Codes Correcteurs d'Erreurs
Les codes convolutifs binaires

Marc Chaumont

November 12, 2008

Plan

Quelques mots

Introduits par Elias en 1954, ce sont probablement les codes les plus populaires. On trouve ceux-ci dans de nombreuses applications : communication sans fils (IMT-2000, GSM, IS-95), communication terrestre et satellitaire, communication spatialle. Leur méthode de décodage la plus populaire repose sur l'algorithme de Viterbi. Récemment, il a été montré que les codes convolutifs combinés avec l'entrelacement dans les schémas de concaténation pouvait s'approcher de la limite théorique de Shannon.

Codes blocs versus codes convolutifs

codes blocs versus codes convolutifs

Les ECC peuvent être divisés en 2 classes :

- **les codes en bloc** : Ils traitent chaque bloc d'information indépendamment les uns des autres.
- **les codes convolutifs** : La sortie d'un codeur convolutif dépend d'un symbole courant à coder ainsi que du symbole précédent et du résultat de codage du symbole précédent.

Codes convolutifs

Codeur convolutif

Un codeur convolutif est une **machine à états finis**.

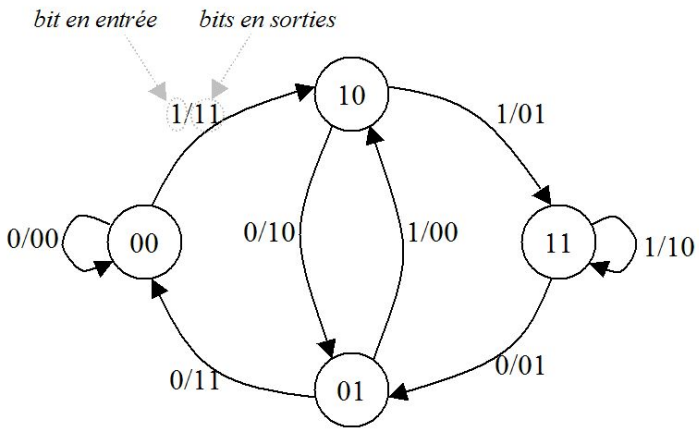


Figure: Diagramme d'état d'un codeur convolutif *2-mémoires 1/2-taux*

Note : le **taux** ou **rendement** de ce code convolutif est de 1/2 (2 bits produits pour 1 bit d'information).

Treillis : le dépliement du diagramme d'état

Le diagramme d'état peut être représenté sous forme de **treillis**. Le treillis est l'équivalent temporel du diagramme d'état. Tous les états sont mis en colonne pour représenter un état à l'instant t. Ces états sont reliés aux mêmes ensembles d'états réprésentant l'instant $t+1$, en accord avec la table de transition.

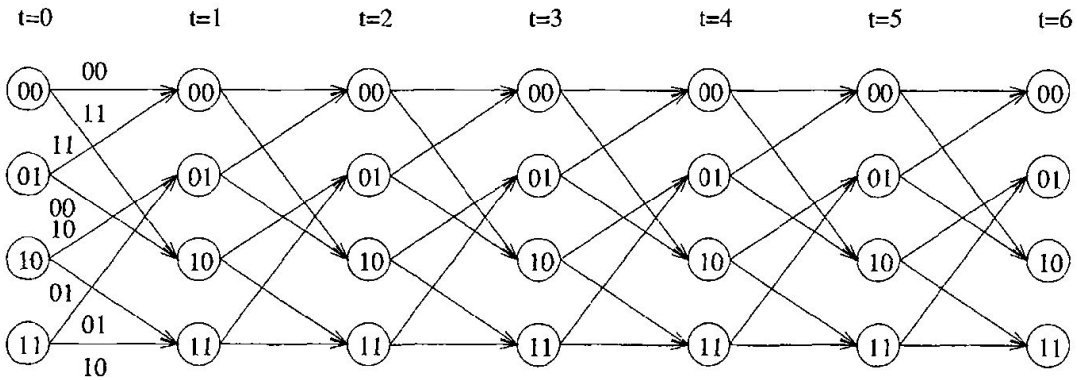


Figure 32 Six sections of the trellis of a memory-2 rate-1/2 convolutional encoder.

Note : L'arc le plus haut dans la transition entre l'état $s[i]$ et l'état $s[i+1]$ est causée par un bit nul en entrée ($u[i] = 0$).

Exemple de lecture sur Treillis

Soit l'entrée $u = (110100)$.

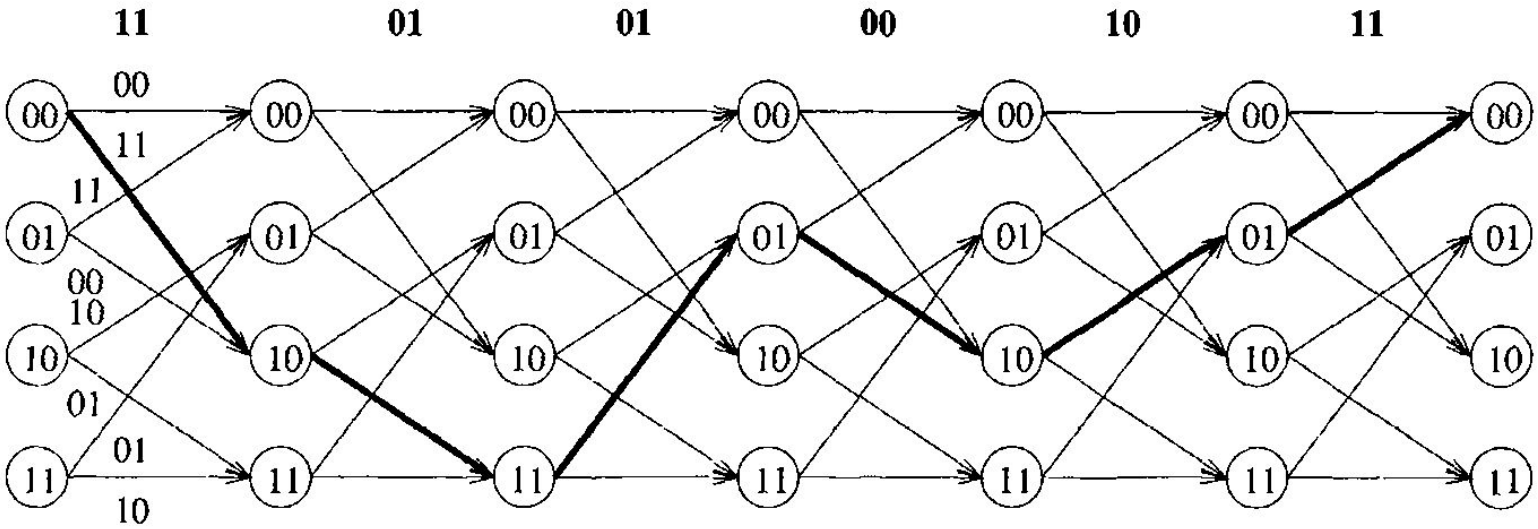


Figure 33 A path in the trellis of a memory-2 rate-1/2 convolutional encoder.

La sortie est $v = (11, 01, 01, 00, 10, 11)$.

Le circuit logique

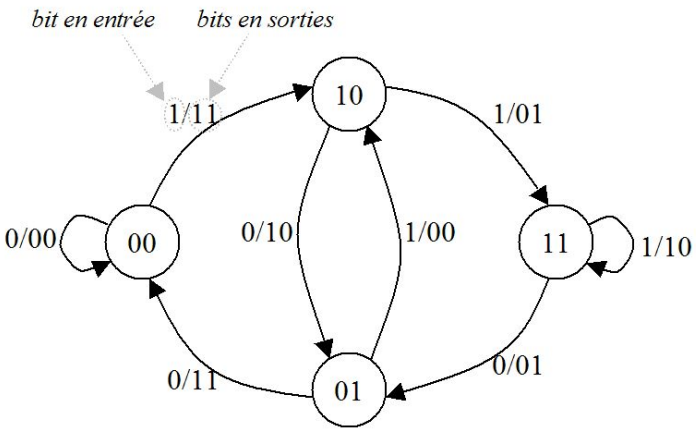
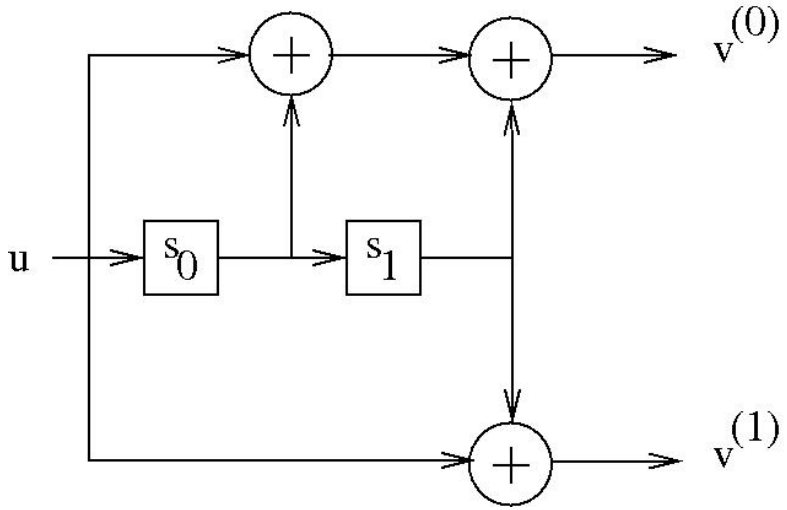
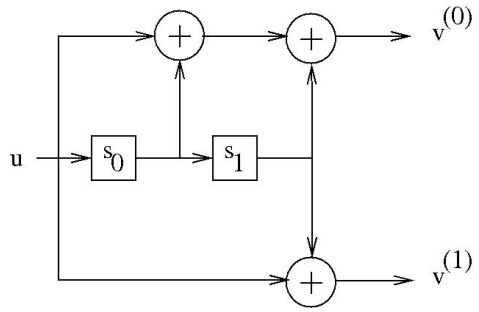


Diagramme d'état



Circuit logique associé
s_0 et s_1 : registres à décalage

Table de transition



Etat initial $s_0[i]s_1[i]$	*Information* $u[i]$	*Etat final* $s_0[i+1]s_1[i+1]$	*Sortie* $v^{(0)}[i]v^{(1)}[i]$
00	0	00	00
00	1	10	11
01	0	00	11
01	1	10	00
10	0	01	10
10	1	11	01
11	0	01	01
11	1	11	10

Note : i représente le temps horloge.

Longueur de contrainte

Codeur à m mémoires

Un codeur à m mémoires sera référencé codeur m-mémoires.

Dans l'exemple précédent m=2.

Longueur de contrainte

La **longueur de contrainte** K d'un codeur convolutif correspond au nombre de temps horloge influençant les sorties du codeur. Pour un 1/2-taux codeur, K=m+1.

Dans l'exemple précédent les temps i, $i-1$ et $i-2$ influencent les sorties du décodeur au temps i. K vaut donc 3.

Remarques supplémentaires

- Un codeur convolutif m-mémoires $1/n$-taux peut être représenté par un **diagramme d'état** à 2^m états.
- $1/n$-taux signifie qu'il y a 1 bit d'information et donc deux arcs entrent et sortent d'un état du diagramme d'état.
- Un codeur convolutif m-mémoires $1/n$-taux peut être considéré comme un système temps-invariant linéaire discret. La réponse (sortie) du codeur à une "impulsion" (exemple u=(1000...00..)) spécifie complètement le code.

Convention

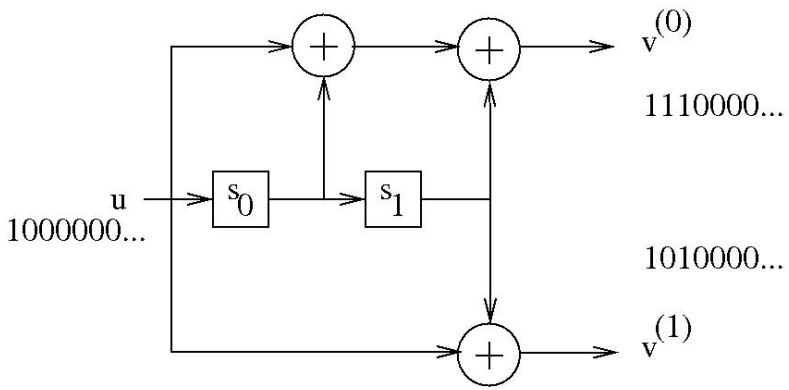
Quand on écrit une séquence de symboles, le premier symbole à entrer dans le codeur est celui qui est le plus à gauche !

