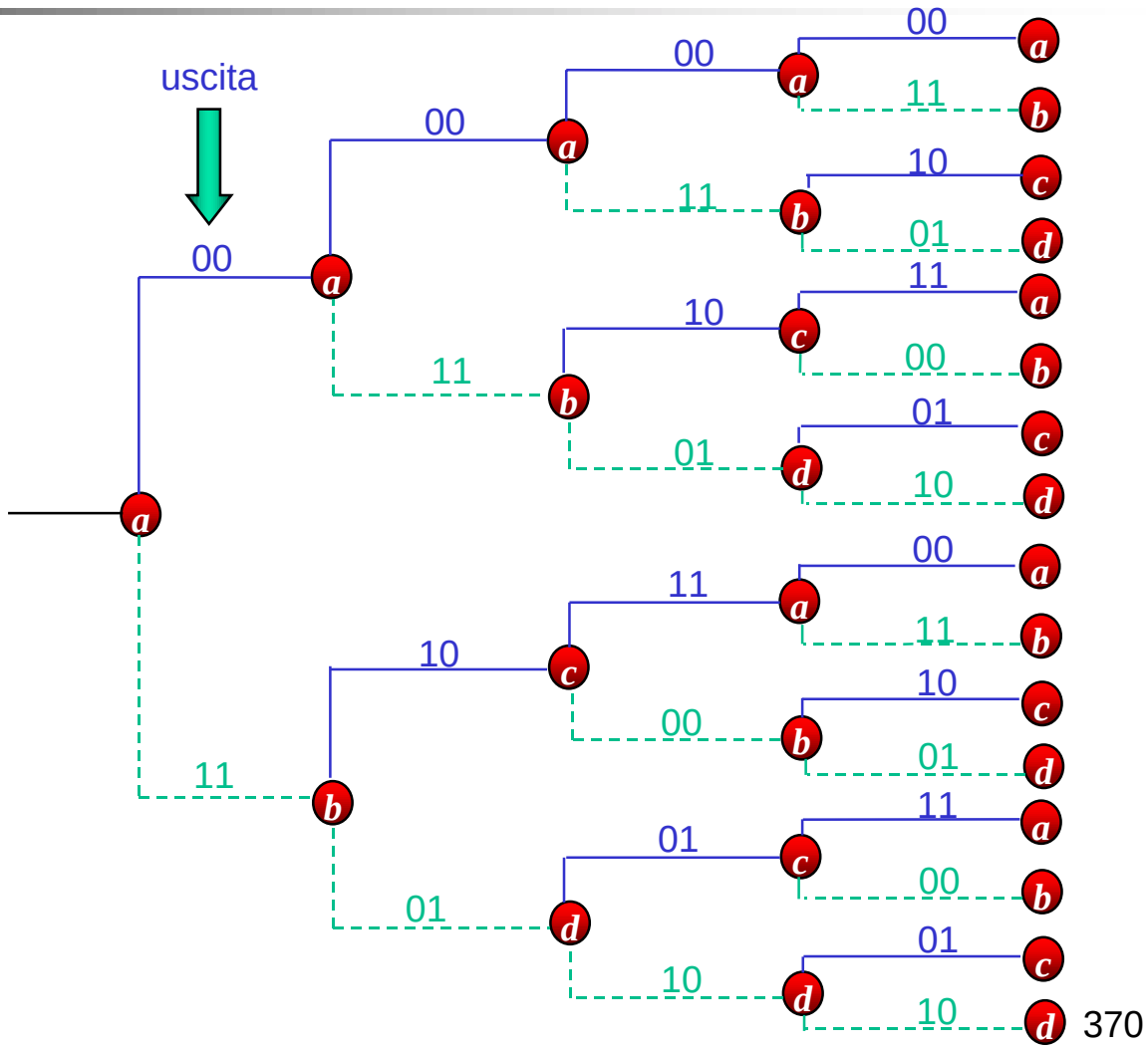
- Consideriamo il codice convoluzionale *(2,1,2)* (ovvero avente n=2, k=1, L=2) ottenuto con il seguente codificatore:



- Supponiamo che il registro contenga tutti “0” quando arriva il primo bit di messaggio m_1.

