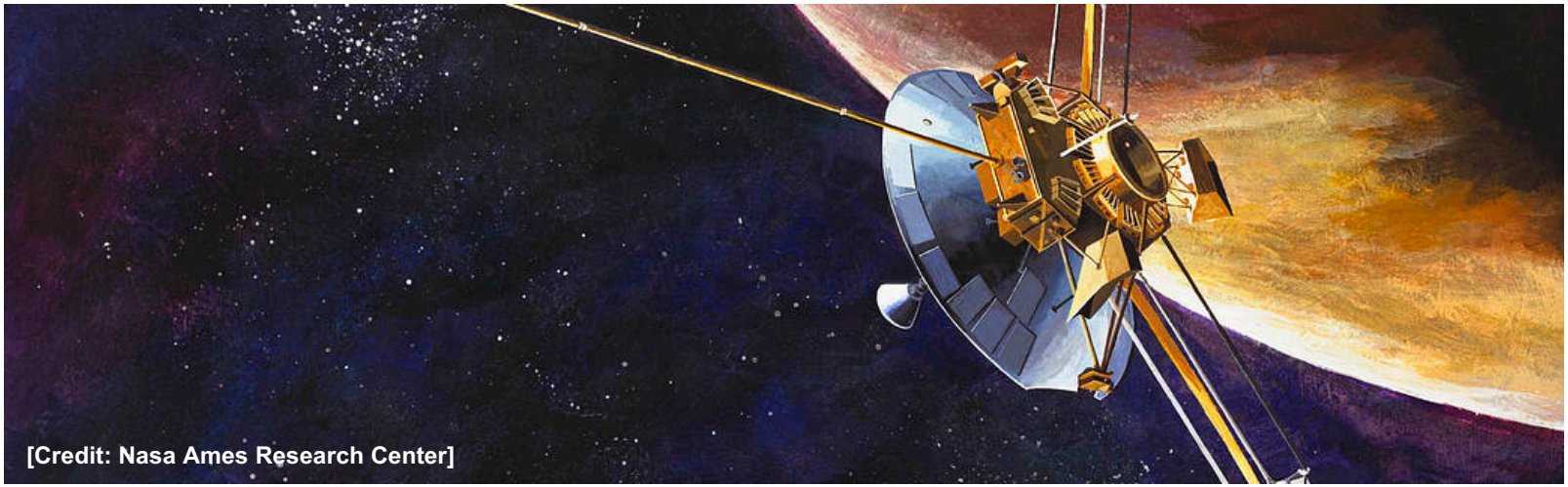
Codici convoluzionali

Introduzione ai codici convoluzionali

Poiché sono molto potenti in termini di capacità di correzione di errore e, dall'introduzione dell'Algoritmo di Viterbi, hanno complessità limitata, i codici convoluzionali sono una famiglia di codici lineari utilizzati in moltissime applicazioni, tra cui:

- WiFi (802.11) e reti cellulari 2G, 3G, 4G e 5G.
- Tv digitale (DVB-T).
- Deep space communications.



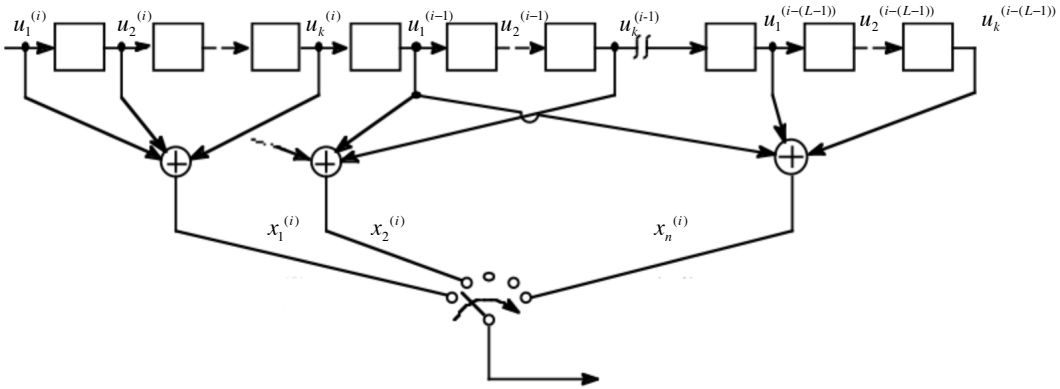
Introduzione ai codici convoluzionali

- Diversamente dai codici a blocco in forma sistematica, i *codici convoluzionali* sono in generale non-sistematici.
- In un codice convoluzionale, il codificatore trasmette solo i bit di parità.
- Ad ogni tempo di bit, il codificatore combina i bit all'interno di una finestra mobile di lunghezza L (*constraint length* del codice) per generare gli n bit in uscita.
- Il processo di codifica può essere interpretato come una convoluzione in GF(2) ed il codificatore di un codice convoluzionale (n, k, L) si può rappresentare come n filtri lineari in GF(2) in parallelo.

Introduzione ai codici convoluzionali

Un *codificatore convoluzionale* $\mathcal{C}(n, k, L)$ è costituito da uno shift register e da n sommatori.

- Ad ogni periodo di clock (definito dall'intero i) nel codificatore entrano k (tipicamente $k = 1$) cifre binarie rappresentate dal vettore $\mathbf{u}^{(i)} = [u_1^{(i)}, u_2^{(i)}, \ldots, u_k^{(i)}]$ ed escono n bit, raccolti nel vettore di uscita $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]$.
- All'interno del codificatore, $\mathbf{u}^{(i)}$ è combinato con i precedenti $L - 1$ vettori di ingresso $\mathbf{u}^{(i-1)}$,$\mathbf{u}^{(i-2)}, \ldots, \mathbf{u}^{(i-L+1)}$ per formare l'uscita $\mathbf{x}^{(i)}$.



Introduzione ai codici convoluzionali

La principale differenza rispetto ai codici a blocco è costituita dal fatto che l'uscita attuale non dipende solo dalla parola attuale ma anche dalla storia passata contenuta nelle $L-1$ parole precedenti.

- Il numero L di parole che contribuiscono all'uscita attuale è la *constraint length* del codice.
- Il numero complessivo di celle di memoria del codificatore è $kL-1$: $(L-1)\times k$ per memorizzare la memoria del sistema più $k-1$ per memorizzare i bit (tutti eccetto l'ultimo) della parola attuale.

I generatori di un codice convoluzionale

Dato un codice convoluzionale $\mathcal{C}(n, k, L)$, i generatori definiscono completamente il codice, così come la matrice generatrice o il polinomio generatore definiscono i codici a blocco e i codici ciclici.

- A ciascuna delle n uscite corrisponde un generatore e a ciascun generatore corrisponde un sommatore in GF(2).
- Ciascun generatore ha kL elementi, uno per ciascun ingresso che contibuisce all'uscita del codificatore.
- Quando un elemento è 1 significa che il corrispondente ingresso è connesso al sommatore, altrimenti se è 0 non c'è connessione.

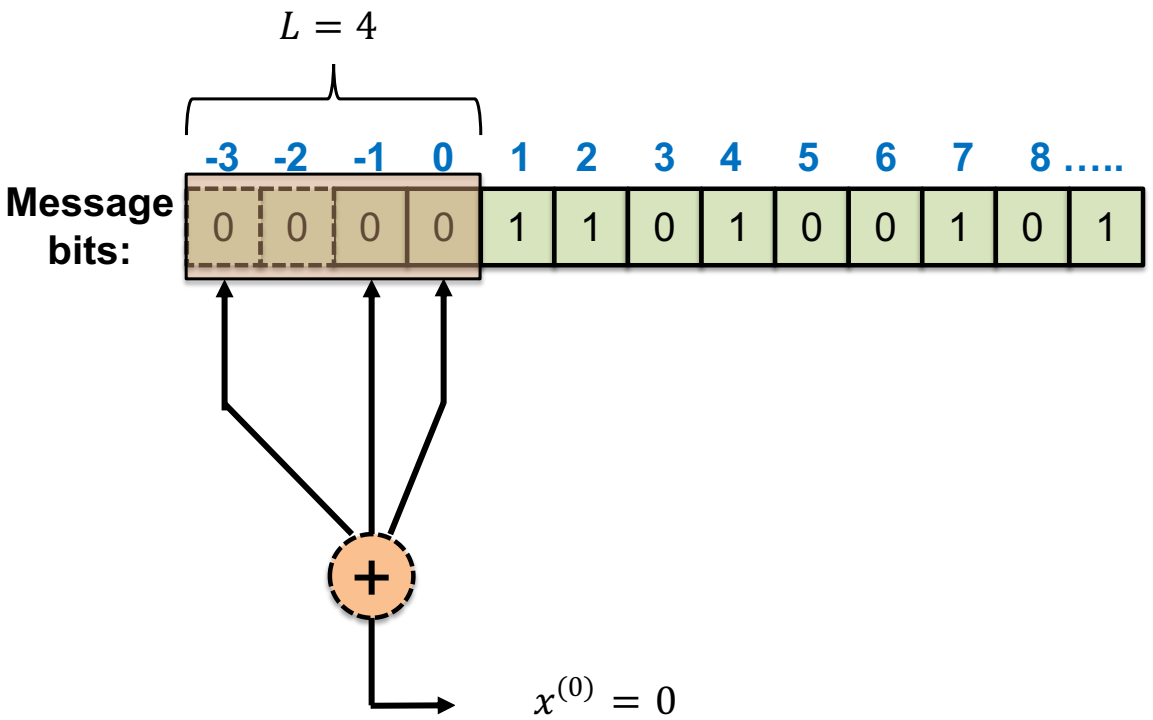
I generatori di un codice convoluzionale