Plan

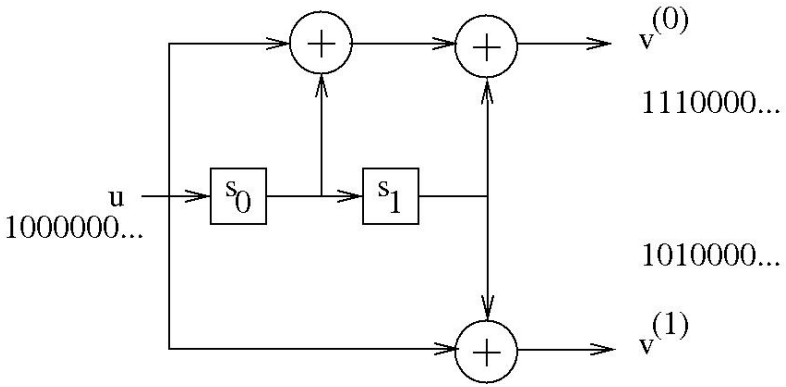
Générateur de séquences (ou générateurs) du code convolutif

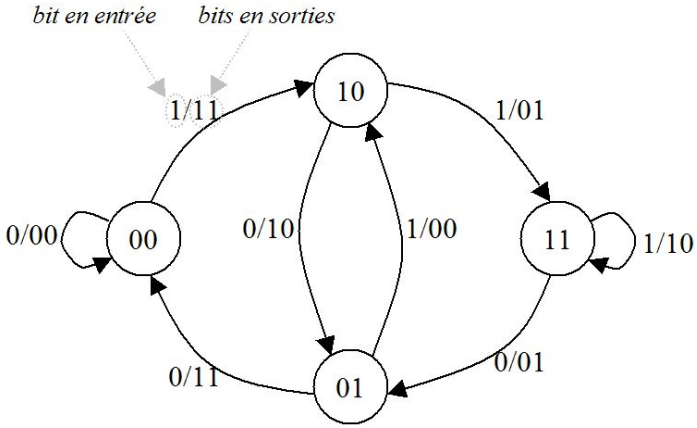
Générateur de séquences (ou générateurs)

Soit $g_0, ..., g_{n-1}$ les réponses impulsionelles d'un codeur convolutif $1/n$-taux. Par réponse impulsionelle on entend séquences générées pour l'ensemble des sorties $v^{(j)}$, $j = 0, ..., n-1$ lorsque le codeur prend en entré une impulsion $u = (10000...)$. Les réponses impulsionelles sont appelés **générateurs**.



Exemple : Générateur de séquences (ou générateurs) du code convolutif

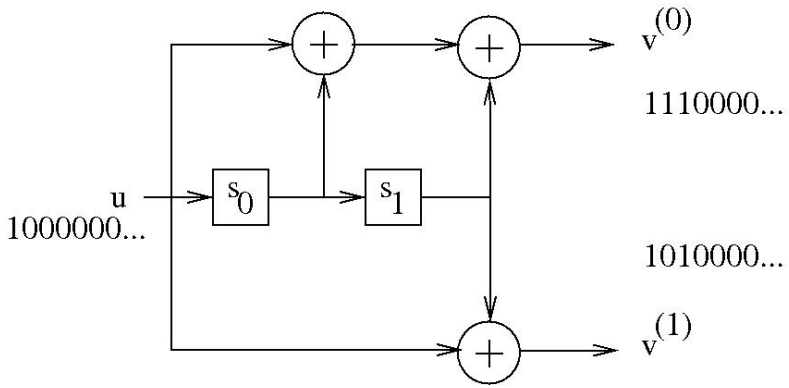




Pour le codeur convolutif 2-mémoires 1/2-taux, les générateurs sont $g_0 = (1, 1, 1)$ et $g_1 = (1, 0, 1)$. Remarquons que les générateurs de séquences valent 0 après K bits (K : longueur de contrainte).

Expression des générateurs par des polynomes

Les générateurs sont écrits sous forme de polynômes en D : $g_0(D)$, ..., $g_{n-1}(D)$.



Pour l'exemple, $g_0(D) = 1 + D + D^2$ et $g_1(D) = 1 + D^2$.

Un peu d'histoire

Quand on fait référence à un code convolutif, les générateurs sont exprimés sous la forme $(g_0, ..., g_{n-1})$ et écrit sous forme octale. Pour l'exemple précédent le générateur est (7,5). Le plus utilisé des codeurs convolutifs est le 6-mémoires 1/2-taux de générateur (171, 133). C'est le **code standard NASA** qui à été utilisé dans un premier temps pour les missions de Voyager dans l'espace. Il est également utilisé par exemple dans le standard CCSD (communication digital).

Rappel : forme binaire $\leftrightarrow$ forme octale

binaire	octale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Exemple : $(57)_8 = (101111)_2$

Les polynômes générateurs sont donnés sous la forme $g_j(D)$ avec $0 \leq j < n$:

$$g_j(D) = g_j[0] + g_j[1]D + g_j[2]D^2 + ... + g_j[m]D^m$$

Les séquences en sorties $v^{(j)}[i]$ sont égales à la **convolution discrète** entre la séquence d'entrée $u(D)$ et les polynômes générateurs $g_j(D)$ (ce qui explique le nommage 'code convolutif') :

$$v^{(j)}[i] = \sum_{l=0}^{m} u[i-l]g_j[l]$$

On peut donc écrire ceci sous la forme d'une multiplication matricielle :

$$v = u.G.$$

Matrice génératrice

$$v^{(j)}[i] = \sum_{l=0}^{m} u[i-l]g_j[l]$$

donc dans le cas d'un code convolutif 1/2-taux m-mémoires :

$$G = \left(\begin{array}{cc|cc|cc|cc|cc|cc} g_0[0] & g_1[0] & g_0[1] & g_1[1] & \dots & \dots & g_0[m] & g_1[m] & 0 & 0 & \dots & \dots \\ 0 & 0 & g_0[0] & g_1[0] & \dots & \dots & \dots & \dots & g_0[m] & g_1[m] & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & \dots & g_0[1] & g_1[1] & \dots & \dots & g_0[m] & g_1[m] \\ & & & & & & \dots & & & & & \end{array}\right)$$

Illustration pour le codeur 2-mémoires 1/2-taux

Pour le codeur convolutif 2-mémoires 1/2-taux, les générateurs sont $g_0 = (1, 1, 1)$ et $g_1 = (1, 0, 1)$.

$$G = \left(\begin{array}{cc|cc|cc|cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 & & & & & & \\ & & 1 & 1 & 1 & 0 & 1 & 1 & & & & \\ & & & & 1 & 1 & 1 & 0 & 1 & 1 & & \\ & & & & & & 1 & 1 & 1 & 0 & 1 & 1 \\ & & & & & & & \ldots & & & & \end{array}\right)$$

Si $u = (110100)$ alors $v = u.G = (11, 01, 01, 00, 10, 11)$.

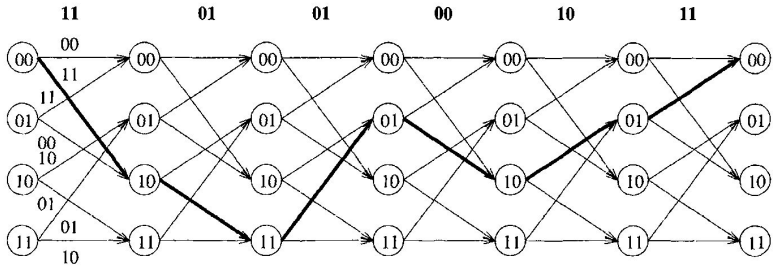


Figure 33 A path in the trellis of a memory-2 rate-1/2 convolutional encoder.

Plan

Distance libre

Distance libre

La distance libre d_f d'un code convolutif est la plus petite distance entre deux séquences codées de longueurs supérieures à la longueur contrainte K.

La distance libre peut être obtenue à partir du polynôme de distribution de poids (pas vu dans ce cours).

Plan

Décodage à maximum de probabilité

Un décodage (du mot r) à maximum de probabilité sélectionne le mot de code v qui maximise :

$$P(r/v) = \prod_{i=0}^{n-1} P(r[i]/v[i])$$

Décodage à maximum de probabilité pour un BSC

Pour un Canal Symétrique Binaire (BSC)

Pour un BSC de probabilité d'erreur p, on a (pas démontré ici) :

$$P(r/v) = \prod_{i=0}^{n-1} (1-p) \left(\frac{p}{1-p} \right)^{d_H(r[i],v[i])},$$

avec d_H la distance de Hamming.

Décodage à maximum de probabilité pour un BSC

Un décodage (du mot r) à maximum de probabilité **pour un BSC** sélectionne le mot de code v qui maximise :

$$P(r/v) = \prod_{i=0}^{n-1} (1-p) \left(\frac{p}{1-p}\right)^{d_H(r[i],v[i])},$$

- maximiser $log(P(r/v))$,
- $=$ maximiser $\sum_i d_H(r[i], v[i]).log(p/(1-p))$,
- $=$ minimiser $\sum_i d_H(r[i], v[i])$ (sachant $p \leq 0.5$ et $d_H \geq 0$).

Pour un Canal Symétrique Binaire (BSC)

Pour un BSC il faut donc choisir le mot de code **le plus proche** au sens de la **distance de Hamming**

$$d_H(r, v) = \sum_{i=0}^{n-1} d_H(r[i], v[i])$$

Décodage à maximum de probabilité pour un AWGN

Pour un Canal Additif à Bruit Blanc Gaussien (AWGN)

Pour un AWGN, on a (pas démontré ici) :

$$P(r/v) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{\pi No}} e^{-\frac{1}{No}(r[i]-m(v[i]))^2}$$

avec la fonction de modulation : $m : \{0,1\} \rightarrow \{-\sqrt{E}, +\sqrt{E}\}$

Décodage à maximum de probabilité pour un AWGN

Un décodage (du mot r) à maximum de probabilité sélectionne le mot de code v qui maximise

$$P(r/v) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{\pi No}} e^{-\frac{1}{No}(r[i]-m(v[i]))^2}$$

- maximiser $log(P(r/v))$,
- $=$ maximiser $\sum_i -\frac{1}{No}(r[i] - m(v[i]))^2$,
- $=$ minimiser $\sum_i (r[i] - m(v[i]))^2$.

Pour un Canal Additif à Bruit Blanc Gaussien (AWGN)

On montre qu'il faut choisir le mot de code le **plus proche** au sens de la **distance Euclidienne**.

$$d_E(r, v) = \sum_{i=0}^{n-1} (r[i] - m(v[i]))^2$$

Plan

Décodeur de Viterbi

L'algorithme de Viterbi permet à la réception d'un mot r de déterminer le mot de code v le plus proche (à maximum de probabilité).

Décodeur de Viterbi

L'algorithme de Viterbi est un algorithme de programmation dynamique et prend en compte la structure répétitive du treillis.

Remarque : L'algorithme de Viterbi est un algorithme de calcul de chemin le plus court (distance = poids de Hamming ou distance = distance Euclidienne ou ...) et est à peu de chose près l'algorithme de Dijkstra (algorithmique des graphes).

Illustration du déroulement de l'algorithme de décodage

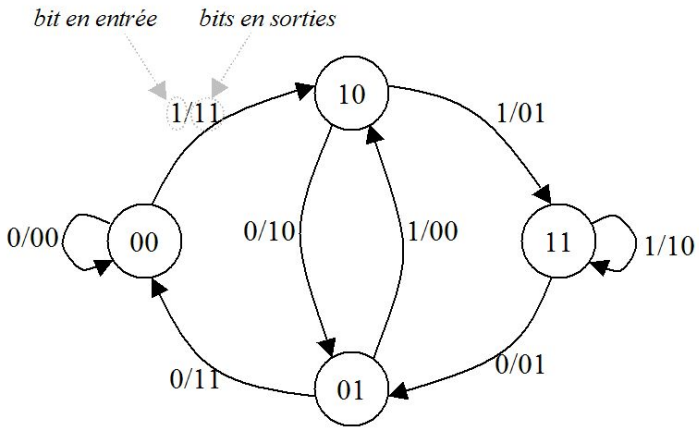


Figure: Diagramme d'état d'un codeur convolutif *2-memoires 1/2-taux*

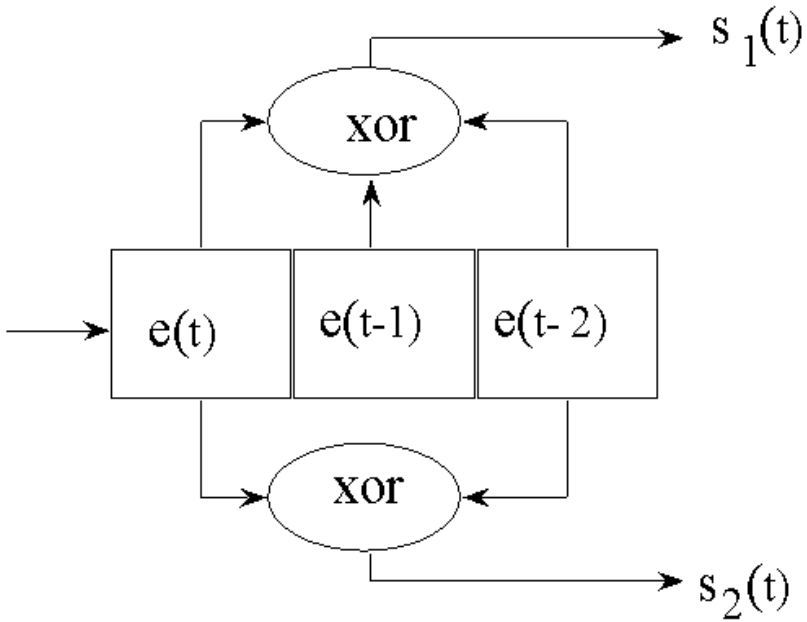
Fonctionnement de l'algorithme de Viterbi