ALBERO DEL CODICE: ESEMPIO

Stati
a=00
b=10
c=01
d=11

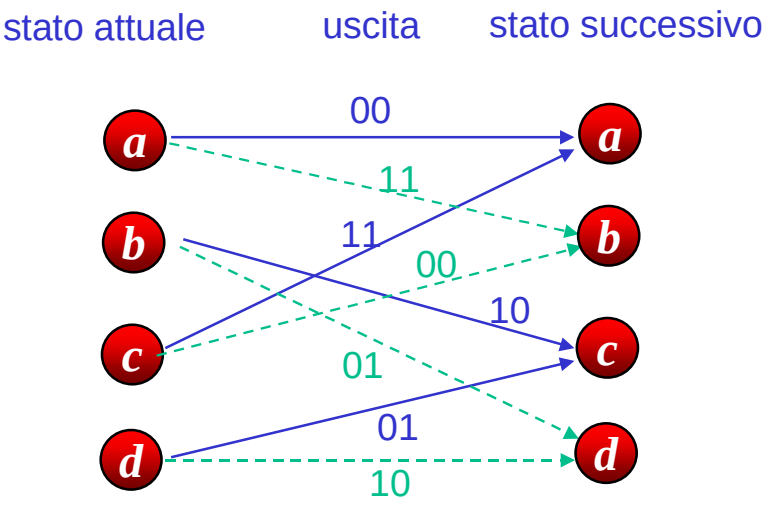


transizione di stato che si ha in conseguenza di un ingresso $m_i=0$

transizione di stato che si ha in conseguenza di un ingresso $m_i=1$

TRALICCIO DEL CODICE: ESEMPIO

Stati
$a=00$
$b=10$
$c=01$
$d=11$

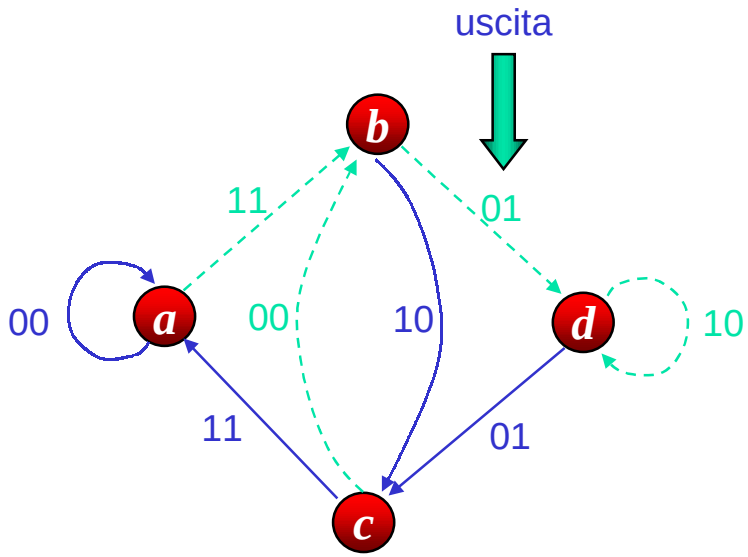


transizione di stato che si ha in conseguenza di un ingresso $m_i=0$

transizione di stato che si ha in conseguenza di un ingresso $m_i=1$

DIAGRAMMA DEGLI STATI: ESEMPIO

Stati
a=00
b=10
c=01
d=11

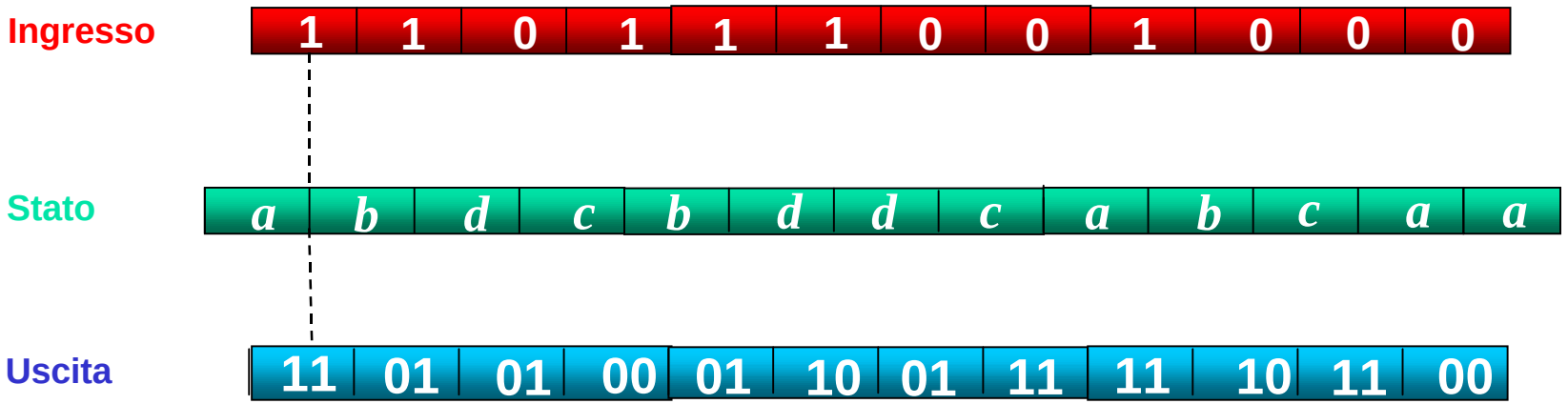


transizione di stato che si ha in conseguenza ad un ingresso $m_i=0$

transizione di stato che si ha in conseguenza ad un ingresso $m_i=1$

CODICI CONVOLUZIONALI: ESEMPIO

- Data una sequenza in ingresso e lo stato iniziale, è possibile ottenere la sequenza dei bit in uscita dal codificatore mediante l'analisi del traliccio o del diagramma degli stati.



CODICI CONVOLUZIONALI: DISTANZA LIBERA

- Abbiamo visto che la capacità correttiva e/o rivelativa di un codice a blocchi dipende dalla distanza minima del codice.
- Nei codici a blocchi la distanza minima è determinata analizzando i pesi delle parole di codice.
- Nel caso di un codice convoluzionale, non essendo individuabili delle parole di codice, si considera il peso dell'intera sequenza X generata da una determinata sequenza di messaggi.
- Si definisce distanza libera di un codice convoluzionale la quantità:

$$d_l=[w(X)]_{min} \qquad X\neq(0\ 0\ 0\ \ldots.)$$

CODICI CONVOLUZIONALI: DISTANZA LIBERA

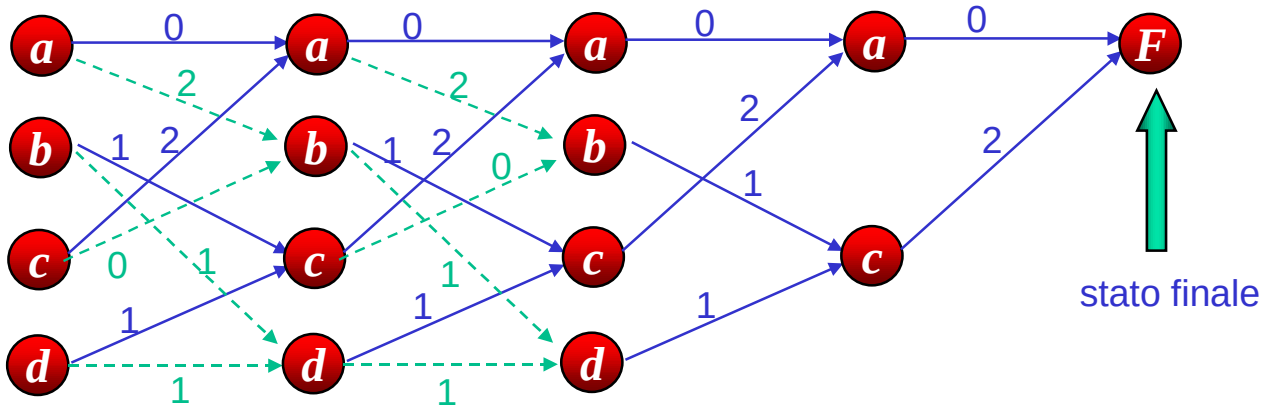
- Il calcolo di d_l può essere effettuato facilmente senza dover analizzare il peso w di tutte le possibili sequenze $w(X)$.

- Si deve identificare il percorso a peso minimo che si "stacca" dal percorso tutto nullo per riportarsi in esso in modo "stabile".

- Supponiamo di terminare la sequenza con una serie di "0" in modo da riportare lo *shift register* al suo stato iniziale. Tale procedura elimina alcuni rami dalle L transizioni finali del traliccio del codice.

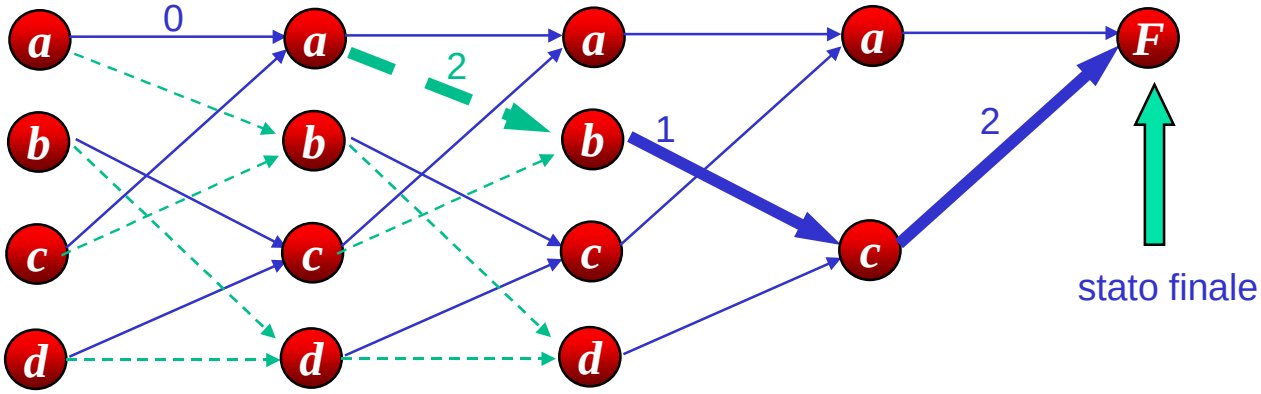
CODICI CONVOLUZIONALI: DISTANZA LIBERA

- Consideriamo il traliccio dell'esempio precedente ed associamo ad ogni ramo il numero di "1" che compaiono nella sequenza codificata corrispondente:



CODICI CONVOLUZIONALI: DISTANZA LIBERA

- Consideriamo il percorso a peso minimo (non tutto nullo) che parte dallo stato a ed arriva allo stato finale F:



- Prendiamo ora in esame la sequenza di ingressi X che permette di ottenere la sequenza di stati $aaaa\ldots abcF$.

- Il peso di tale sequenza è $w(X)=5$.

CODICE CONVOLUZIONALE: DISTANZA LIBERA

- Si può facilmente osservare che non esistono percorsi corrispondenti a sequenze di ingresso X non identicamente nulle con un peso $w(X)$ minore di quello del percorso $aaaa\ldots abcF$.

- Pertanto, nel caso in esame si può affermare che $d_l=5$.

- La strategia descritta permette quindi di calcolare facilmente d_l senza dover analizzare tutti i possibili percorsi X.

DISTANZA LIBERA E FUNZIONE GENERATRICE

- Un altro approccio per il calcolo della distanza libera d_l consiste nel considerare la funzione generatrice di un codice convoluzionale.

- Tale funzione può essere vista come la funzione di trasferimento del codificatore rispetto alle transizioni degli stati (mette quindi in relazione stati iniziali e stati finali).

- La funzione generatrice fornisce importanti indicazioni sulla distanza libera d_l e sulla probabilità di errore che si può ottenere in decodifica (per maggiori dettagli sulla funzione generatrice fare riferimento al *Carlson*).

CODICI CONVOLUZIONALI: METODI DI DECODIFICA

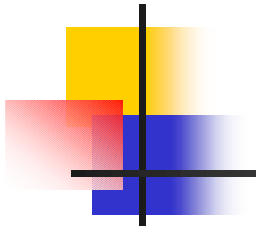
- Esistono sostanzialmente tre metodi per la decodifica di codici convoluzionali:
  - Algoritmo di Viterbi (effettua una decodifica ottima a massima verosimiglianza, ma richiede *hardware* molto complesso);
  - Decodifica Sequenziale (può realizzare compromessi differenti tra complessità *hardware* e bontà della decodifica);
  - Decodifica a Retroazione (richiede *hardware* molto semplice, ma degrada le prestazioni della decodifica).