- I vettori generatori rappresentano la risposta impulsiva degli n filtri lineari in GF(2).
- Assumendo che sia $k = 1$, il generatore per il bit m-esimo è $\mathbf{g}_m = [g_m^{(0)}, g_m^{(1)}, \ldots, g_m^{(L-1)}]$ e l'm-esimo bit di uscita $x_m^{(i)}$ si ottiene così

$$x_m^{(i)} = \sum_{\ell=0}^{L-1} g_m^{(\ell)} u^{(i-\ell)}.$$

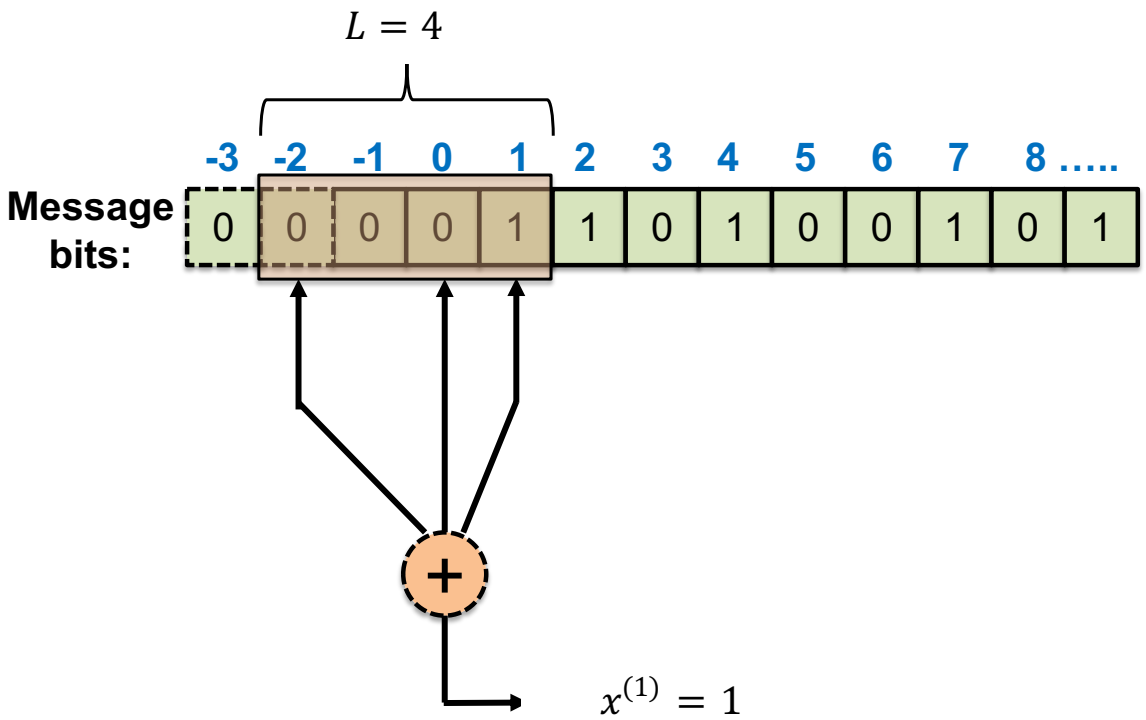
- L'uscita m-esima è la convoluzione discreta in GF(2) tra i vettori $[\mathbf{u}^{(i)}, \mathbf{u}^{(i-1)}, \ldots, \mathbf{u}^{(i-L+1)}]$ e $\mathbf{g}_m$, da cui il nome di *codice convoluzionale*.

Esempio di generatore $\mathbf{g} = [1101]$



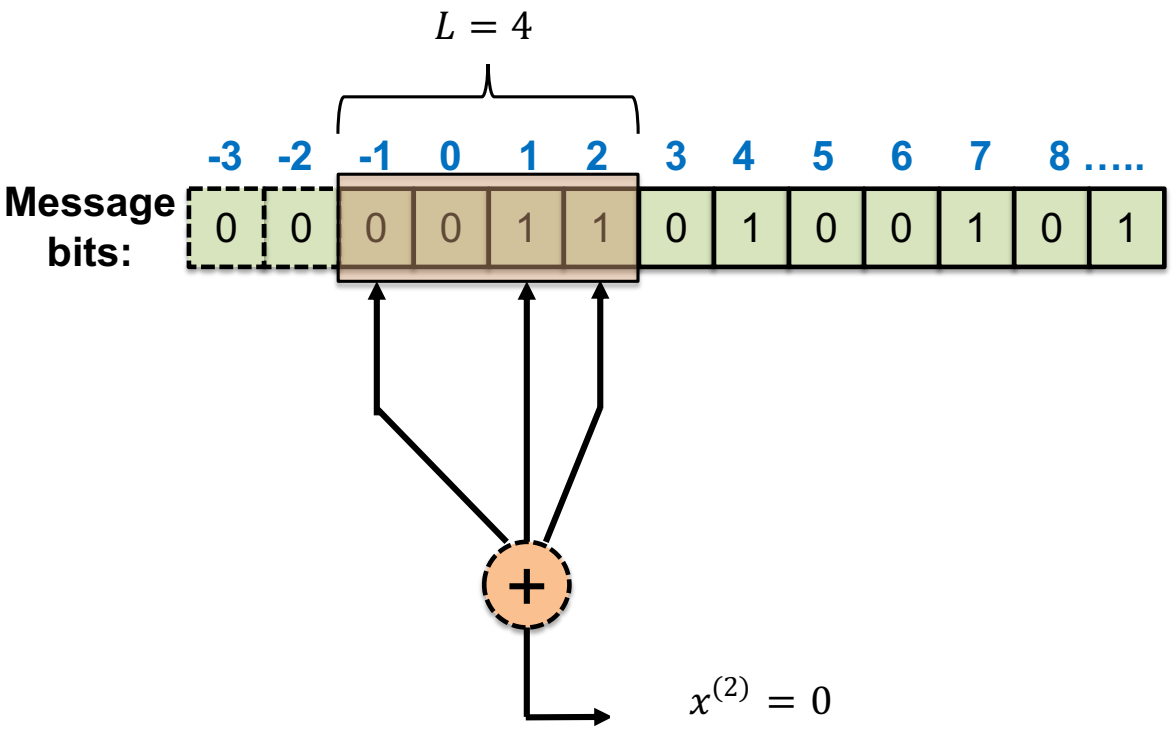
- **Output: 0**

Esempio di generatore $\mathbf{g} = [1101]$



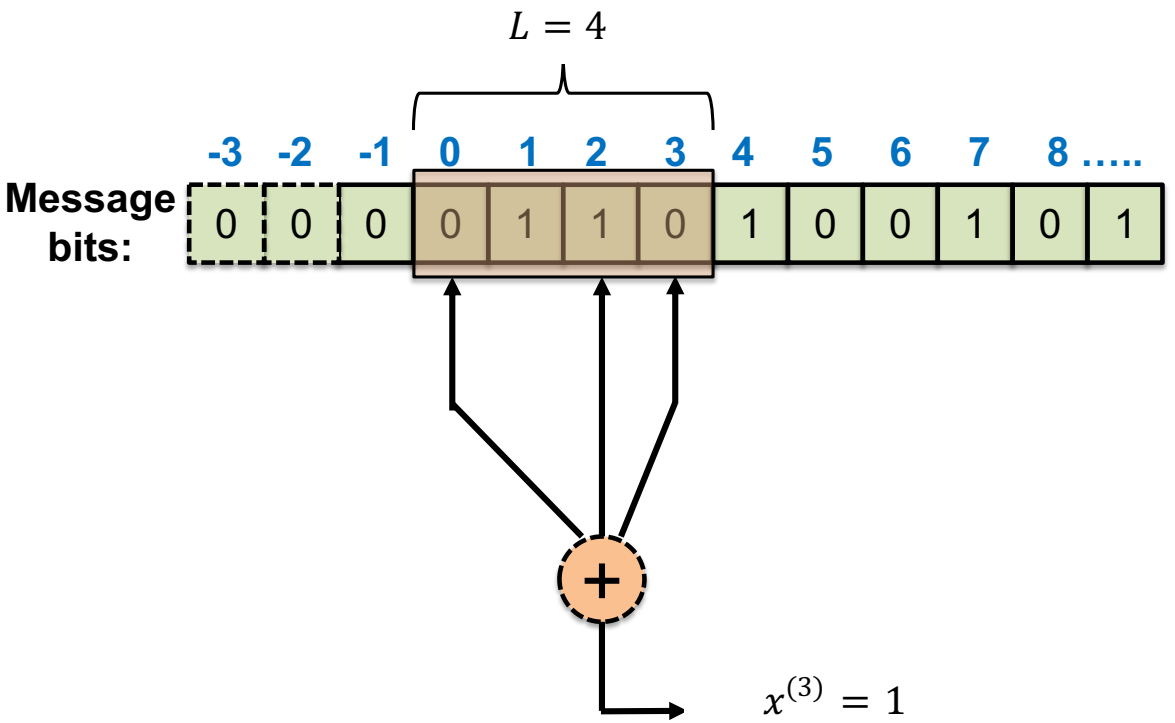
- **Output: 01**

Esempio di generatore $\mathbf{g} = [1101]$



- **Output: 010**

Esempio di generatore $\mathbf{g} = [1101]$



- **Output: 0101**

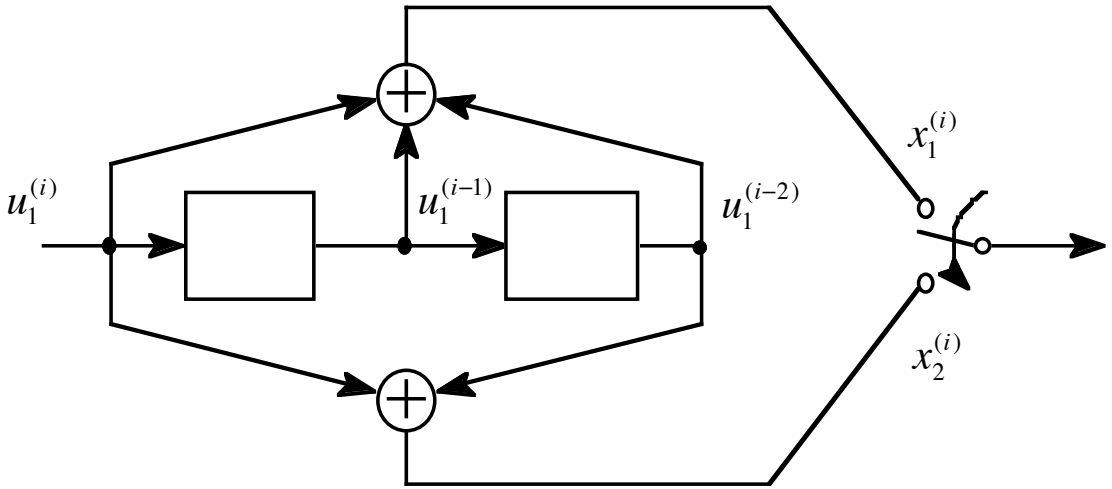
Esempio di codice convoluzionale: i generatori

Consideriamo il codice con $k = 1$ e $R = 1/2$, con generatori $\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$. La constraint length è $L = 3$.