J. Le Roux

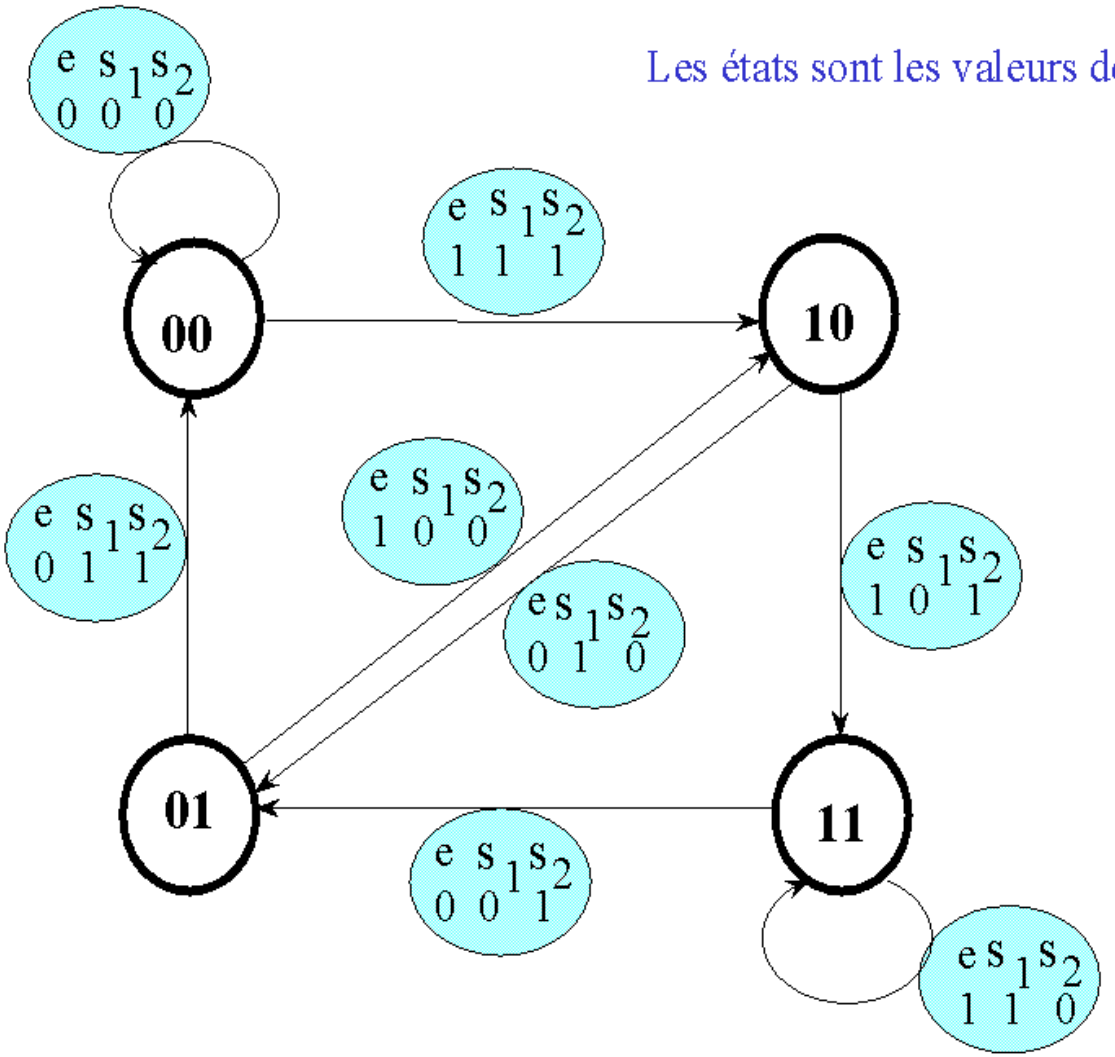
leroux@essi.fr

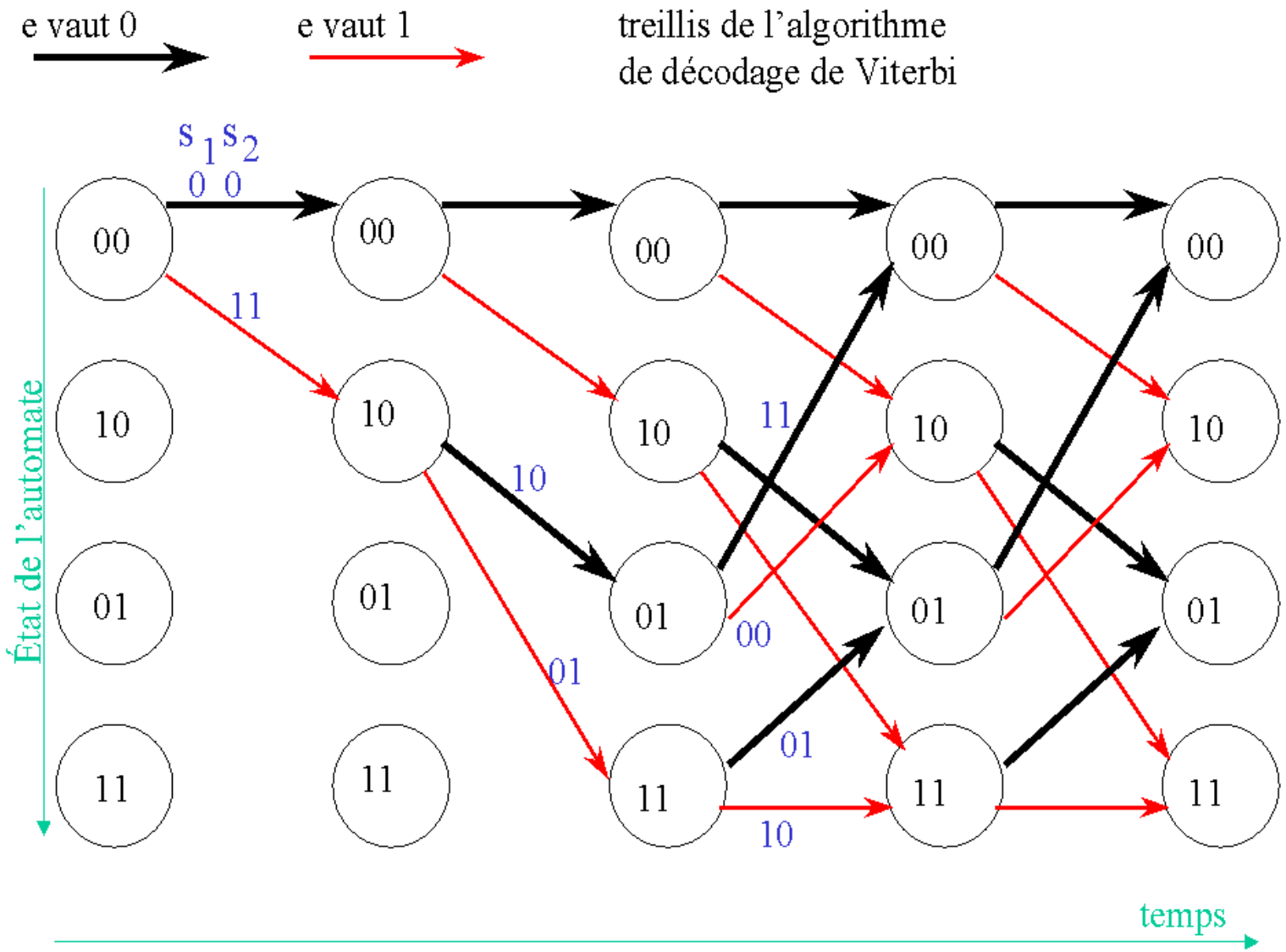
http://www.essi.fr/~leroux

Émission des donnees dans
un codeur convolutionnel
avec un registre a decalage
et des ou exclusifs

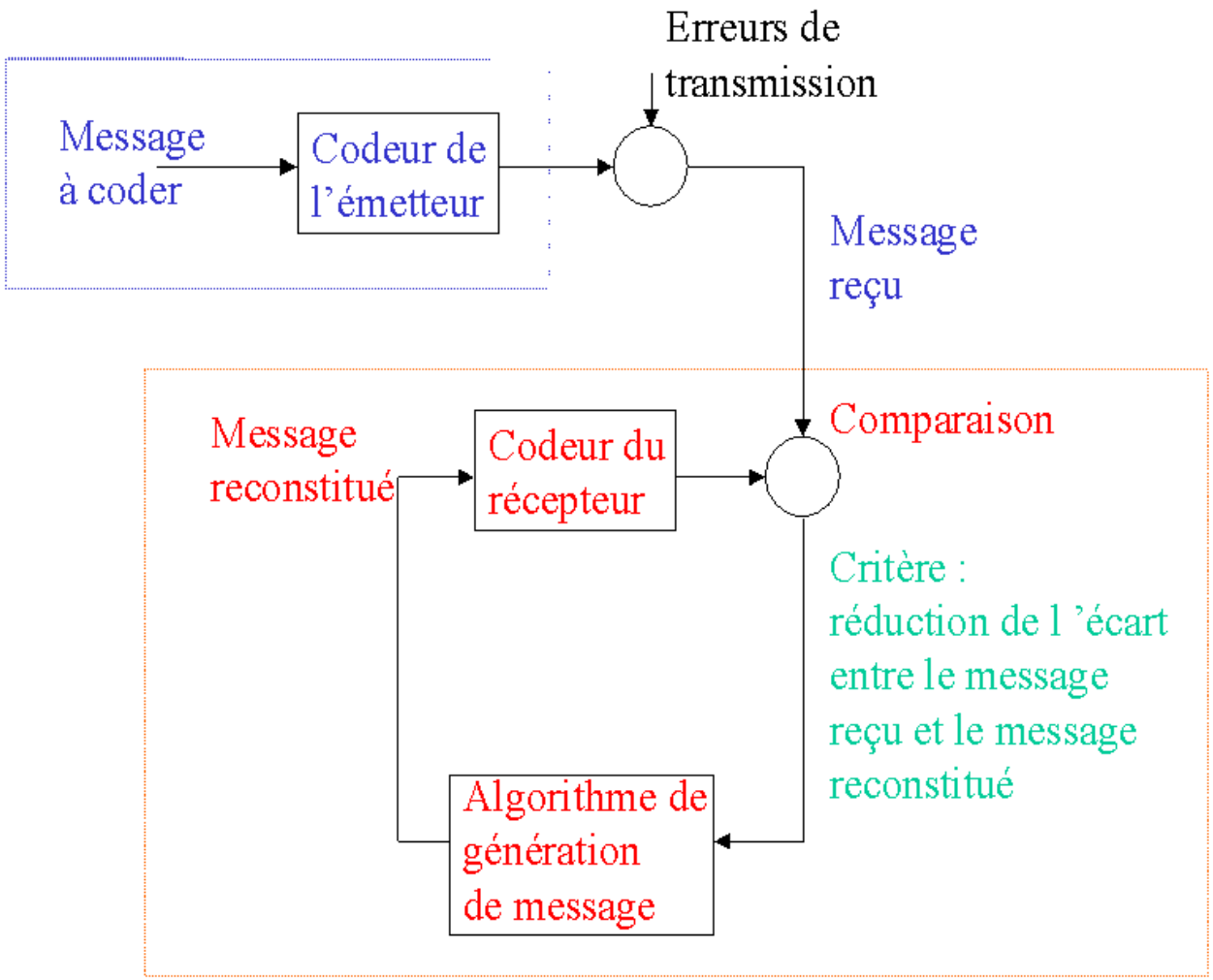


Représentation de l 'émetteur sous la forme d 'un automate




e vaut 0
e vaut 1
treillis de l'algorithme
de décodage de Viterbi
s 1 s 2
0 0
00
10
01
11
État de l'automate
temps