ALGORITMO DI VITERBI

- Un decodificatore ottimo a massima verosimiglianza deve esaminare l'intera sequenza ricevuta Y e trovare il percorso "lecito" che ha la minore distanza di Hamming da Y.

- Se il messaggio trasmesso è costituito da N bit (ovvero Y è costituito da Nn/k bit) ⇒ si hanno 2^N possibili percorsi.

- Nei casi reali, essendo tipicamente N>>1, un confronto esaustivo di Y con tutti i possibili percorsi è proibitivo.



ALGORITMO DI VITERBI

- L'algoritmo di Viterbi, indipendentemente da N, consente di limitare il confronto a 2^{kL} *percorsi superstiti*, rendendo possibile l'applicazione della decodifica ottima a massima verosimiglianza.

- Ad ogni ramo di ogni percorso viene assegnata una metrica che indica la sua distanza di Hamming dal corrispondente ramo di Y.

- La metrica di ogni percorso è ottenuta sommando i valori della metrica dei rami che lo compongono.

- La sequenza Y è decodificata come il percorso a metrica minima.

ALGORITMO DI VITERBI: ESEMPIO

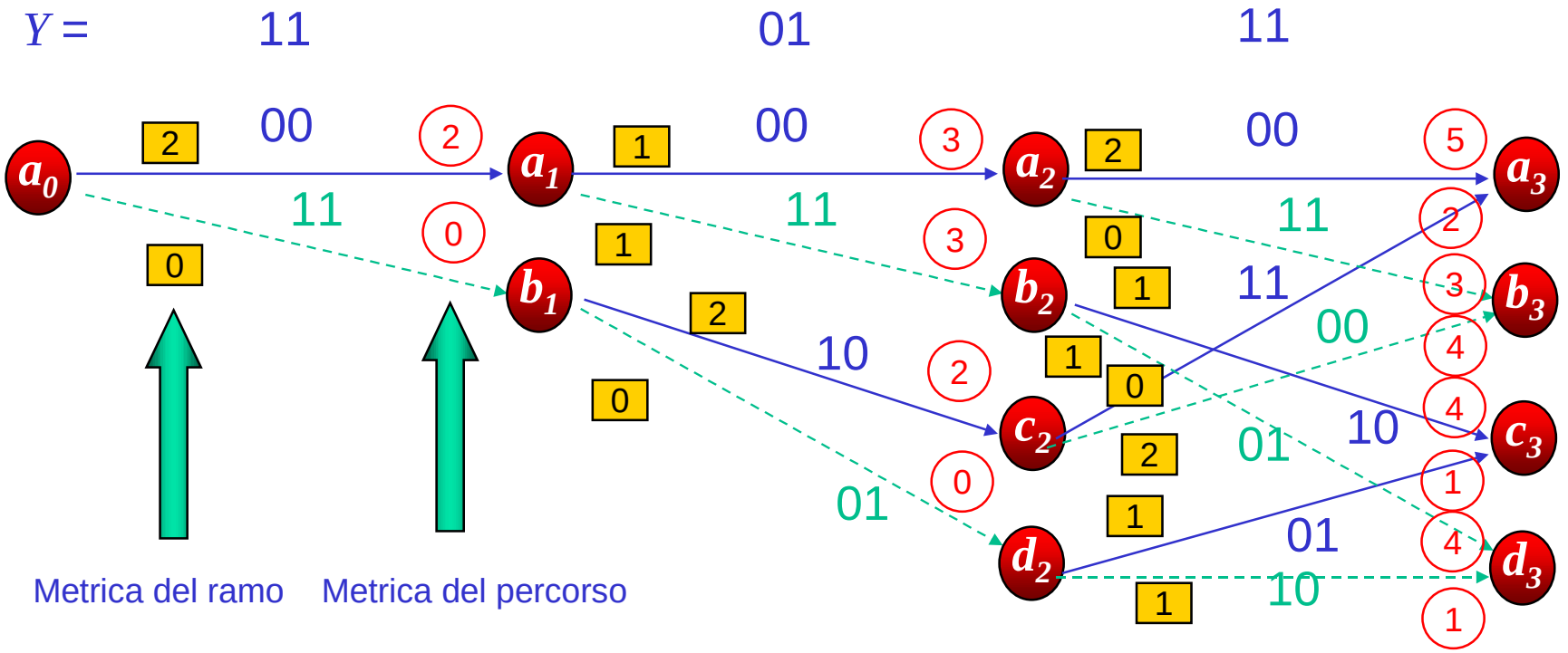
- Studiamo il problema della decodifica di Viterbi con un esempio.

- Consideriamo il decodificatore associato al codice convoluzionale $n=2$, $k=1$, $L=2$ visto negli esempi precedenti e supponiamo di ricevere la sequenza Y=11 01 11.

- Analizziamo il traliccio del codice per capire come si assegna la metrica ai diversi rami, come si possono selezionare i *percorsi superstiti* e come si applica l'algoritmo di Viterbi.