- Le due uscite sono

$$\begin{cases} x_1^{(i)} = u_1^{(i)} + u_1^{(i-1)} + u_1^{(i-2)} \\ x_2^{(i)} = u_1^{(i)} + u_1^{(i-2)} \end{cases}$$

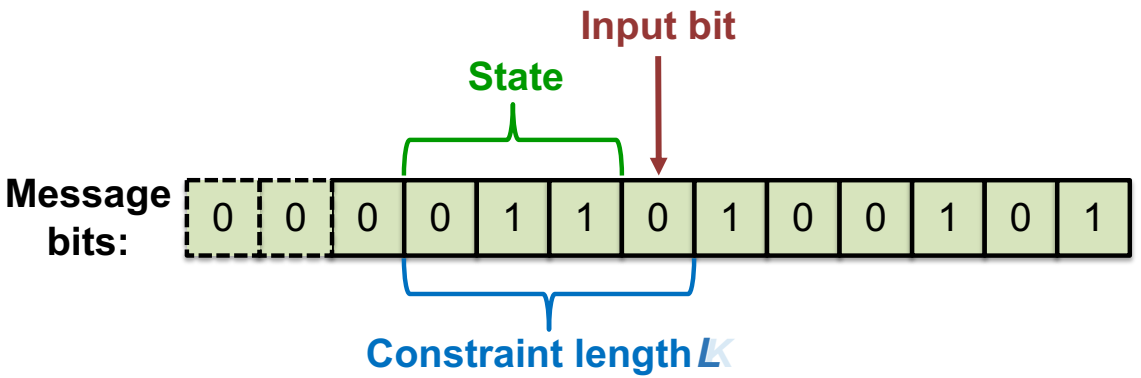
- Il diagramma a blocchi del codificatore è



Rappresentazione di un codice convoluzionale come macchina a stati

- Ad ogni istante di segnalazione l'uscita del codificatore convoluzionale dipende dall'ingresso attuale e dalla memoria.
- Le ultime $k(L-1)$ celle dello shift register memorizzano la storia passata del codificatore che è riassunta nel vettore di stato

$$\boldsymbol{\sigma}^{(i)} = \left(\mathbf{u}^{(i-1)}, \mathbf{u}^{(i-2)}, \ldots, \mathbf{u}^{(i-L+1)}\right)$$



Rappresentazione di un codice convoluzionale come macchina a stati

- Il codificatore può pensarsi come una *macchina a stati finiti* la cui evoluzione nel tempo è descritta da equazioni di stato

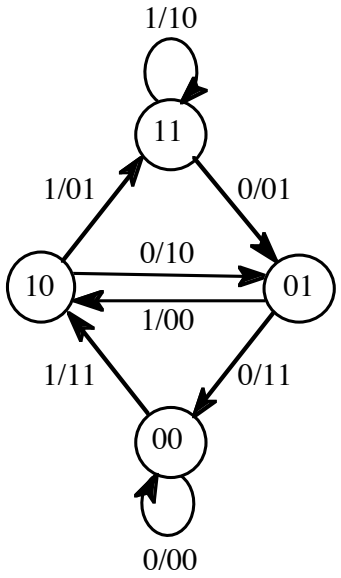
$$\begin{cases} \mathbf{x}^{(i)} = & \delta\left(\mathbf{u}^{(i)}, \boldsymbol{\sigma}^{(i)}\right) \\ \boldsymbol{\sigma}^{(i+1)} = & \lambda\left(\mathbf{u}^{(i)}, \boldsymbol{\sigma}^{(i)}\right) \end{cases}$$

  La prima è detta *equazione di uscita*, la seconda *equazione di transizione di stato*. Il numero di stati complessivo è $2^{k(L-1)}$.
- Il funzionamento del codificatore si può esprimere con un *diagramma di stato* nel quale sono indicate le transizioni da uno stato all'altro sotto l'effetto dei diversi ingressi.

Esempio di codice convoluzionale: il diagramma di stato

Consideriamo il codice con $k = 1$ e $R = 1/2$, con generatori $\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$.

- Il numero di stati è $2^{k(L-1)} = 4$ ed il diagramma di stato è

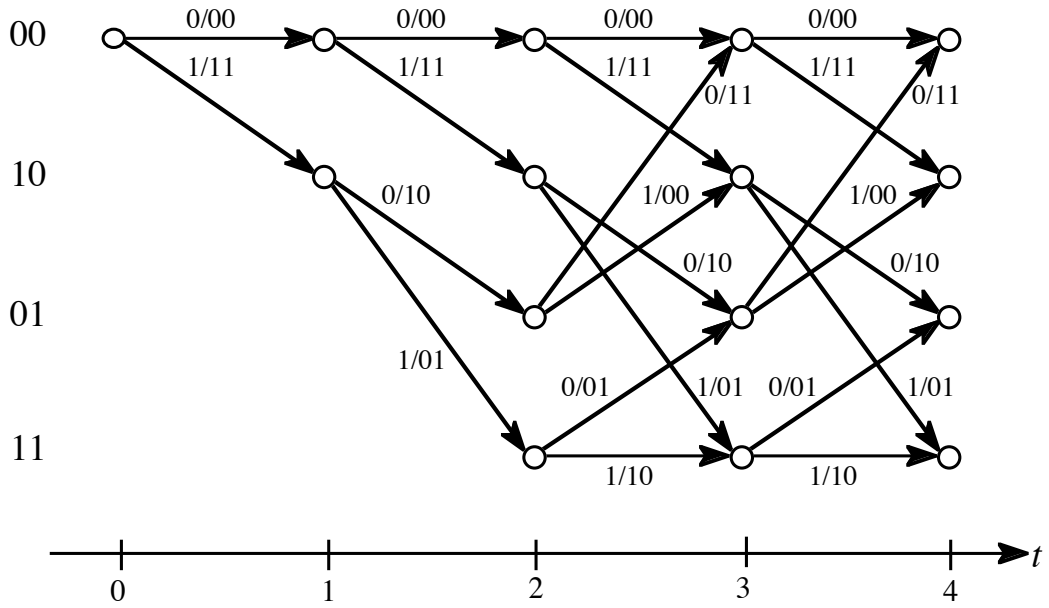


Introduzione ai codici convoluzionali

- Il diagramma di stato non ha un indicatore temporale. Introducendo un indicatore temporale il diagramma di stato si trasforma nel *diagramma a traliccio*, su cui è possibile seguire l'evoluzione degli stati e delle uscite del codificatore in seguito a una determinata sequenza di vettori di ingresso.
- Partendo da un qualsiasi stato, si può raggiungere qualsiasi altro stato del traliccio in un numero massimo di $L - 1$ passi.

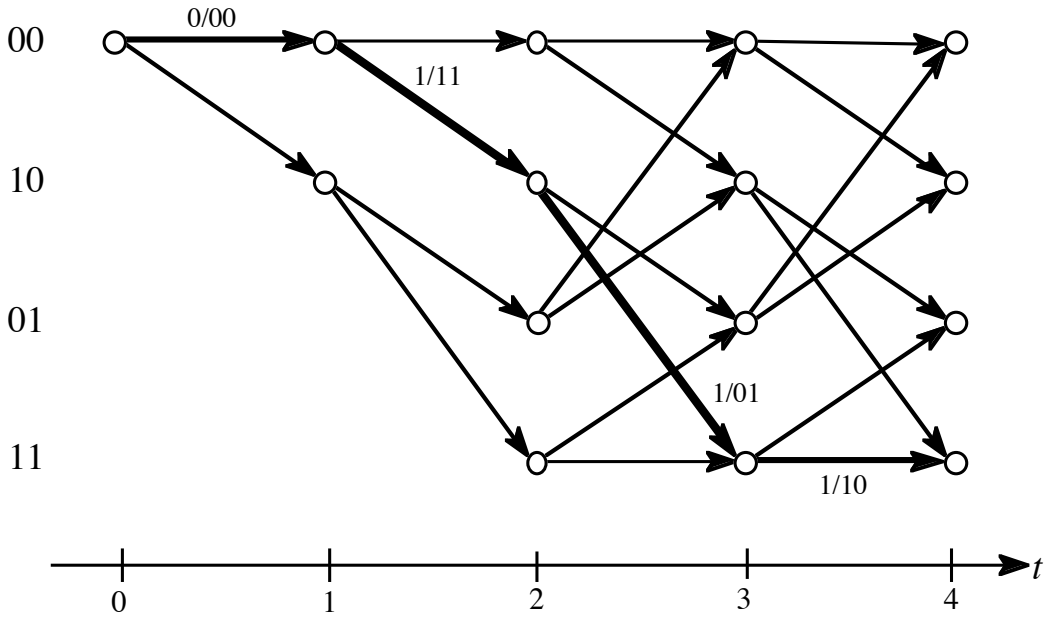
Esempio di codice convoluzionale: il diagramma a traliccio

- Il diagramma a traliccio per il il codice con $k = 1$ e $R = 1/2$, con generatori $\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$ è



Esempio di codice convoluzionale: il diagramma a traliccio

Il diagramma a traliccio permette di osservare l'evoluzione del codificatore in risposta ad una determinata sequenza di ingresso. Per esempio per il il codice con $k = 1$ e generatori $\mathbf{g}_1 = 7_8$ e $\mathbf{g}_2 = 5_8$ se $\boldsymbol{\sigma}^{(0)} = 00$ e in ingresso si ha la sequenza $u^{(0)} = 0$, $u^{(1)} = 1, u^{(2)} = 1$ e $u^{(3)} = 1$ il percorso sul traliccio sarà



Distanza colonna per un codice convoluzionale

- Sia dato un codice convoluzionale $\mathcal{C}(n, k, L)$. Consideriamo $\mathcal{X}_{\mathcal{C}}(\ell, \boldsymbol{\sigma})$, l'insieme di tutte le possibili sequenze che originano dallo stato $\boldsymbol{\sigma}$ e hanno lunghezza pari a ℓn, ottenute cioè dopo ℓ passi sul traliccio.
- La *distanza colonna* del codice $\mathcal{C}$ al passo ℓ è la minima distanza di Hamming fra due sequenze $\mathbf{x}$ e $\mathbf{x}'$ in $\mathcal{X}_{\mathcal{C}}(\ell, \boldsymbol{\sigma})$ che siano differenti nei primi n bit, in corrispondenza quindi della prima parola di uscita del codificatore

$$d_c(\ell) = \min_{\substack{\mathbf{x},\mathbf{x}' \in \mathcal{X}_{\mathcal{C}}(\ell,\boldsymbol{\sigma}) \\ \mathbf{x}^{(1)} \neq \mathbf{x}'^{(1)}}} d_H(\mathbf{x}, \mathbf{x}')$$

- Come nel caso del calcolo della d_{min} di un codice a blocco, non fa nessuna differenza la particolare sequenza di riferimento, e quindi nel calcolo della $d_c(\ell)$ si sceglie lo stato $\boldsymbol{\sigma}$ in modo che una delle due sequenza sia il vettore di tutti zeri.