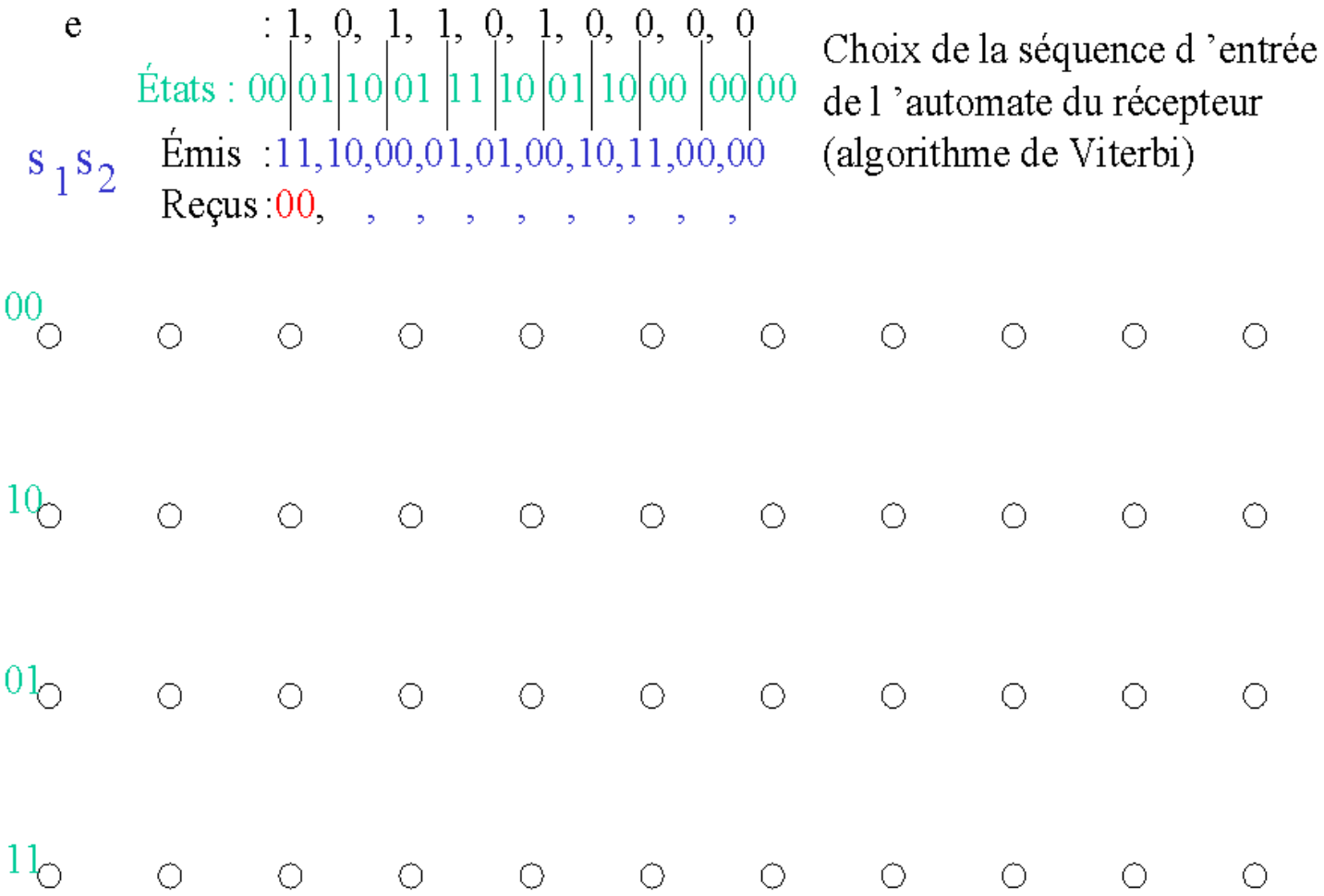
Correction des codes convolutionnels



Nous allons dérouler le décodeur dans le cadre où les 2 premiers bits ont été erronés lors de la transmition :

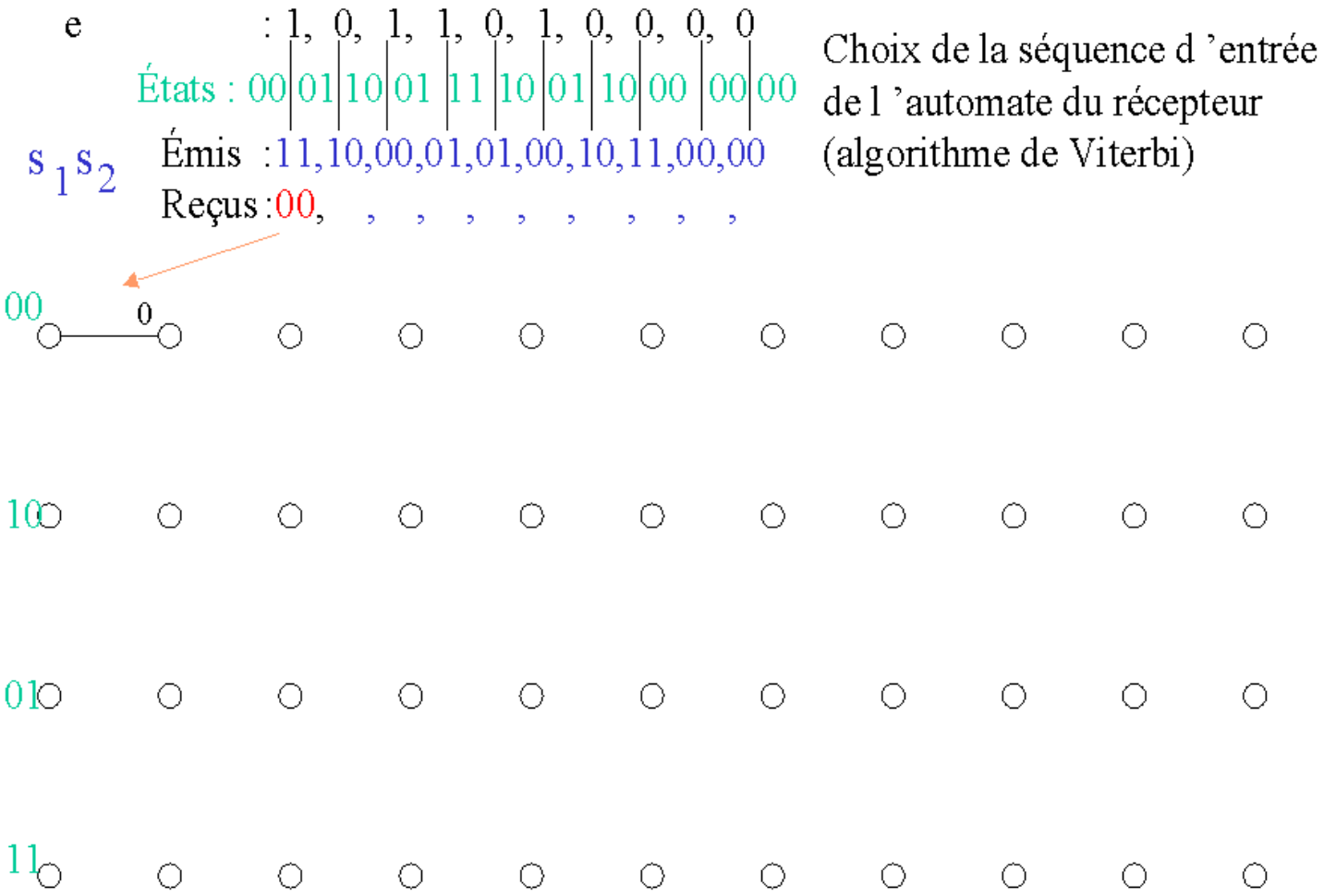
- Le mot de code à l'émission est 11 10 00 01 01 00 10 11 00 00,
- Le mot reçu est 00 10 00 01 01 00 10 11 00 00;

Le codeur (et le décodeur) partent dans l'état 0 et retourne à l'état 0 en fin de codage. Puisque l'on force le codeur à revenir dans l'état 0, 2 zéros ont donc été ajoutés au message avant codage (pour le codeur 2-mémoires 1/2-taux).



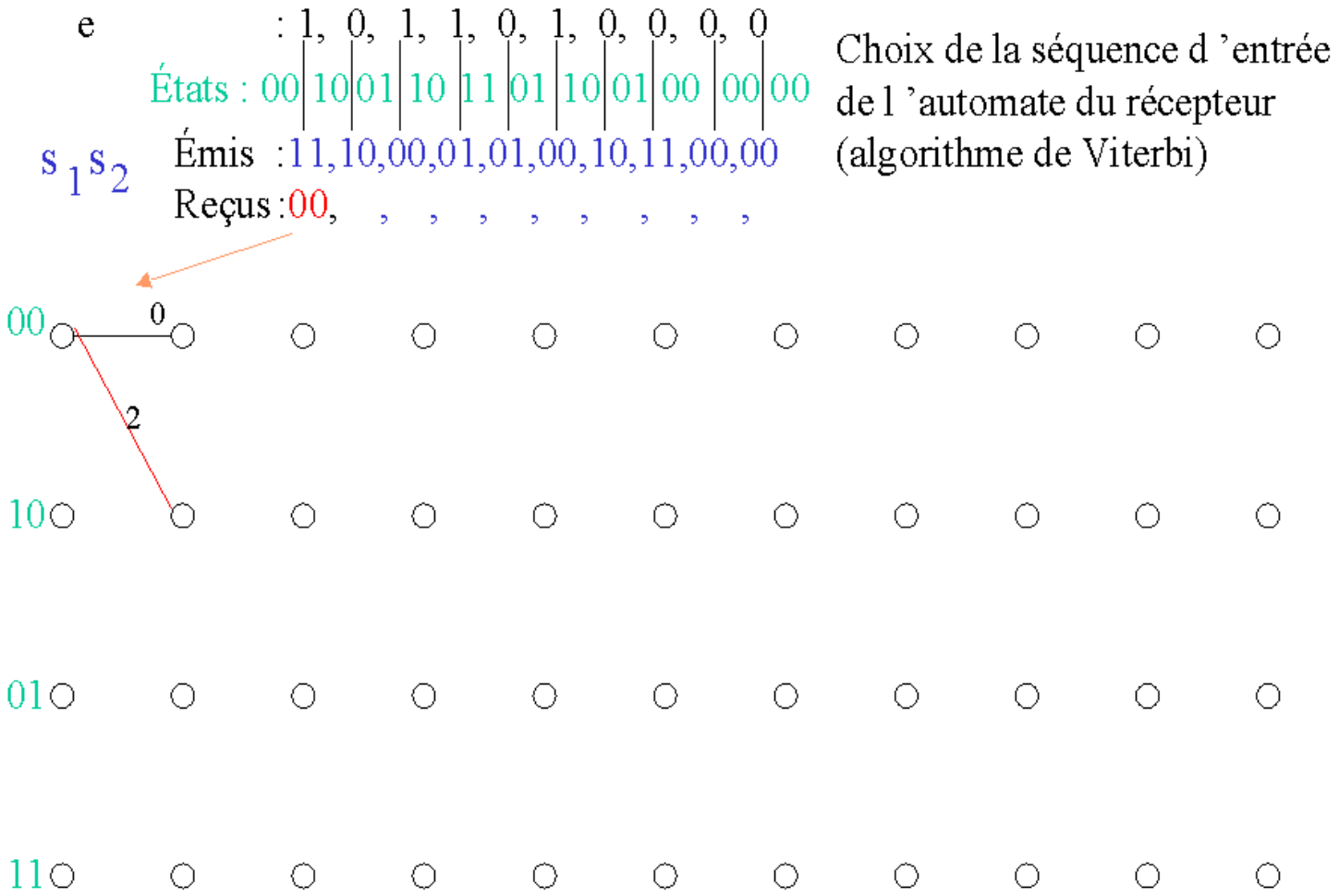
Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Éléments du treillis que va parcourir la représentation du message en cours de décodage



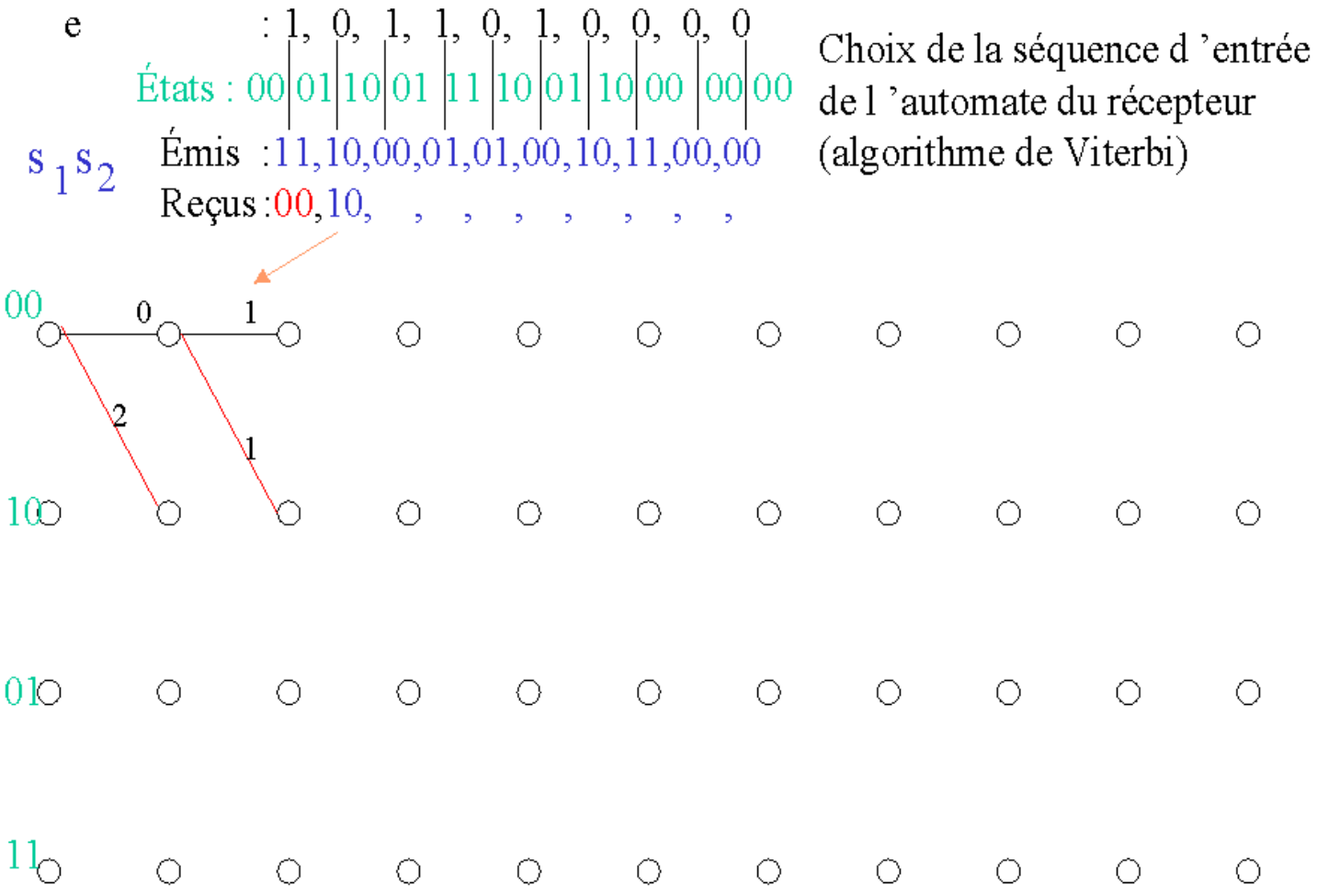
Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Choix d'une première hypothèse et comparaison aux données reçues