ALGORITMO DI VITERBI: ESEMPIO

- Iniziamo l'analisi dal livello $j=L+1=3$ (j è una variabile), ovvero consideriamo le prime 3 transizioni partendo dal nodo iniziale a_0:



ALGORITMO DI VITERBI: ESEMPIO

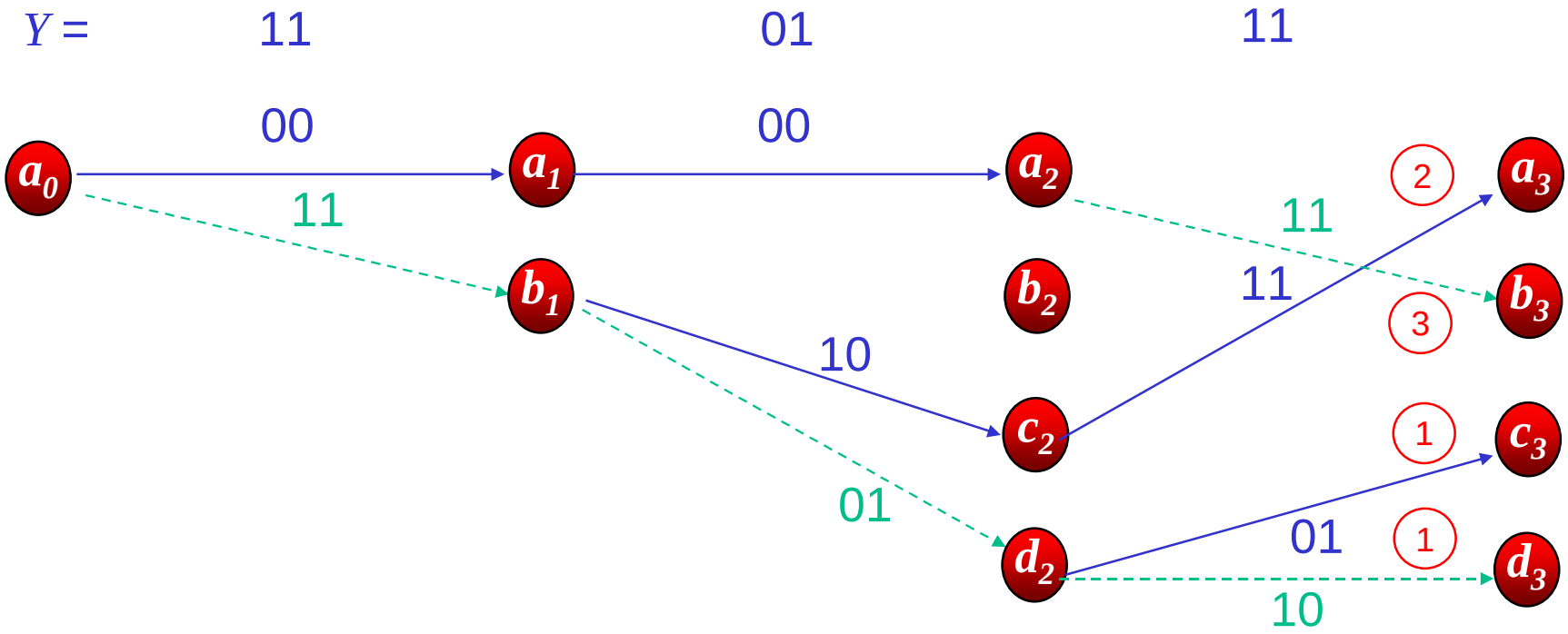
- Osserviamo che ogni stato che si può avere alla terza transizione (ovvero o a_3 o b_3 o c_3 o d_3) può essere raggiunto con 2 percorsi differenti.

- Ad esempio, possiamo raggiungere b_3 passando da a_0,a_1,a_2,b_3 o da a_0,b_1,c_2,b_3. Indipendentemente da cosa verrà trasmesso successivamente, il percorso che include il tratto a_0,a_1,a_2,b_3 ha una metrica minore ed è quindi più probabile che rappresenti la sequenza trasmessa X.

- Ragionando in questo modo, per ognuno dei nodi a_3 ,b_3 ,c_3 ,d_3 che si hanno alla terza transizione possiamo eliminare il percorso con la metrica maggiore, riducendo così il numero di percorsi da memorizzare.

ALGORITMO DI VITERBI: ESEMPIO

- Nel nostro esempio si ottengono $2^{kL}=4$ percorsi superstiti di lunghezza $j=L+1=3$:



ALGORITMO DI VITERBI: ESEMPIO

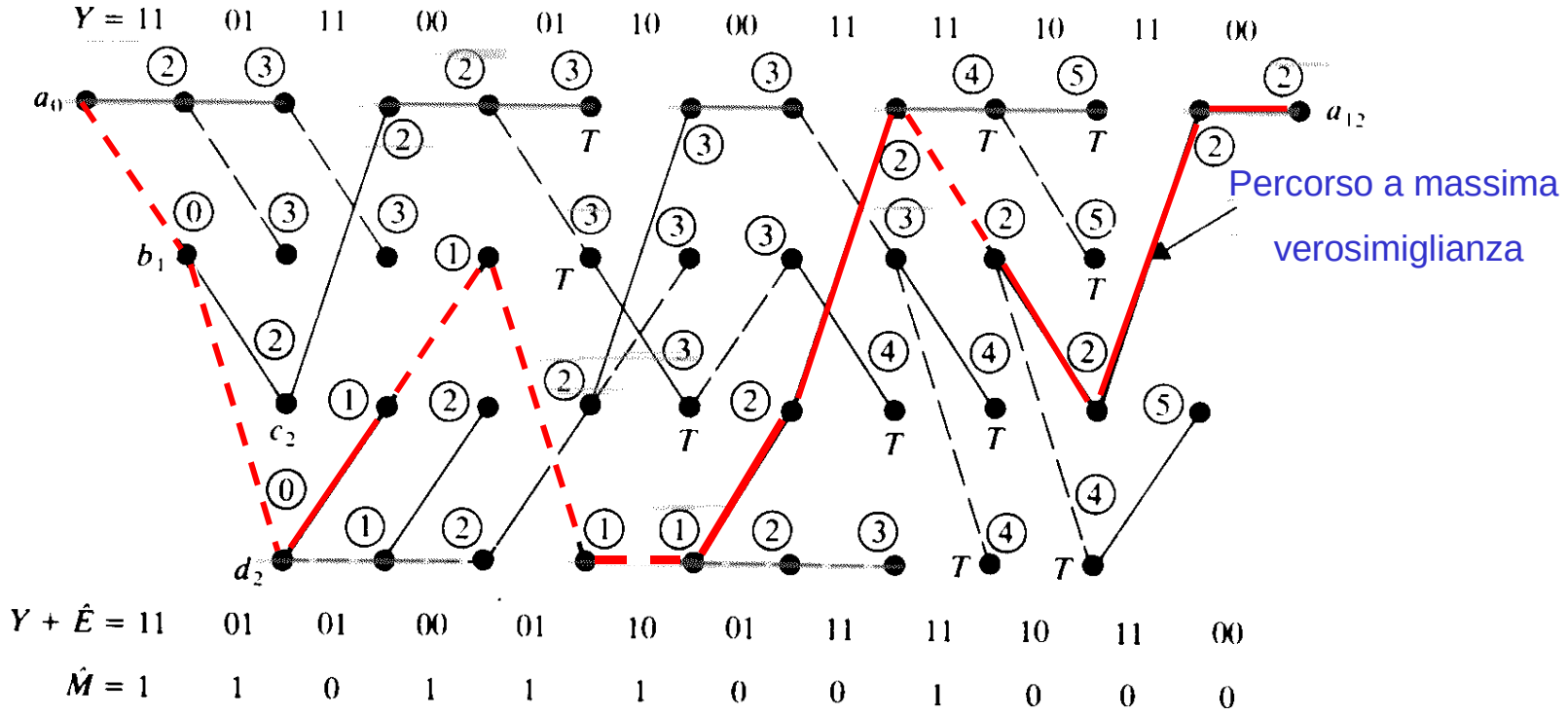
- A questo punto, l'algoritmo di Viterbi prevede di incrementare iterativamente j ($j=j+1$) e di ripetere l'applicazione dello stesso tipo di procedimento ai nuovi percorsi (tutti costituiti da un ramo in più).

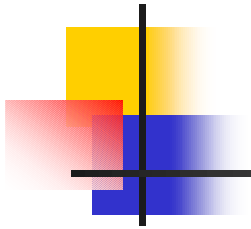
- In questo modo si arriva alla generazione di un nuovo insieme di $2^{kL}=4$ percorsi superstiti (ciascuno caratterizzato da lunghezza j).

- Il processo iterativo termina quando si raggiunge l'ultimo bit della sequenza ricevuta.

ALGORITMO DI VITERBI: ESEMPIO

- Vediamo cosa accade nel nostro esempio nel caso di trasmissione di N=12 bit:





ALGORITMO DI VITERBI

- Con l'algoritmo di Viterbi è necessario calcolare 2 metriche per ogni nodo e memorizzare 2^{kL} percorsi superstiti (ognuno costituito da N rami).

- Per quanto detto, la complessità dell'algoritmo di Viterbi cresce esponenzialmente con L e linearmente con N. Questo limita l'applicazione di tale algoritmo a codici con valori di L piccoli.

ALGORITMO DI VITERBI: EFFETTO DI DIVERGENZA DELLA METRICA

- Quando $N>>1$ si può adottare una strategia semplificata che consente di ridurre la quantità di memoria richiesta dall'algoritmo.

- Questa strategia è basata sull'osservazione che, se da un nodo partono due percorsi superstiti, la metrica del percorso meno probabile tende ad aumentare molto più rapidamente di quella dell'altro percorso superstite. Tale effetto diventa evidente dopo circa $5L$ rami.

ALGORITMO DI VITERBI: EFFETTO DI DIVERGENZA DELLA METRICA

- Dopo che sono stati ricevuti i primi $5Ln$ bit, i primi k bit di messaggio possono essere decodificati (e conseguentemente i primi insiemi di rami possono essere cancellati dalla memoria).

- Gruppi successivi di k bit sono decodificati dopo ogni gruppo ulteriore di n bit ricevuti.

- Quindi, la decodifica semplificata non richiede di dover aspettare la fine della sequenza trasmessa.

DECODIFICA SEQUENZIALE

- Abbiamo detto che l'applicazione dell'algoritmo di Viterbi è limitata a codici con valore di L piccolo.

- Problema: valore di L piccolo ⇨ capacità correttiva molto limitata.

- Possibile soluzione: incrementare L ed utilizzare una tecnica di decodifica non ottima, ma computazionalmente molto semplice

⇩

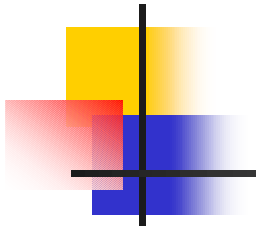
decodifica sequenziale

DECODIFICA SEQUENZIALE

- La decodifica sequenziale si basa sull'effetto di divergenza della metrica.

- Partendo dallo stato iniziale a_0, un decodificatore sequenziale intraprende il percorso che al nodo successivo presenta la metrica minore (se 2 percorsi hanno la stessa metrica si sceglie in modo casuale).

- Se il decodificatore intraprende un percorso che si rivela essere "troppo" improbabile, in pochi rami la metrica di tale percorso tende a crescere molto velocemente. In questi casi, il decodificatore può "decidere" di tornare indietro e tentare un altro percorso (*backtracking*).