Distanza libera per un codice convoluzionale

- La distanza libera di un codice convoluzionale $\mathcal{C}$, d_{free} è il limite della distanza colonna per ℓ che tende all'infinito, i.e,

$$d_{free} = \lim_{\ell \to \infty} d_c(\ell)$$

- In pratica la distanza libera coincide con la distanza colonna quando i due percorsi sul traliccio a massima distanza confluiscono.
- La distanza libera, come la $d_{\min}$ nei codici a blocco, è una misura della bontà del codice: tanto maggiore è la distanza libera tanto più è difficile confondere due sequenze che originano dallo stesso stato.

Strategia di decodifica a massima verosimiglianza

- Si consideri una trasmissione su N intervalli di segnalazione. La trasmissione è codificata con il codice convoluzionale $\mathcal{C}(n, k, L)$, che ad ogni segnalazione associa k bit di informazione a n bit codificati.
- Sia $\mathbf{y}$ la sequenza ricevuta di lunghezza nN bit, la strategia di decodifica a massima verosimiglianza consiste nel trovare la sequenza $\hat{\mathbf{x}}$ che, fra tutte le 2^{kN} possibili sequenze di tentativo $\tilde{\mathbf{x}}$, massimizza la probabilità condizionata $P(\mathbf{y}|\tilde{\mathbf{x}})$, ie

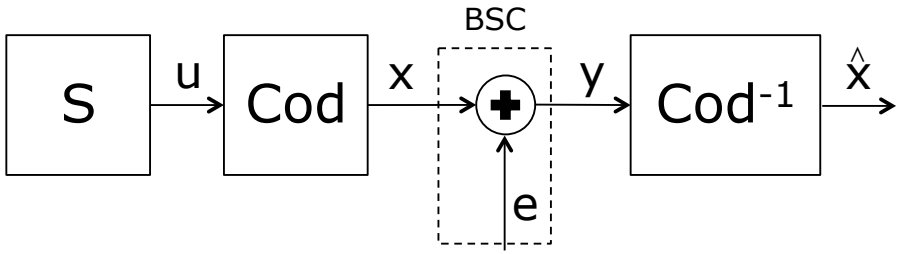
$$\hat{\mathbf{x}} = \arg\max_{\tilde{\mathbf{x}}} P(\mathbf{y}|\tilde{\mathbf{x}})$$

- Poichè gli eventi di errore sono indipendenti da bit a bit è conveniente riscrivere la probabilità condizionata come il prodotto delle probabilità condizionate ottenute per ciascuna parola di codice trasmessa

$$P(\mathbf{y}|\tilde{\mathbf{x}}) = \prod_{\ell=1}^{N} P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}\right) = \prod_{\ell=1}^{N} P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)}\right) \qquad (1)$$

Strategia di decodifica a massima verosimiglianza

- Si assume che la probabilità di ricevere un bit errato sia p.



- La probabilità condizionata $P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)}\right)$ si ottiene a partire dal calcolo della distanza di Hamming $\lambda_{\tilde{\mathbf{x}}}^{(\ell)} = d_H\left(\mathbf{y}^{(\ell)}, \tilde{\mathbf{x}}^{(\ell)}\right)$, infatti per ogni bit diverso tra le due parole si assume che ci sia stato un errore e quindi

$$P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)}\right) = p^{\lambda_{\tilde{\mathbf{x}}}^{(\ell)}}(1-p)^{n-\lambda_{\tilde{\mathbf{x}}}^{(\ell)}} = (1-p)^n\left(\frac{p}{1-p}\right)^{\lambda_{\tilde{\mathbf{x}}}^{(\ell)}} \tag{2}$$

Strategia di decodifica a massima verosimiglianza

- Poiché la funzione logaritmo è crescente, la sequenza di tentativo che massimizza $P(\mathbf{y}|\tilde{\mathbf{x}})$ massimizza anche $\log(P(\mathbf{y}|\tilde{\mathbf{x}}))$.
- Sostituendo in (1) il valore trovato in (2) e prendendo il logaritmo del risultato si ottiene

$$\log(P(\mathbf{y}|\tilde{\mathbf{x}})) = nN\log(1-p) + \log\left(\frac{p}{1-p}\right)\sum_{\ell=1}^{N}\lambda_{\tilde{\mathbf{x}}}^{(\ell)}$$

- Trascurando i termini ininfluenti e considerando che è lecito assumere $\log\left(\frac{p}{1-p}\right) < 0$, si ottiene la regola di decisione

$$\hat{\mathbf{x}} = \arg\max_{\tilde{\mathbf{x}}} P(\mathbf{y}|\tilde{\mathbf{x}}) = \arg\min_{\tilde{\mathbf{x}}} \sum_{\ell=1}^{N}\lambda_{\tilde{\mathbf{x}}}^{(\ell)}$$

Strategia di decodifica a massima verosimiglianza

- La strategia a massima verosimiglianza consiste nello scegliere la sequenza $\hat{\mathbf{x}}$ che minimizza la distanza di Hamming dalla sequenza ricevuta, i.e,

$$d_H(\mathbf{y}, \hat{\mathbf{x}}) = \sum_{\ell=1}^{N} \lambda_{\hat{\mathbf{x}}}^{(\ell)}$$

- Se si osserva che la trasmissione dura N intervalli di segnalazione, in cui ogni volta vengono inviati al codificatore k bit, il numero di sequenza da considerare è 2^{kN}.
- ATTENZIONE: La complessità della decodifica cresce *esponenzialmente* con k e N.

Algoritmo di Viterbi

- Consideriamo un codice $\mathcal{C}(n, k, L)$ e assumiamo che si effettui una trasmissione di N parole di codice ciascuna composta da n bit e che lo stato iniziale $\boldsymbol{\sigma}^{(0)}$ e quello finale $\boldsymbol{\sigma}^{(N)}$ del codificatore siano conosciuti anche al ricevitore.
- Data la sequenza ricevuta $\mathbf{y}$ di nN bit, l'obiettivo del decodificatore è individuare sul traliccio il percorso $\hat{\mathbf{x}}$ più breve (a minima distanza di Hamming!) che partendo da $\boldsymbol{\sigma}^{(0)}$ arrivi a $\boldsymbol{\sigma}^{(N)}$.
- La sequenza $\hat{\mathbf{x}}$ deve minimizzare la metrica

$$d_H(\mathbf{y}, \hat{\mathbf{x}}) = \Lambda(\mathbf{y}, \hat{\mathbf{x}}) = \sum_{\ell=1}^{N} \lambda_{\hat{\mathbf{x}}}^{(\ell)}$$

Algoritmo di Viterbi

L'algoritmo a bassa complessità per trovare il percorso più breve sul traliccio è stato brevettato nel 1967 da A.J. Viterbi.

- L'intuizione fondamentale è che un gran numero dei 2^{kN} possibili percorsi sul traliccio non è rilevante al fine del calcolo della distanza minima e quindi può essere scartato.

Andrew James Viterbi
(born Andrea Giacomo Viterbi in Bergamo, Italy)
is Professor of Electrical Engineering at the University of Southern California's Viterbi School of Engineering, which was named after him in recognition of his **$52 million** gift.

Algoritmo di Viterbi

Si definisce:

- *metrica di ramo* $\boldsymbol{\sigma}_j \to \boldsymbol{\sigma}_k$ al passo ℓ, la distanza di Hamming $\lambda^{(\ell)}(\boldsymbol{\sigma}_j, \boldsymbol{\sigma}_k) = d_H\left(\mathbf{y}^{(\ell)}, \mathbf{x}_{\boldsymbol{\sigma}_j \to \boldsymbol{\sigma}_k}\right)$. La metrica è calcolata come la distanza tra la sequenza ricevuta al passo ℓ e l'uscita corrispondente alla transizione sul traliccio dallo stato $\boldsymbol{\sigma}_j$ allo stato $\boldsymbol{\sigma}_k$;
- *metrica cumulata* al passo f allo stato $\boldsymbol{\sigma}_k$, la grandezza $\Lambda_{\mathbf{x}}^{(f)}(\boldsymbol{\sigma}_k)$ ottenuta sommando tutte le f metriche di ramo calcolate su gli f rami sul traliccio di un percorso $\mathbf{x}$ che si fermi allo stato $\boldsymbol{\sigma}_k$ al passo f.