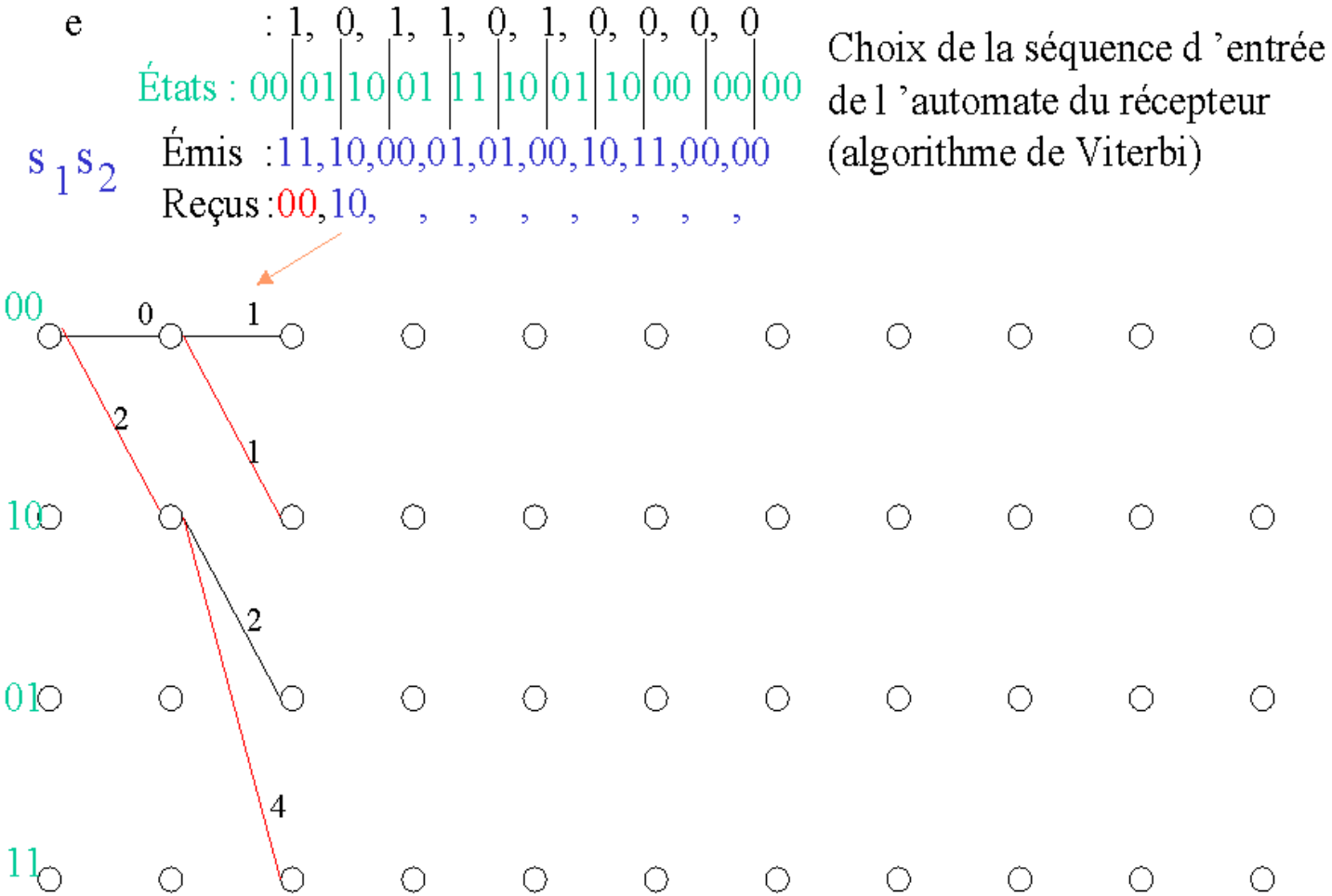
Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Choix d'une deuxième hypothèse et comparaison aux données reçues



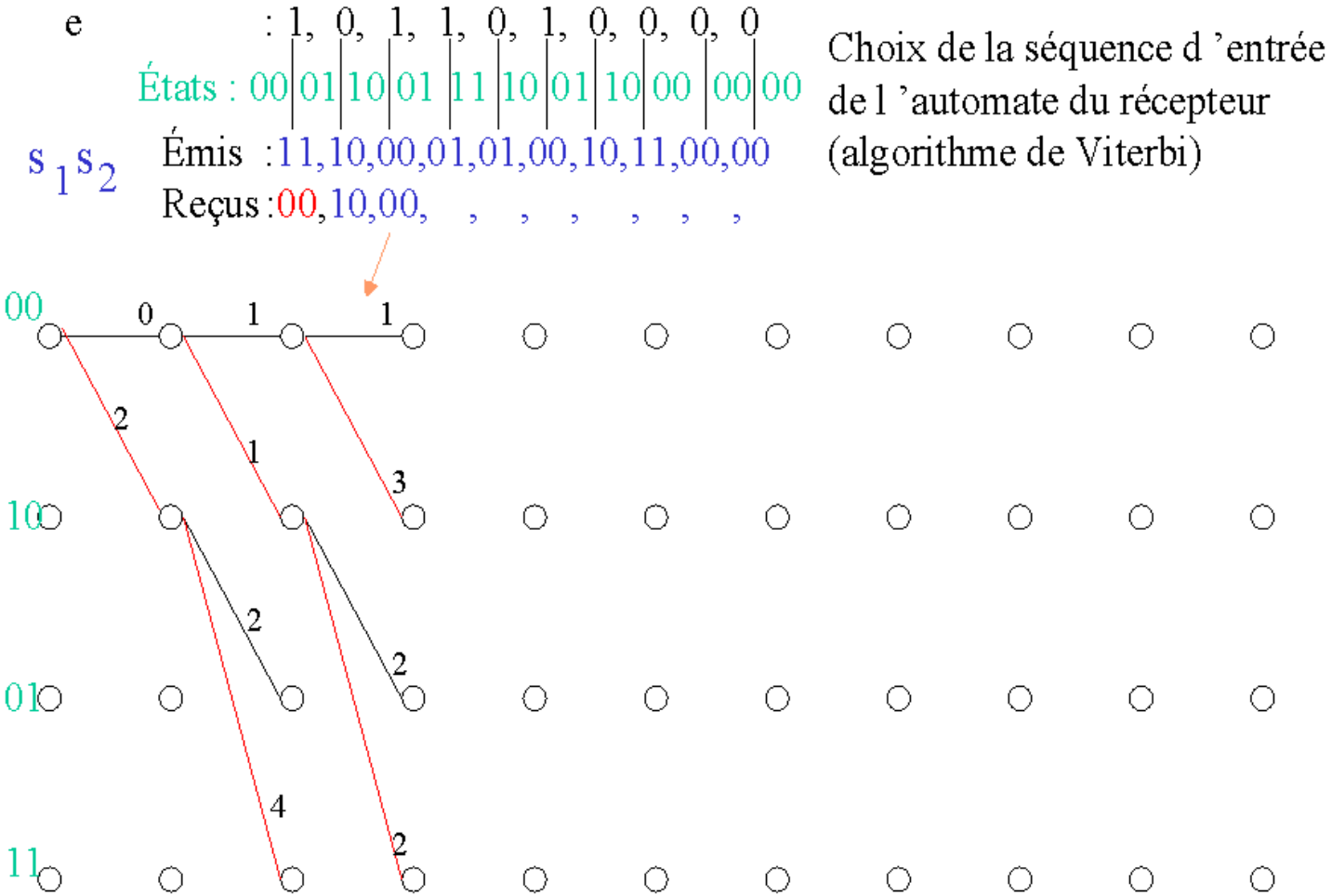
Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Itération du processus pour les deuxièmes données



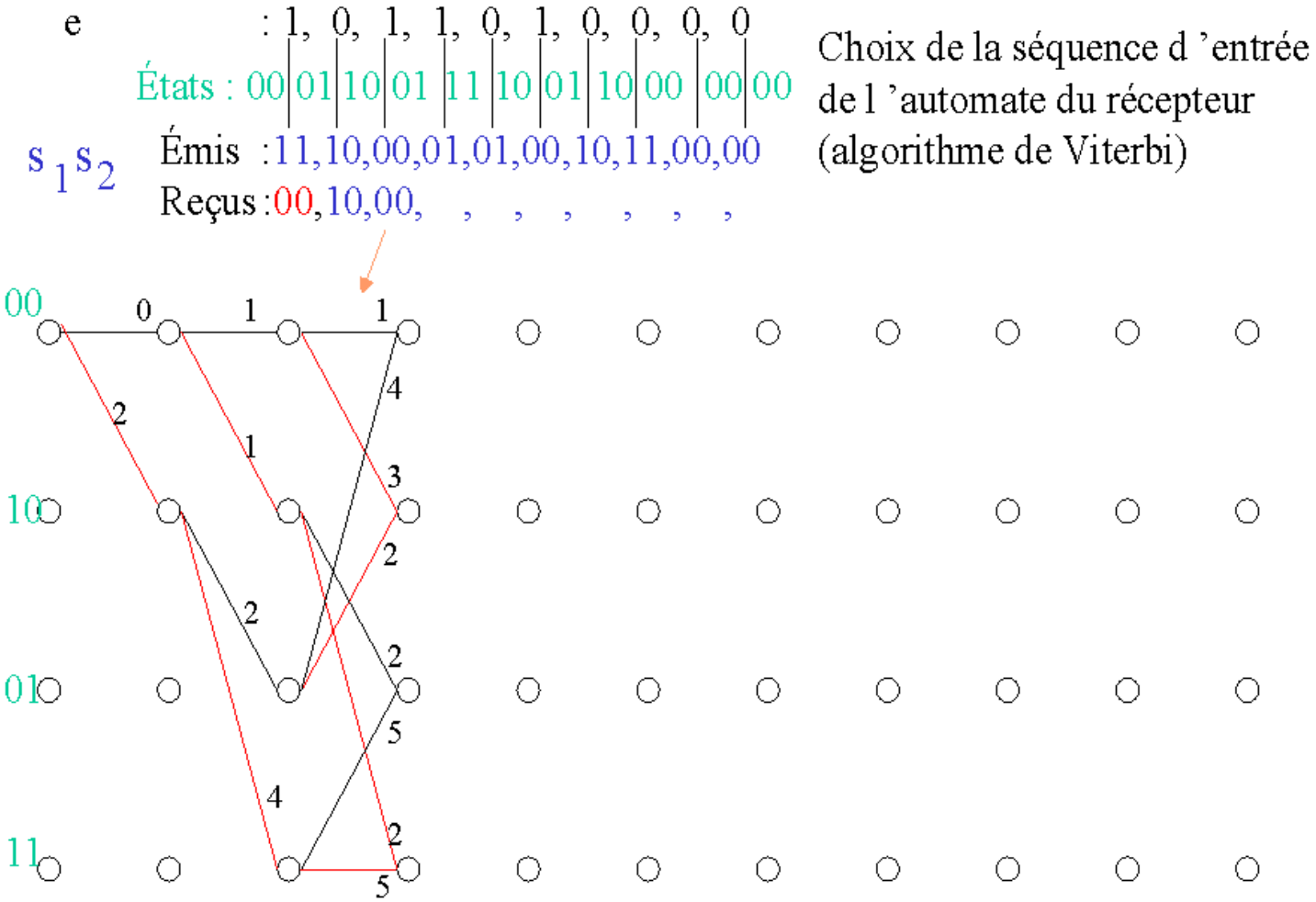
Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Itération du processus pour les deuxièmes données



Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Itération du processus pour les troisièmes données



Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)

Itération du processus pour les troisièmes données

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

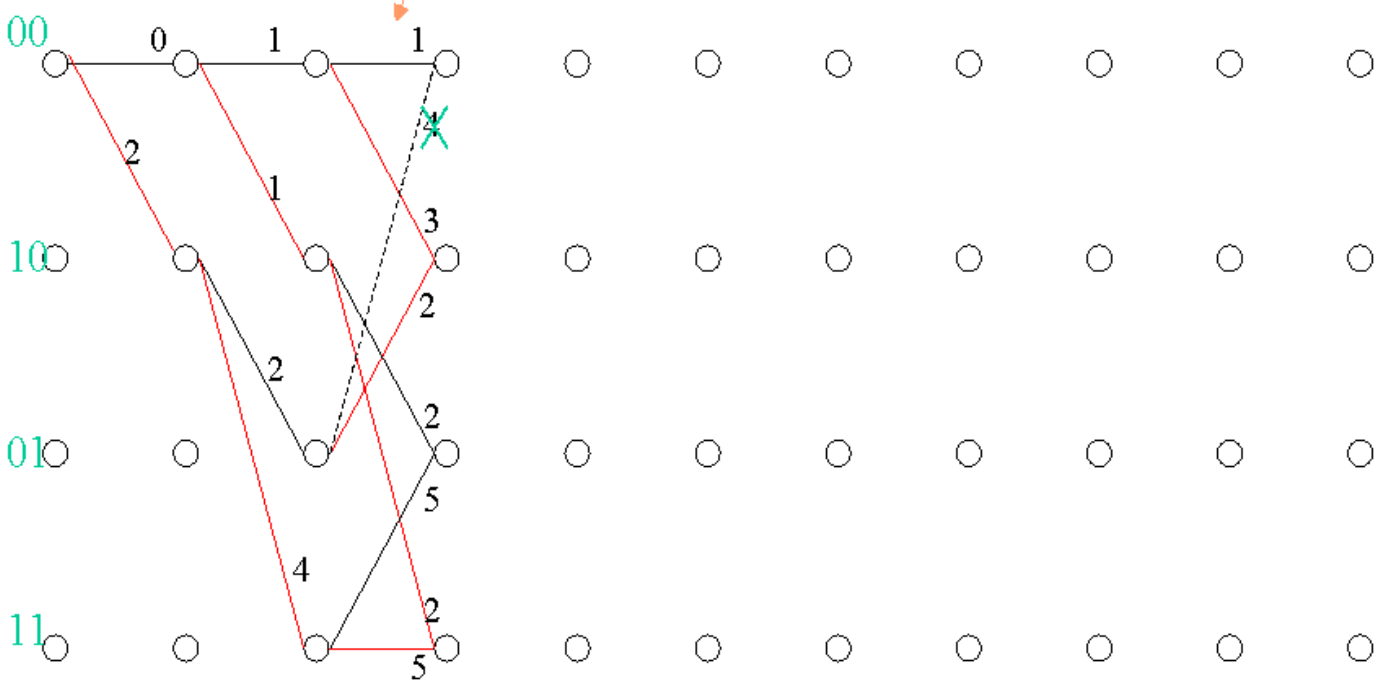
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00, , , , , , , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



En régime stationnaire il y a deux entrées par nœud du treillis :
on choisit de retenir celle qui correspond à la distance la plus faible au message reçu

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

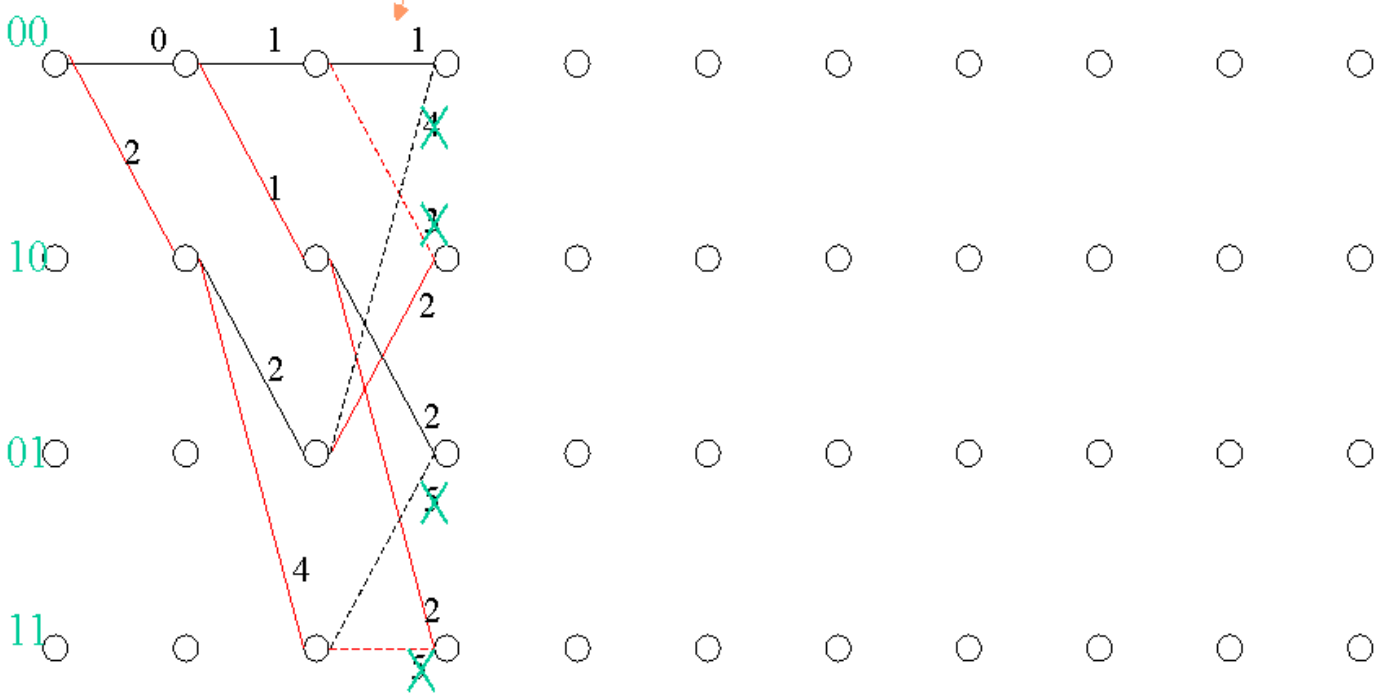
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00, , , , , , , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Choix similaires pour tous les nœuds du treillis

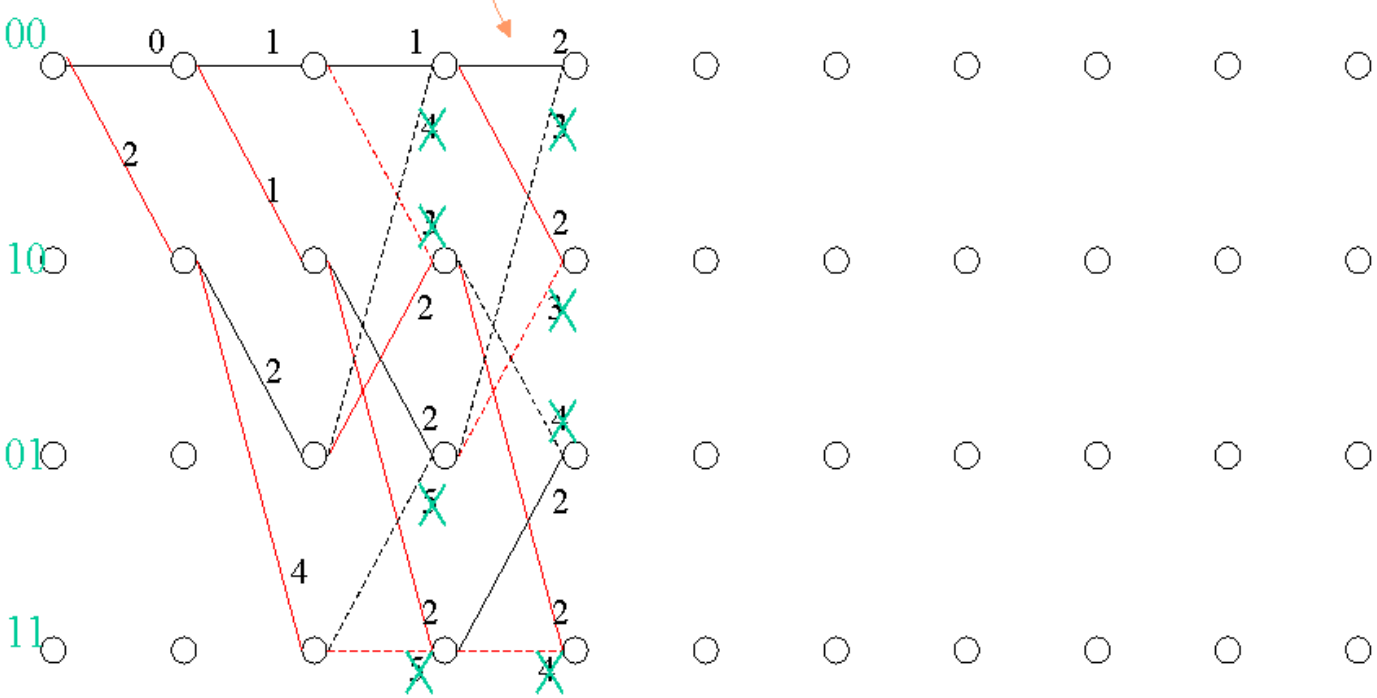
e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$ Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01, , , , , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Itération du processus : calcul des distances pour les différentes hypothèses à partir de chacun des états ; puis sélection des chemins entrants les moins coûteux pour les nouveaux états

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

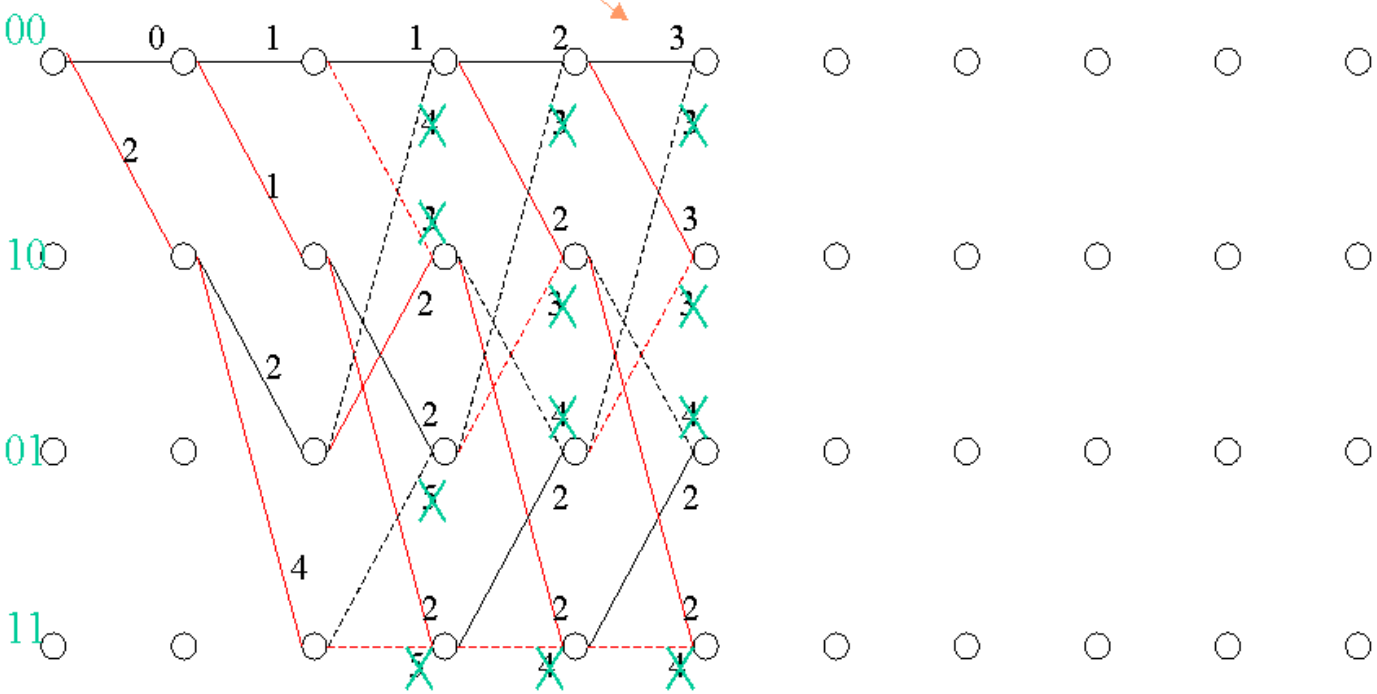
États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01, , , , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Itération du processus : calcul des distances pour les différentes hypothèses à partir de chacun des états ; puis sélection des chemins entrants les moins coûteux pour les nouveaux états