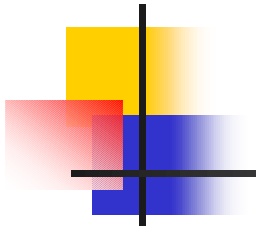


DECODIFICA SEQUENZIALE

- La decisione di *backtracking* è basata sul valore atteso della metrica ad un certo nodo.

- Si può infatti affermare che il valore atteso della metrica al nodo *j*-esimo è pari a jnP_{be} (con P_{be} probabilità di errore sul bit). Questo valore coincide con il numero atteso di bit sbagliati nei primi jn bit della sequenza Y.

- Il decodificatore esegue l'operazione di *backtracking* se la metrica di un percorso supera di una determinata soglia Δ la quantità jnP_{be} .

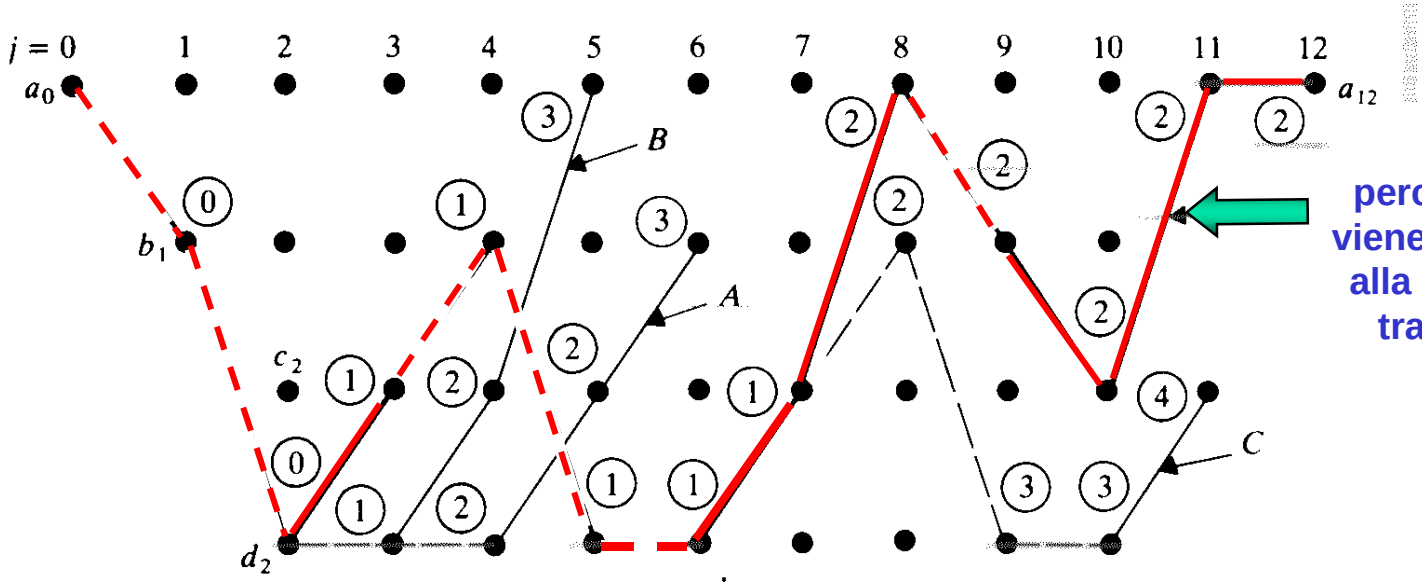


DECODIFICA SEQUENZIALE

- Se Δ è piccolo ⇨ il decodificatore tende ad analizzare tutti i possibili percorsi e le prestazioni si avvicinano a quelle di un decodificatore a massima verosimiglianza (tuttavia, in questo caso, i frequenti *backtracking* rendono la decodifica più complessa di quella dell'algoritmo di Viterbi).

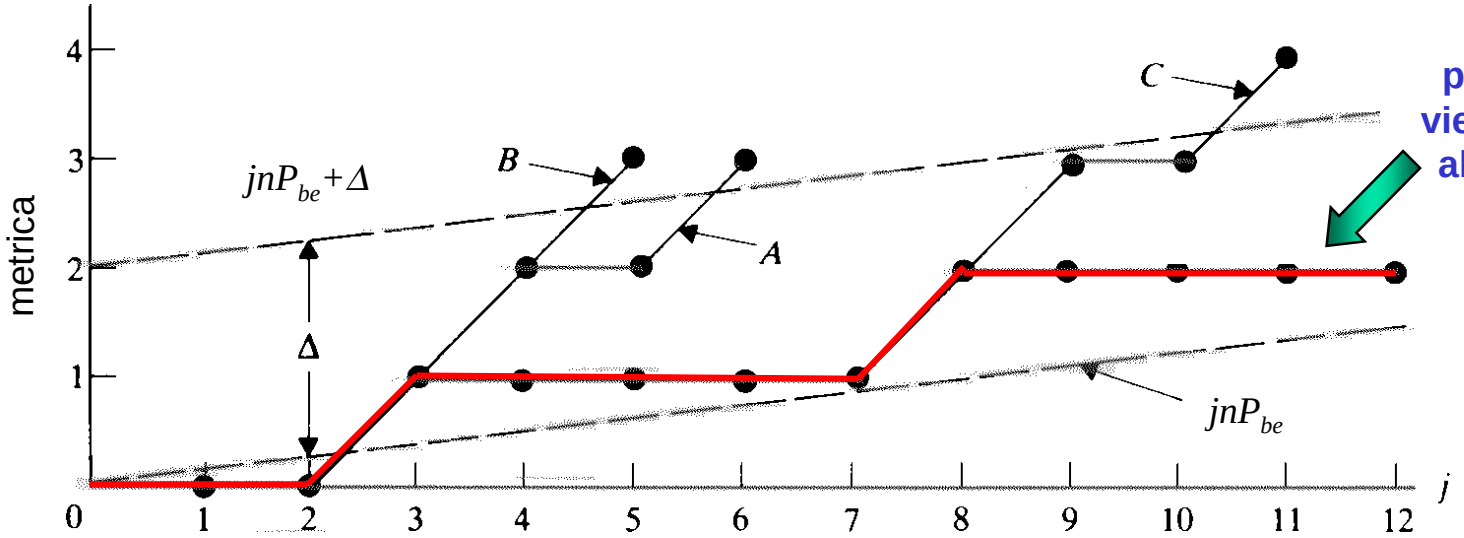
- Se Δ è grande ⇨ si riducono i tempi e la complessità di decodifica, ma aumenta la probabilità di sbagliare.