Algoritmo di Viterbi

- L'algoritmo di Viterbi si basa sulla seguente intuizione:
  - Supponiamo che due diversi percorsi $\mathbf{x}_1$ e $\mathbf{x}_2$ confluiscano al passo f nello stesso nodo $\boldsymbol{\sigma}_k$ sul traliccio e siano $\Lambda_{\mathbf{x}_1}^{(f)}(\boldsymbol{\sigma}_k) < \Lambda_{\mathbf{x}_2}^{(f)}(\boldsymbol{\sigma}_k)$ le metriche cumulate dei due percorsi calcolate al passo f.
  - Supponiamo che al passo successivo $f+1$, i percorsi $\mathbf{x}_1$ e $\mathbf{x}_2$ seguano lo stesso ramo sul traliccio, ad esempio $\boldsymbol{\sigma}_j \to \boldsymbol{\sigma}_q$. In questo caso la metrica di ramo è la stessa per i due percorsi $\lambda^{(f+1)}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_q)$.
  - Per le nuove metriche sarà ancora $\Lambda_{\mathbf{x}_1}^{(f+1)}(\boldsymbol{\sigma}_q) < \Lambda_{\mathbf{x}_2}^{(f+1)}(\boldsymbol{\sigma}_q)$, infatti si ha

$$\begin{aligned}\Lambda_{\mathbf{x}_1}^{(f+1)}(\boldsymbol{\sigma}_q) &= \Lambda_{\mathbf{x}_1}^{(f)}(\boldsymbol{\sigma}_k) + \lambda^{(f+1)}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_q) \\ &< \Lambda_{\mathbf{x}_2}^{(f)}(\boldsymbol{\sigma}_k) + \lambda^{(f+1)}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_q) = \Lambda_{\mathbf{x}_2}^{(f+1)}(\boldsymbol{\sigma}_q)\end{aligned}$$

Algoritmo di Viterbi

Generalizzando questa intuizione si può dire che :

1. In un traliccio arrivato alla sua piena espansione ad ogni nuovo istante di segnalazione arrivano 2^k percorsi a ciascun nodo.
2. I valori delle metriche di ramo in uscita da un nodo sono uguali per tutti i percorsi che arrivano a quel nodo.
3. A parità di percorso futuro, il percorso in ingresso al nodo con la metrica cumulata minore continuerà ad avere metrica cumulata più bassa di tutti gli altri.
4. Ai fini della minimizzazione della metrica cumulata complessiva ad ogni nodo si possono scartare i $2^k - 1$ percorsi con la metrica cumulata più alta e conservare solo quello con la metrica cumulata minima.
5. L'unico percorso rimasto viene definito *sopravvissuto*.

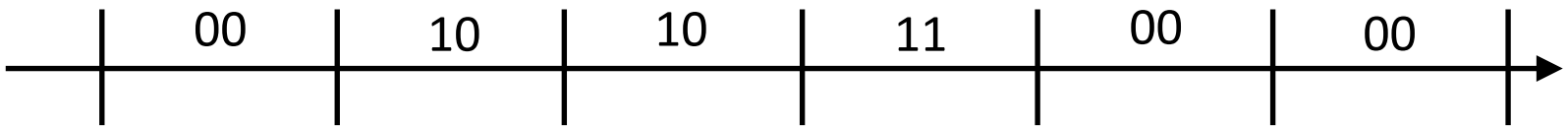
Esempio di applicazione dell'algoritmo di Viterbi

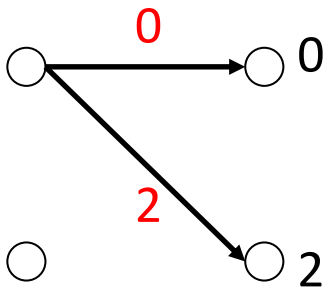
- Consideriamo il codice convoluzionale con rate $R = 1/2$, constraint length $L = 3$ e generatori $\mathbf{g}_1 = 7_8$ e $\mathbf{g}_2 = 5_8$.
- Assumiamo che la sequenza di $N = 6$ bit informativi sia $\mathbf{u} = [0, 1, 0, 0, 0, 0]$ a cui corrisponde la sequenza codificata $\mathbf{x} = [00, 11, 10, 11, 00, 00]$ e supponiamo che la sequenza ricevuta sia $\mathbf{y} = [00, 10, 10, 11, 00, 00]$, in cui il 2o bit della 2a parola di codice è errato.
- In un sistema reale, $N \gg 6$, $N = 6$ è solo a scopo illustrativo.

Esempio di applicazione dell'algoritmo di Viterbi

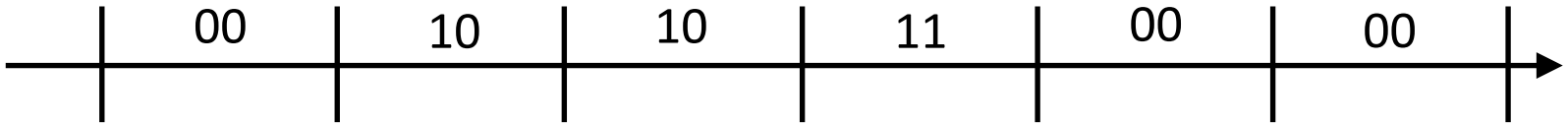
- Le metriche di ramo sono indicate in rosso, quelle cumulate in nero.
- La trasmissione inizia nello stato $\boldsymbol{\sigma}^{(0)} = [0, 0]$ e dopo $N = 6$ segnalazioni torna nello stato $\boldsymbol{\sigma}^{(N)} = [0, 0]$.

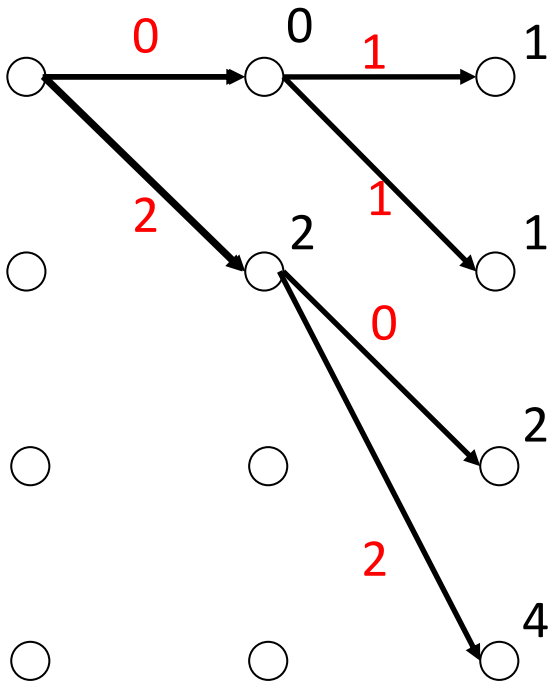
Esempio di applicazione dell'algoritmo di Viterbi



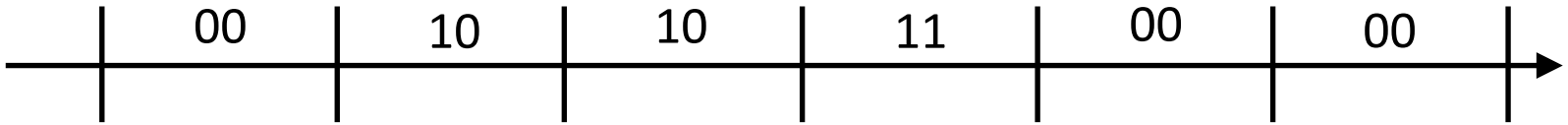


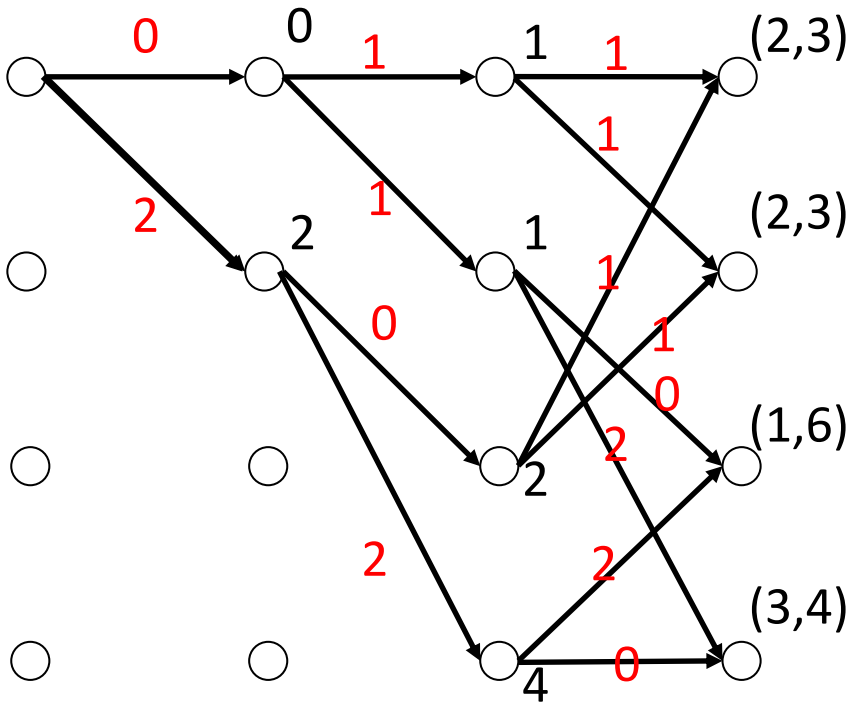
Esempio di applicazione dell'algoritmo di Viterbi