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

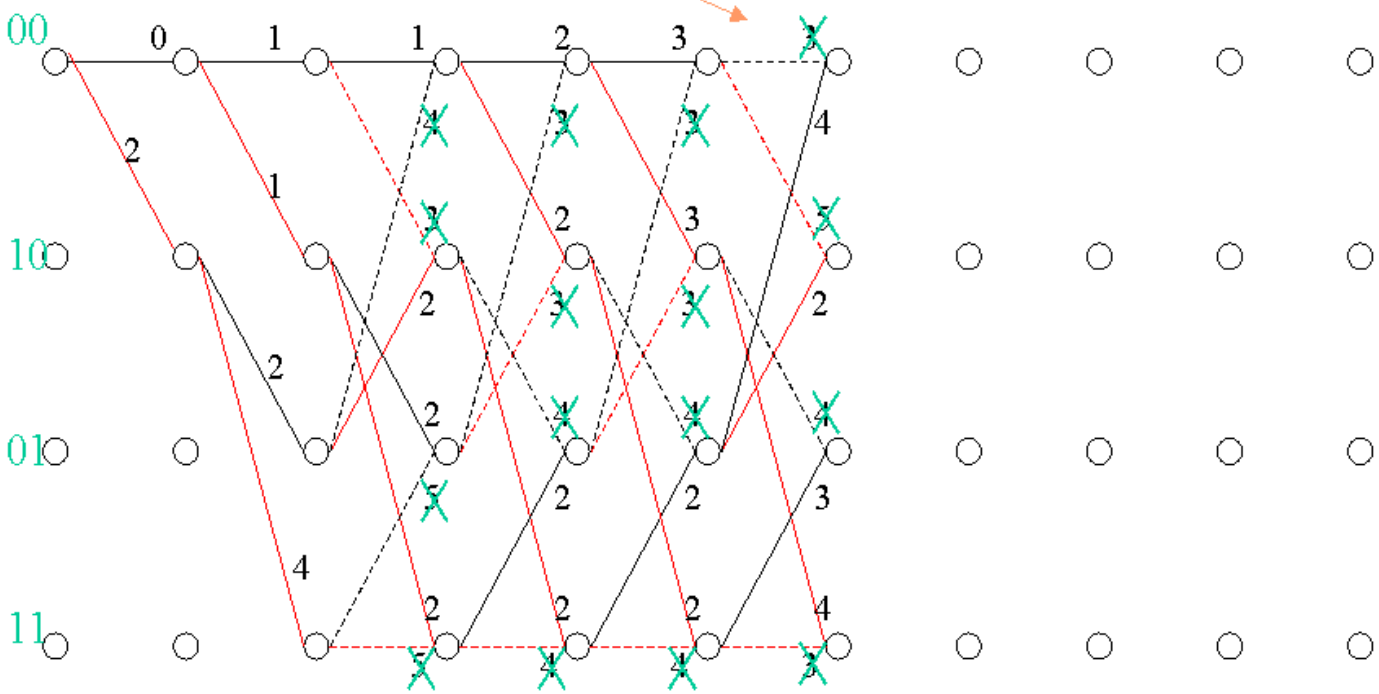
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00, , , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Itération du processus : calcul des distances pour les différentes hypothèses à partir de chacun des états ; puis sélection des chemins entrants les moins coûteux pour les nouveaux états

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

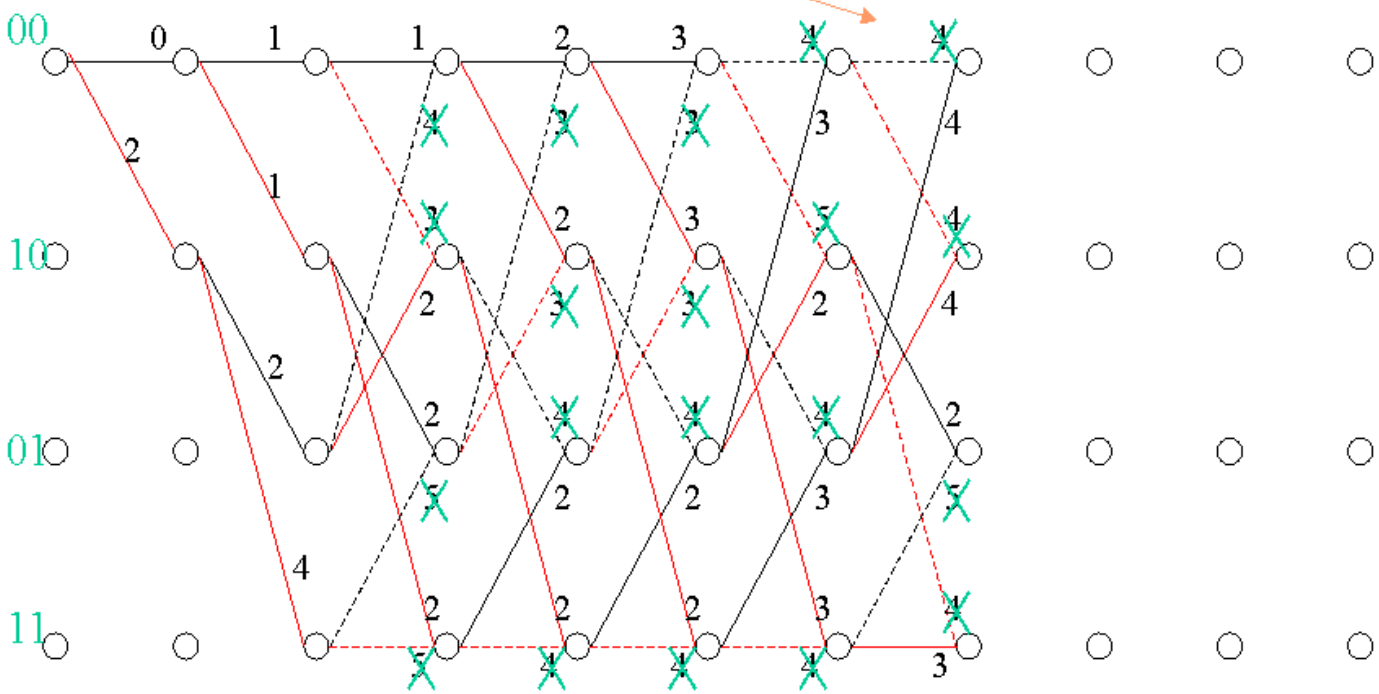
États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10, , ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Itération du processus : calcul des distances pour les différentes hypothèses à partir de chacun des états ; puis sélection des chemins entrants les moins coûteux pour les nouveaux états

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

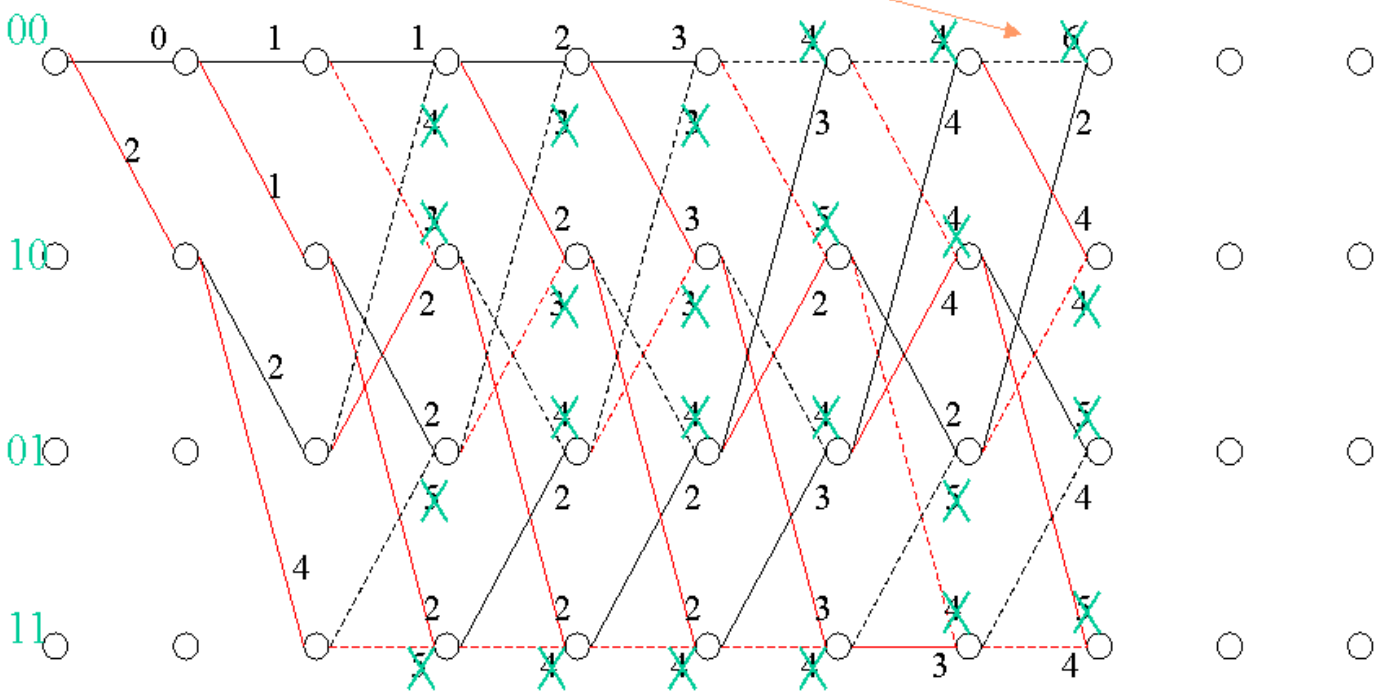
États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11, ,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Itération du processus : calcul des distances pour les différentes hypothèses à partir de chacun des états ; puis sélection des chemins entrants les moins coûteux pour les nouveaux états

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

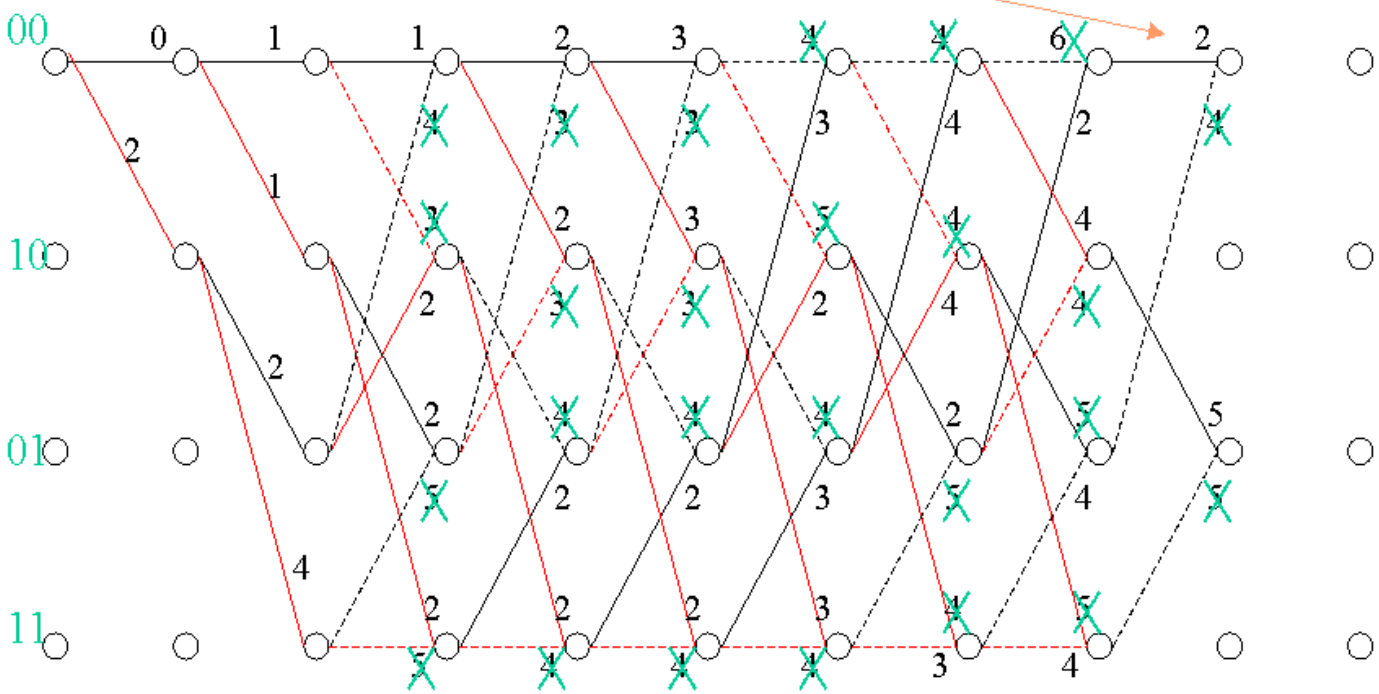
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Les deux dernières entrées sont telles que l'état final est bien défini (ici l 'état 00)