DECODIFICA SEQUENZIALE: ESEMPIO

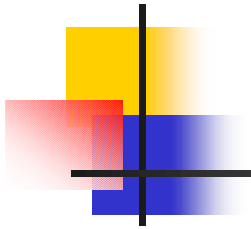


percorso che viene associato alla sequenza trasmessa

DECODIFICA SEQUENZIALE: ESEMPIO



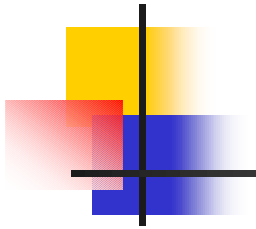
percorso che viene associato alla sequenza trasmessa



DECODIFICA A RETROAZIONE

- Storicamente, la decodifica sequenziale e quella di Viterbi sono state implementate in forma di *software* per computer o microprocessori. Solo recentemente sono sono state realizzate implementazioni in forma di dispositivi *hardware* VLSI.

- Quando si vogliono realizzare decodificatori *hardware* molto semplici, un'alternativa è usare decodificatori a retroazione.



DECODIFICA A RETROAZIONE

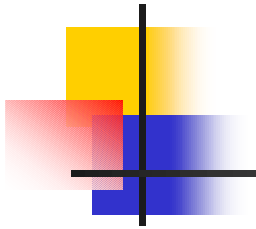
- I decodificatori a retroazione sono sostanzialmente dei decodificatori a blocchi che operano su finestra mobile e che decodificano un bit alla volta basandosi su un blocco di L rami successivi.

- Questi tipi di decodificatori forniscono prestazioni molto inferiori a quelle degli altri due metodi e possono condurre a probabilità di errore di decodifica piuttosto elevate.

CODICI CONVOLUZIONALI: VANTAGGI/SVANTAGGI

- In generale l'operazione di codifica risulta molto semplice, mentre in decodifica si possono ottenere prestazioni elevate solo per mezzo di algoritmi molto sofisticati.

- I codici convoluzionali sono particolarmente indicati nei sistemi di trasmissione via satellite. In tali sistemi, infatti, è indispensabile l'utilizzo di codificatori semplici (vengono montati a bordo dei satelliti), mentre è accettabile l'impiego di decodificatori complessi (la decodifica viene effettuata presso il "segmento di terra" che tipicamente comprende potenti calcolatori).



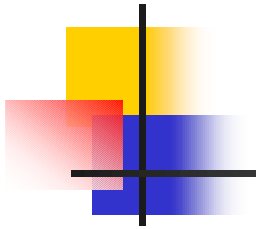
ESEMPI DI APPLICAZIONE DEI CODICI

- La scelta della strategia di codifica (FEC/ARQ) e del tipo di codice dipende dall'applicazione selezionata.

- Consideriamo, ad esempio, alcuni casi importanti:
  - codifica per canali AWGN;
  - codifica per canali "compound-error";
  - codifica per dispositivi di memorizzazione.

CODIFICA PER CANALI AWGN

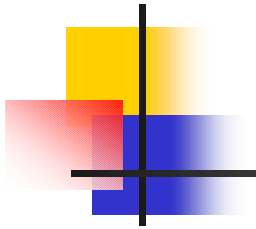
- Quando si considerano canali che possono essere modellati con un rumore AWGN (ciò avviene ad esempio nelle comunicazioni radio via satellite e nelle comunicazioni nello spazio profondo) è particolarmente consigliato l'impiego di strategie FEC.
- A seconda del guadagno di codifica che si vuole ottenere, si possono scegliere tipi di codici differenti:
  - Guadagno di codifica moderato ⇨ codici convoluzionali con piccole constraint-length e decodifica di Viterbi semplificata.
  - Guadagno di codifica elevato ⇨ codici convoluzionali con elevate constraint-length e decodifica sequenziale.



CODIFICA PER CANALI "COMPOUND-ERROR"

- I canali compound-error sono caratterizzati dalla presenza "mista" di errori indipendenti ed errori di tipo *burst* (sequenze continue di errori). Esempi di tale tipo di canali sono:
  - i canali telefonici (gli errori di tipo *burst* si verificano a causa di rumore impulsivo);
  - i canali radio (gli errori di tipo *burst* si verificano a causa di rumore atmosferico, interferenze, ecc.)

- In questi casi, generalmente si utilizzano strategie ARQ con l'impiego di codici a blocchi lineari ciclici.



CODIFICA PER DISPOSITIVI DI MEMORIZZAZIONE

- Questo tipo di applicazioni riguardano l'impiego dei codici per controllo di errori in memorie di computer, nastri magnetici digitali, dischi magnetici, ecc.

- In questi casi, si possono utilizzare sia strategie ARQ sia strategie FEC.

- Tipicamente vengono impiegati codici a blocchi lineari (sono molto utilizzati i codici di Hamming).