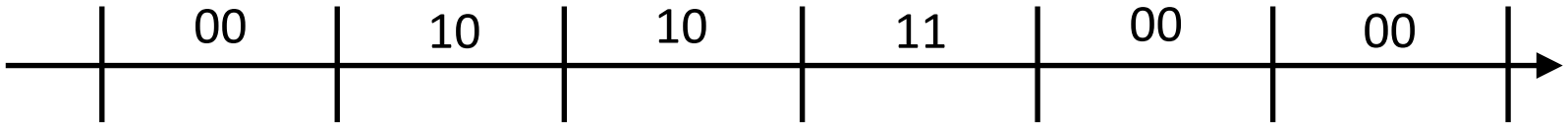


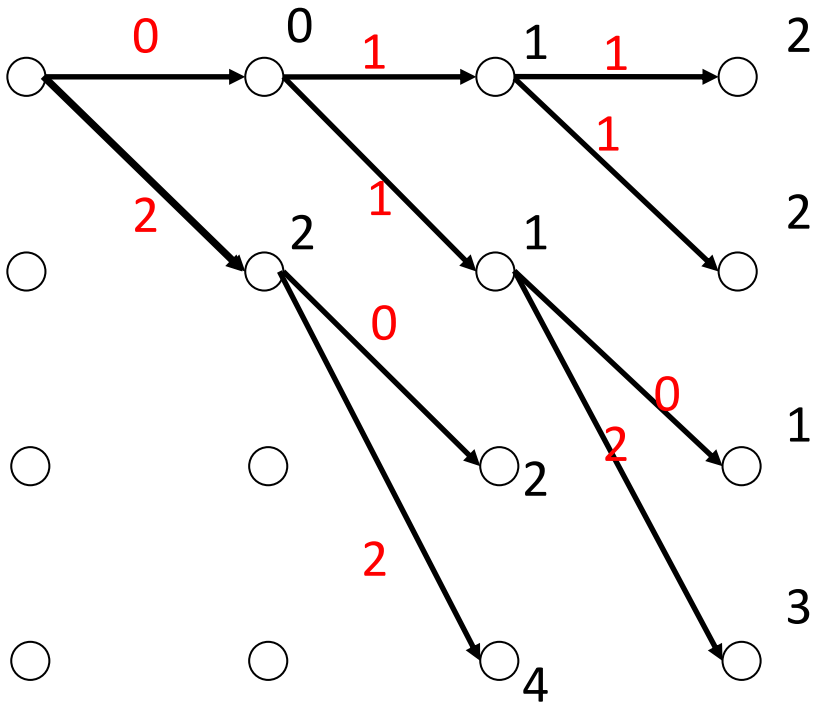
Esempio di applicazione dell'algoritmo di Viterbi



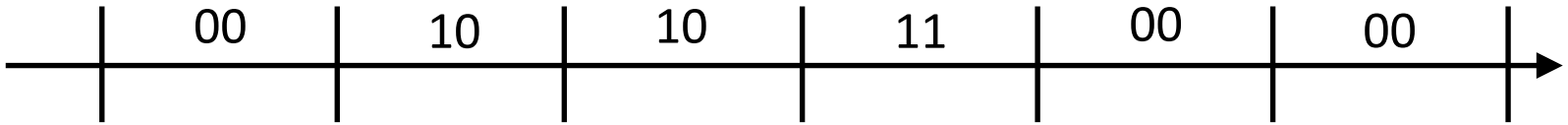


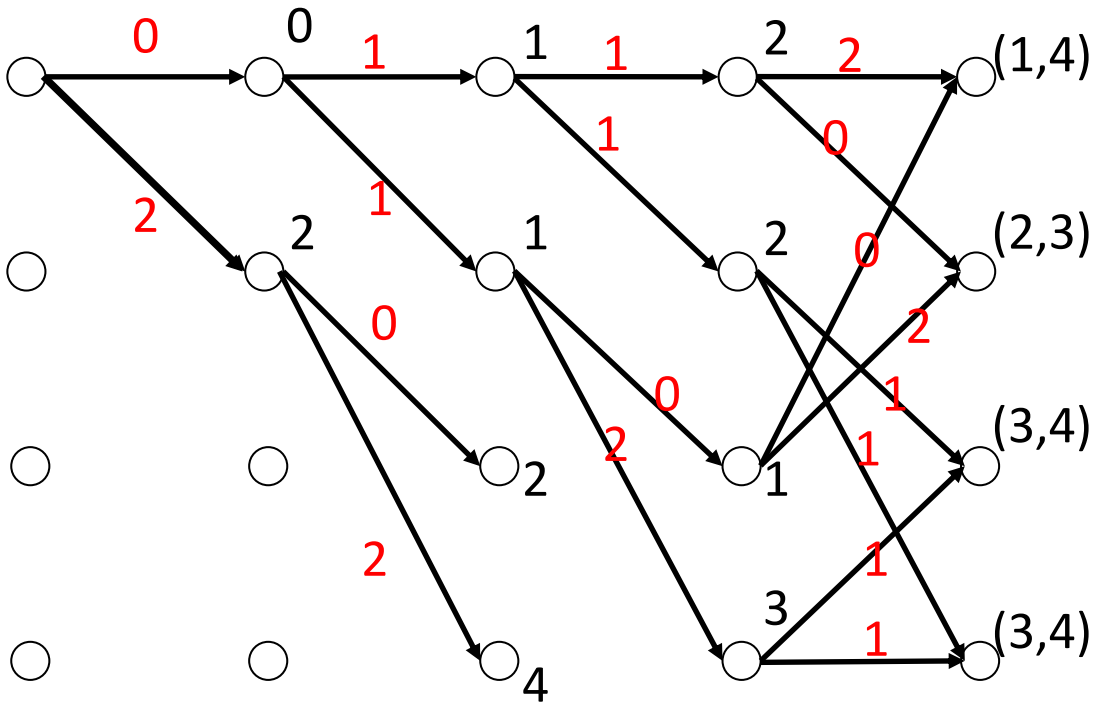
Esempio di applicazione dell'algoritmo di Viterbi



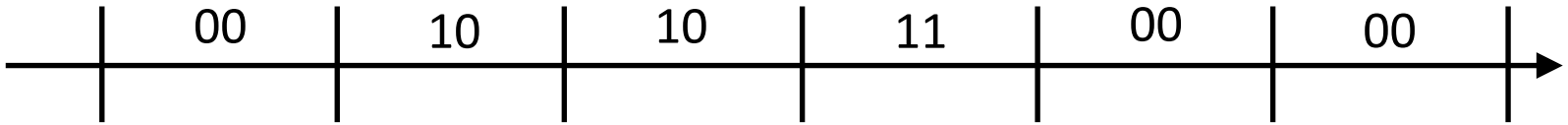


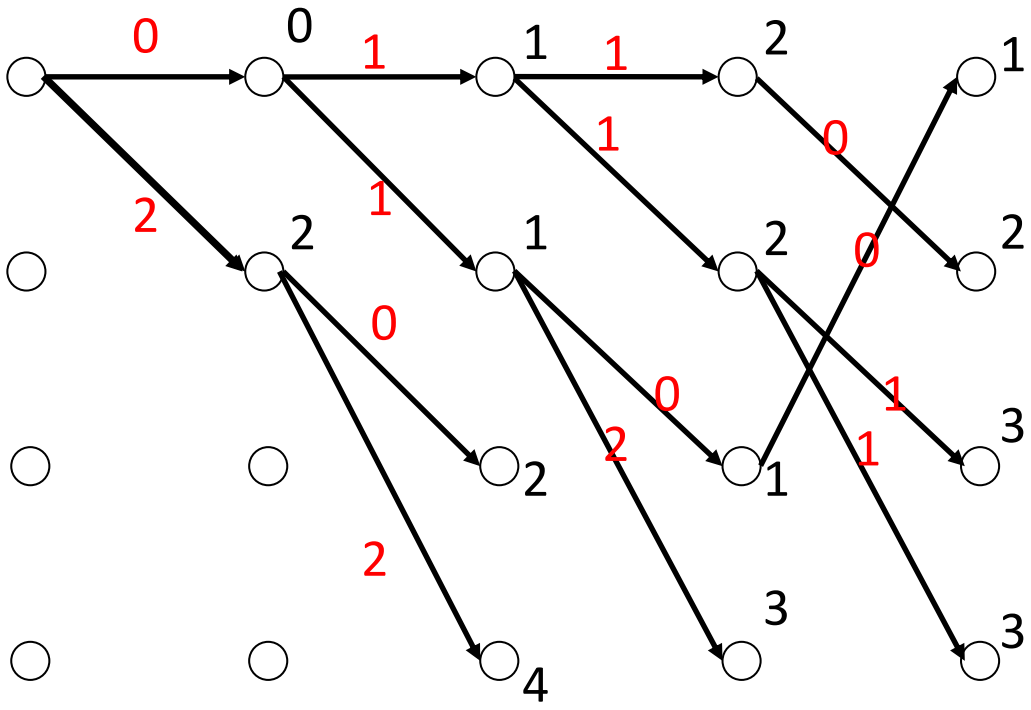
Esempio di applicazione dell'algoritmo di Viterbi



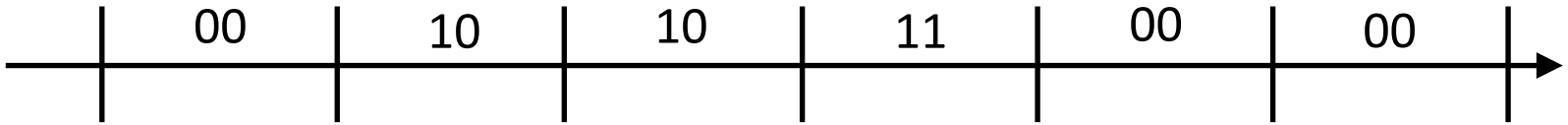


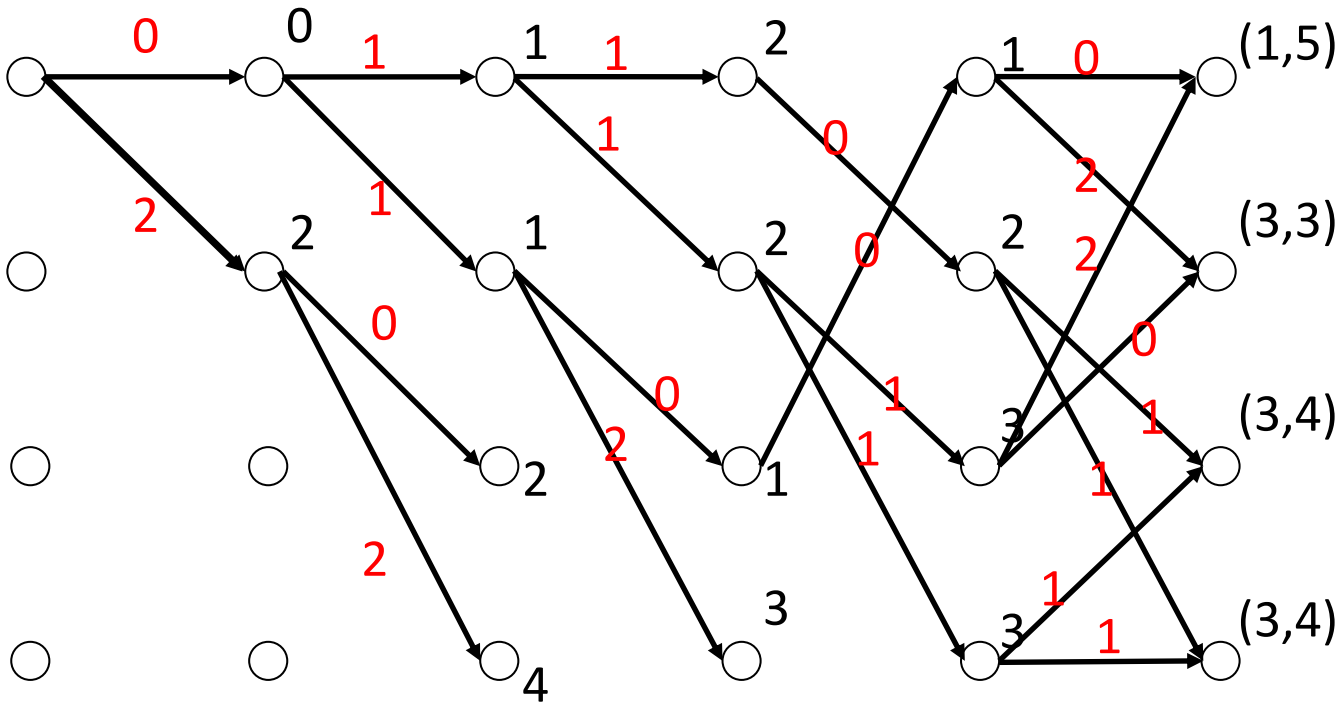
Esempio di applicazione dell'algoritmo di Viterbi



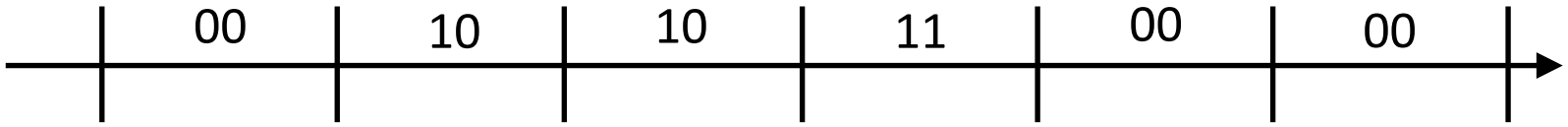


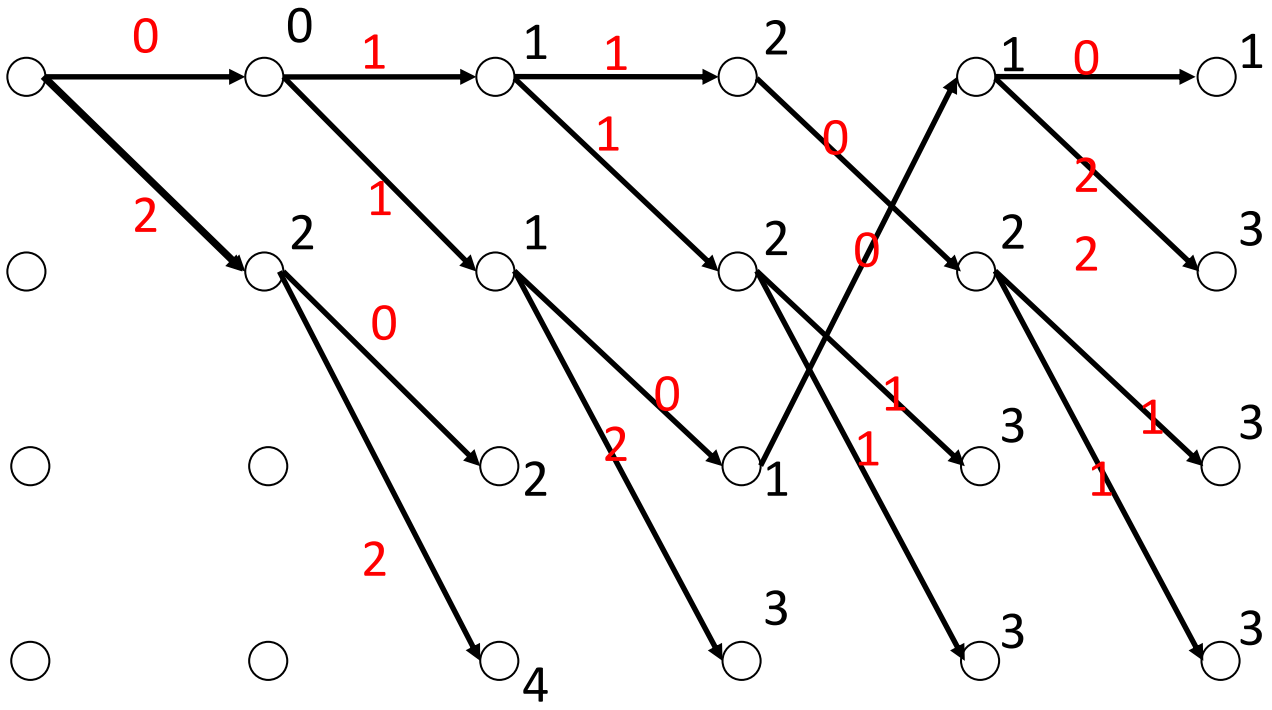
Esempio di applicazione dell'algoritmo di Viterbi



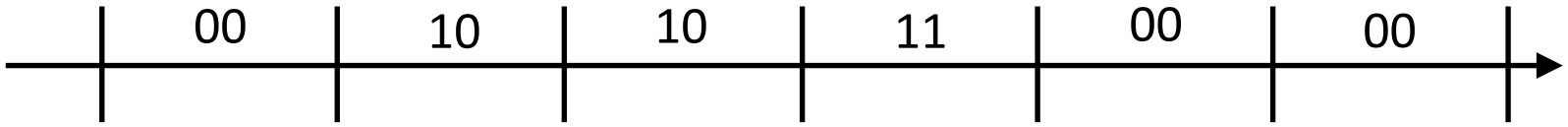


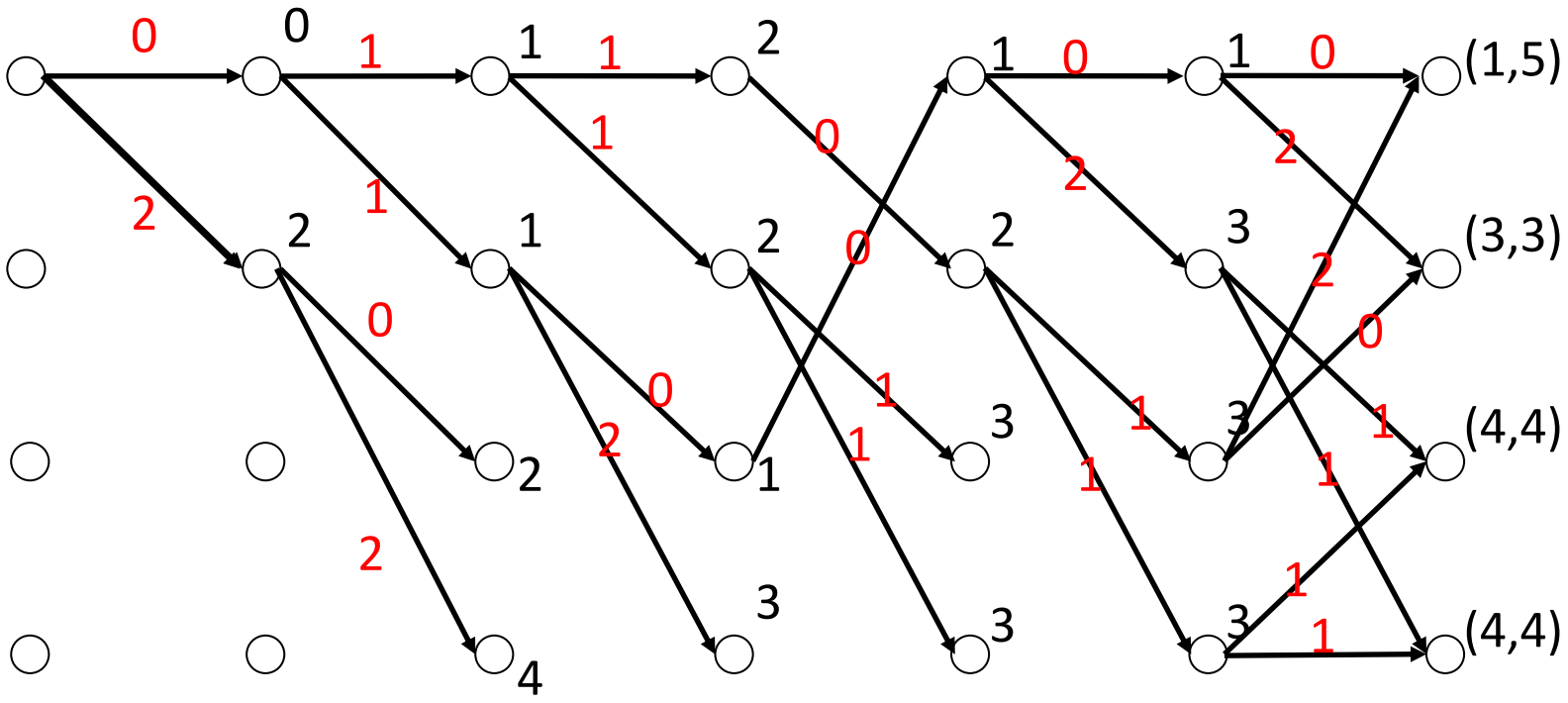
Esempio di applicazione dell'algoritmo di Viterbi



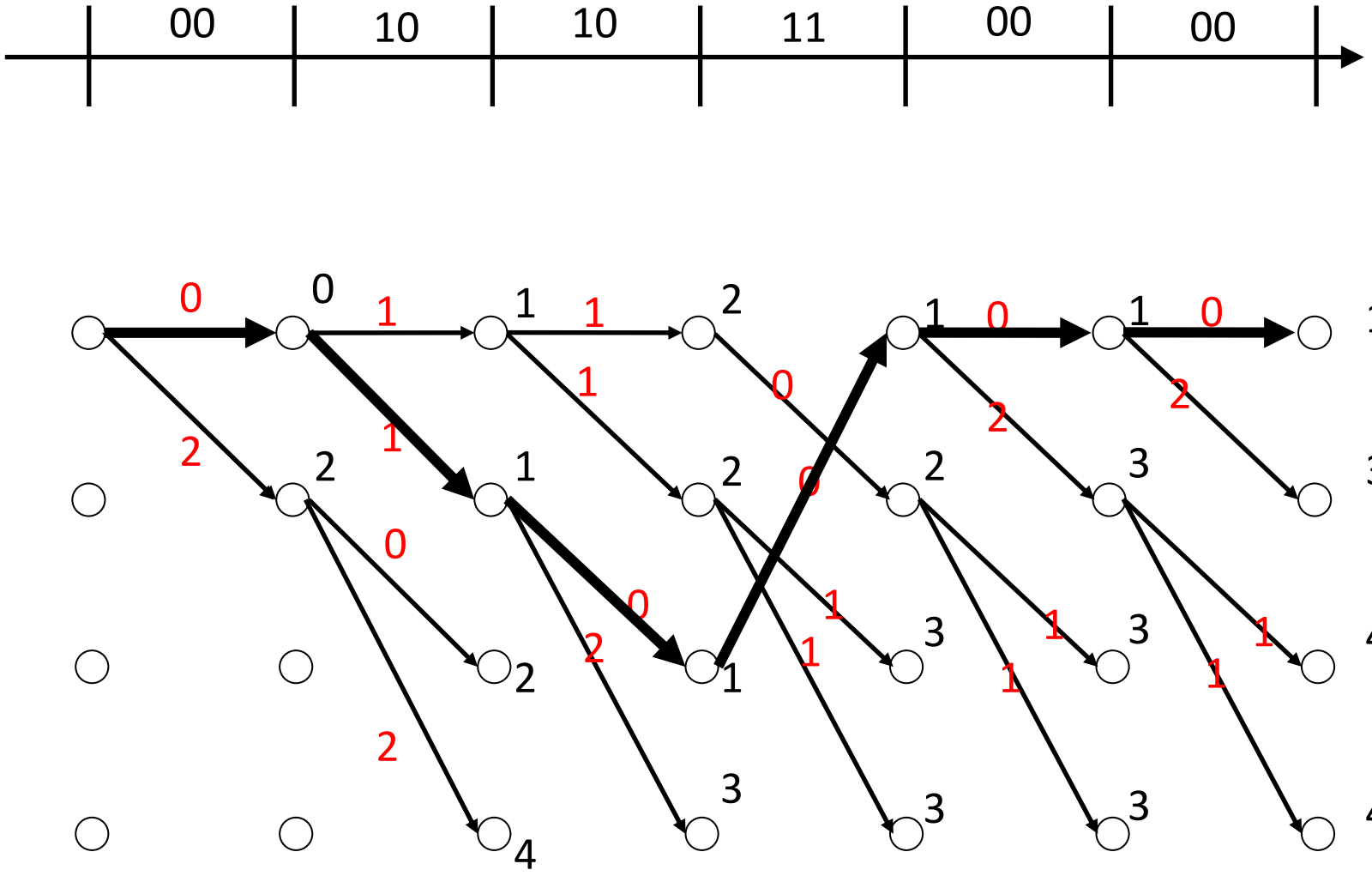


Esempio di applicazione dell'algoritmo di Viterbi





Esempio di applicazione dell'algoritmo di Viterbi



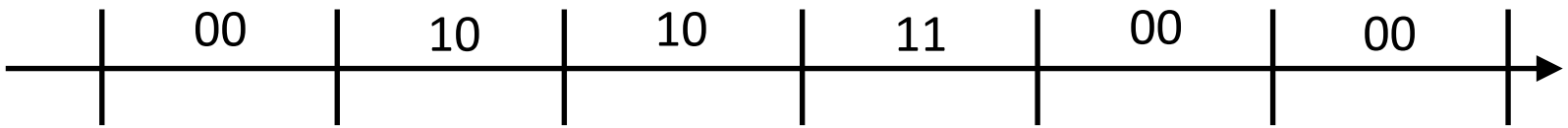
Esempio di applicazione dell'algoritmo di Viterbi

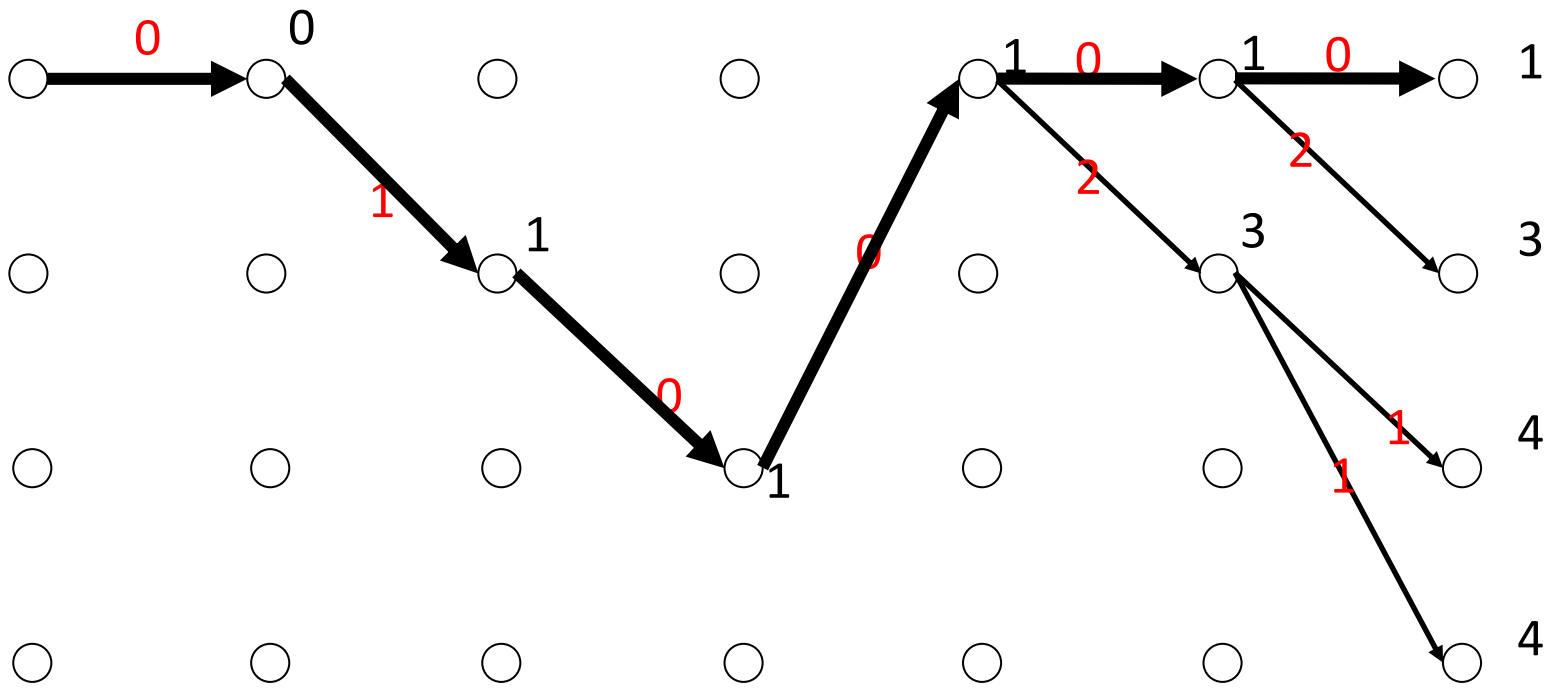
- La trasmissione inizia nello stato 00 e dopo $N = 6$ segnalazioni torna nello stato 00.
- Una volta identificato il percorso a distanza minima sul traliccio, identificare la sequenza $\hat{\mathbf{u}}$ informativa è semplice: è sufficiente associare ad ogni transizione sul traliccio la sequenza di bit di ingresso corrispondente.
- Nel nostro caso si ha $\hat{\mathbf{u}} = [0, 1, 0, 0, 0, 0] = \mathbf{u}$. Il decodificatore ha corretto l'errore introdotto dal canale!

Considerazioni finali sull'algoritmo di Viterbi

- A partire dal momento in cui il traliccio ha raggiunto il suo pieno sviluppo, ad ogni passo l'algoritmo deve
  1. Calcolare le metriche di ramo per tutte le transizioni possibili;
  2. Calcolare per ogni nodo le metriche cumulate dei percorsi in ingresso;
  3. Per ogni nodo trovare il percorso sopravvissuto associato alla metrica cumulata minima.
- La complessità dell'algoritmo di Viterbi cresce *linearmente* con la lunghezza della sequenza da decodificare e non esponenzialmente, come avviene con la ricerca esaustiva.

Considerazioni finali sull'algoritmo di Viterbi





Considerazioni finali sull'algoritmo di Viterbi

Osservando il traliccio al passo ℓ, si trova che andando a ritroso di D passi, i sopravvissuti tendono a fondersi in un unico percorso. Poiché si è visto empiricamente che D è una quantità fissa dell'ordine di kL, possiamo trarre le seguenti conclusioni:

1. Il decodificatore di Viterbi non deve necessariamente attendere di essere arrivato alla fine della segnalazione per produrre decisioni, ma è sufficiente introdurre un ritardo di decodifica pari a D e poi prendere decisioni di decodifica ad ogni nuovo istante di segnalazione.
2. La parte comune dei sopravvissuti (quella su cui sono già state prese le decisioni) non ha bisogno di essere memorizzata. È sufficiente memorizzare i sopravvissuti nell'ultimo tratto da $\ell - D$ a ℓ. Questo riduce enormemente l'esigenza di memoria dell'algoritmo.