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

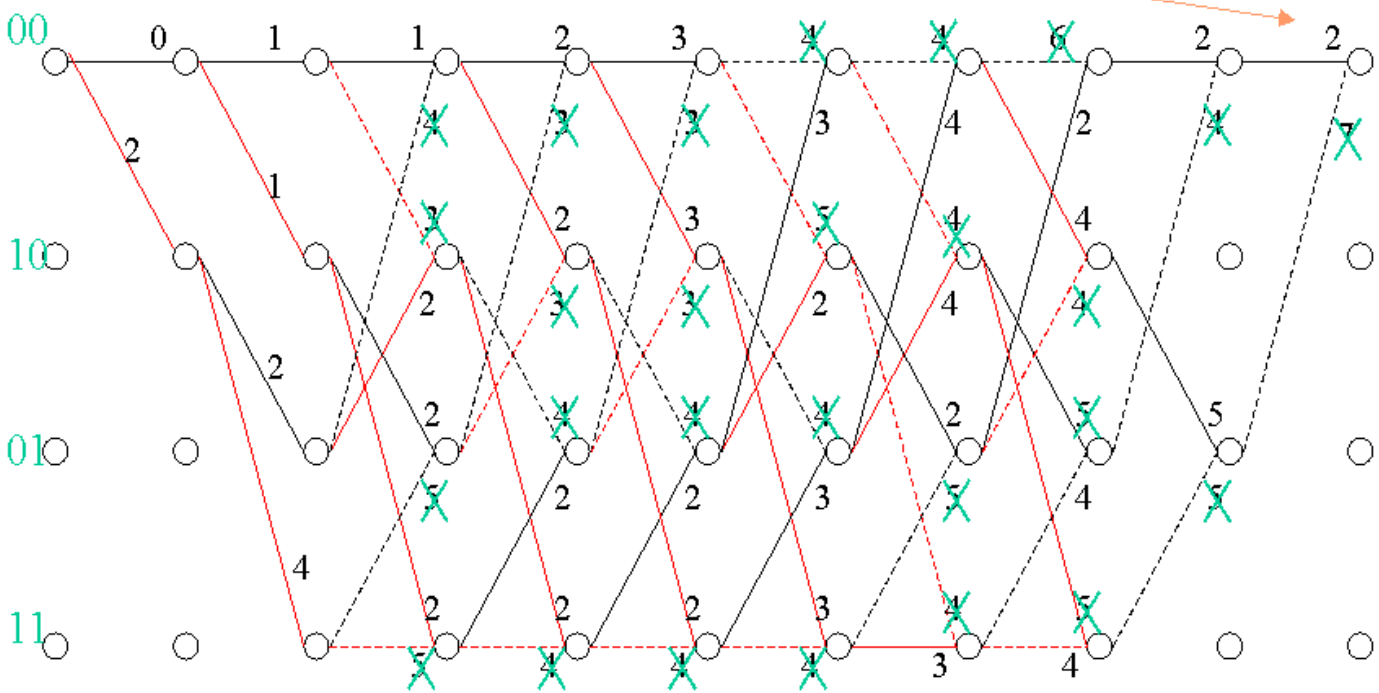
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Les deux dernières entrées sont telles que l'état final est bien défini (ici l 'état 00)

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

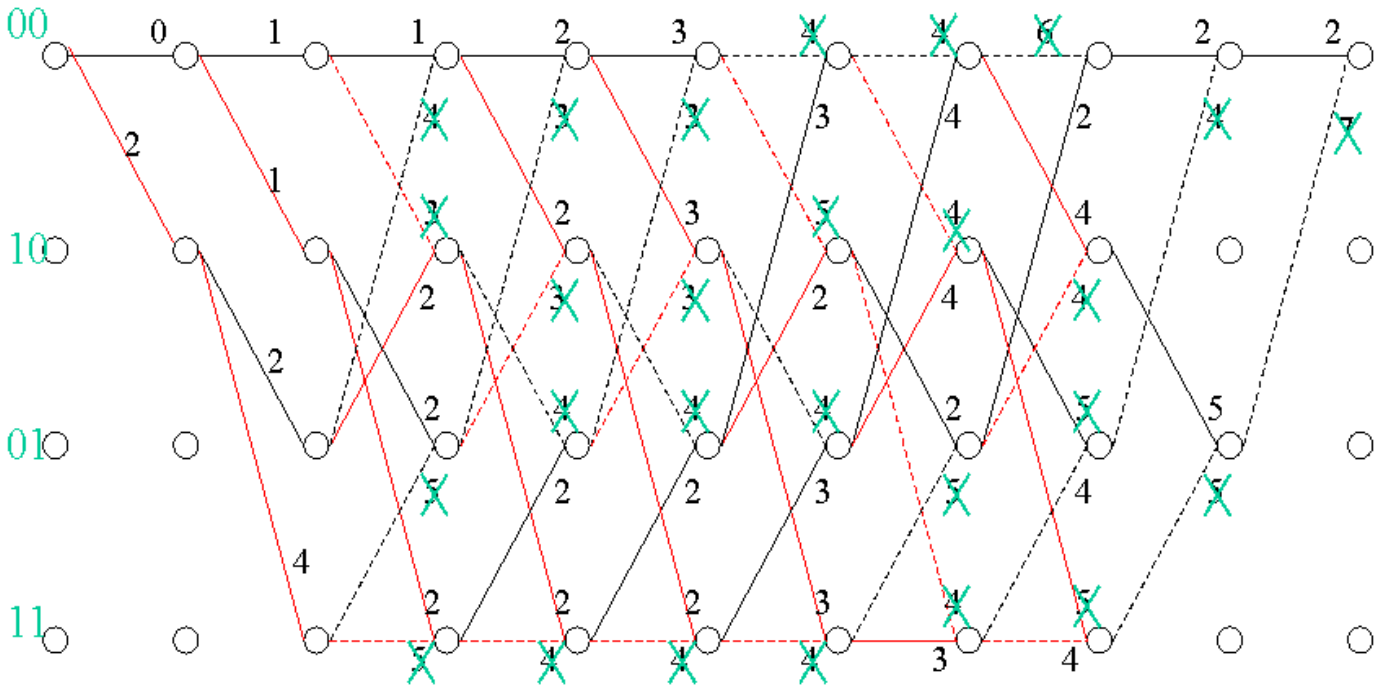
États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Traits continus : chemins autorisés; traits pointillés : chemins interdits

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

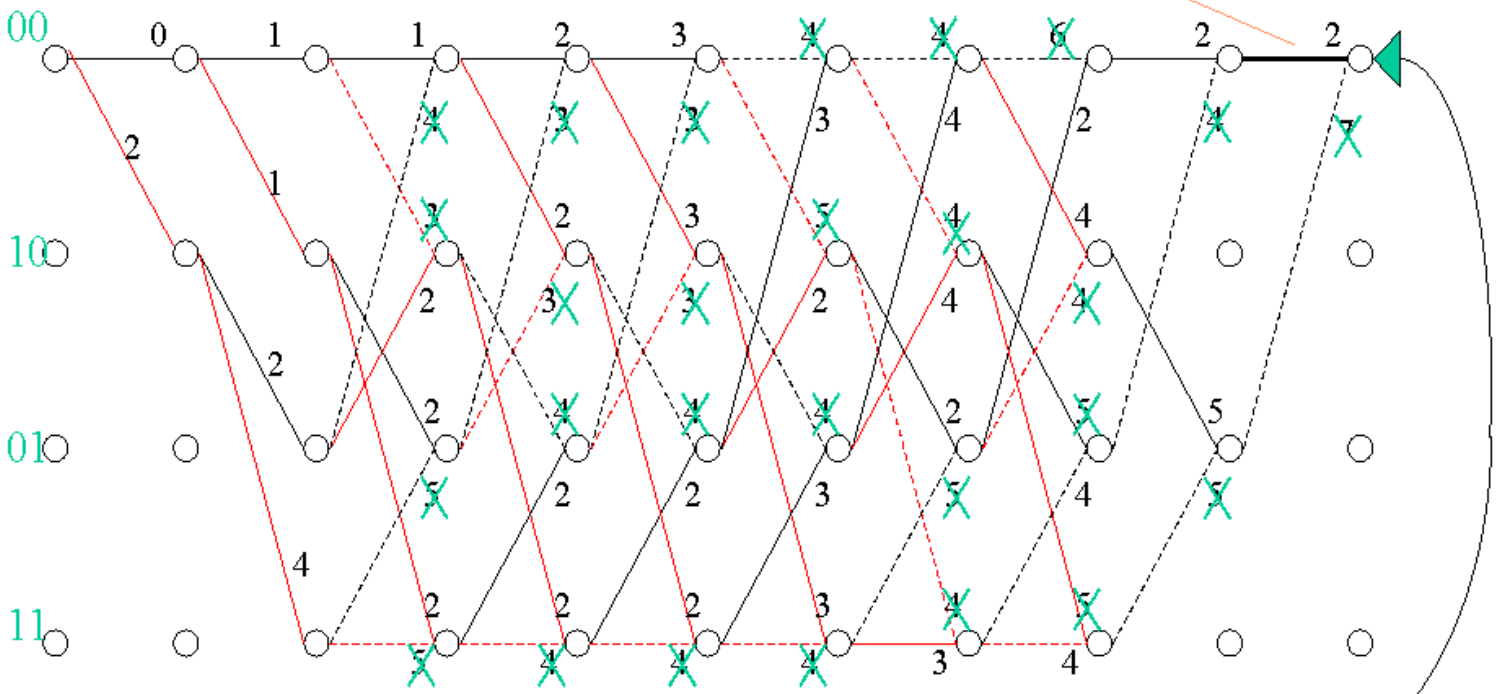
États : 00|01|10|01|11|10|01|10|00|00|00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



Parcours du graphe en remontant à partir de la fin

e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

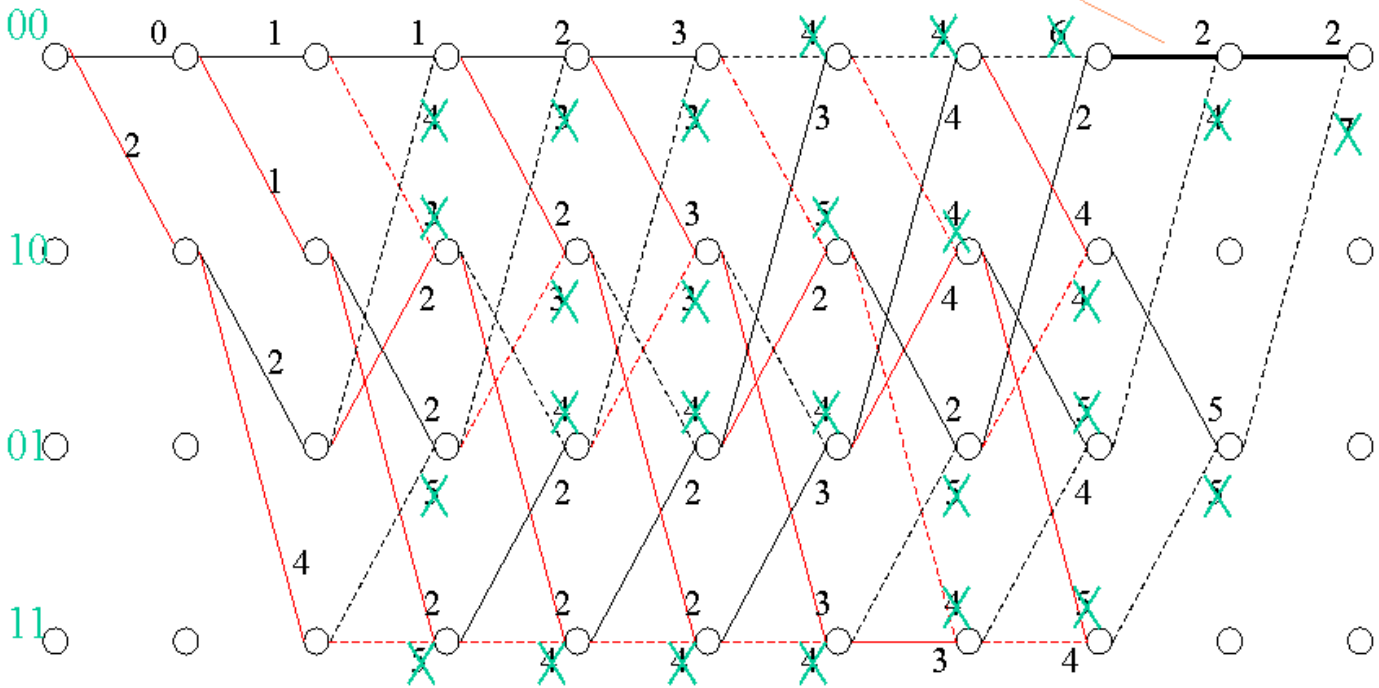
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

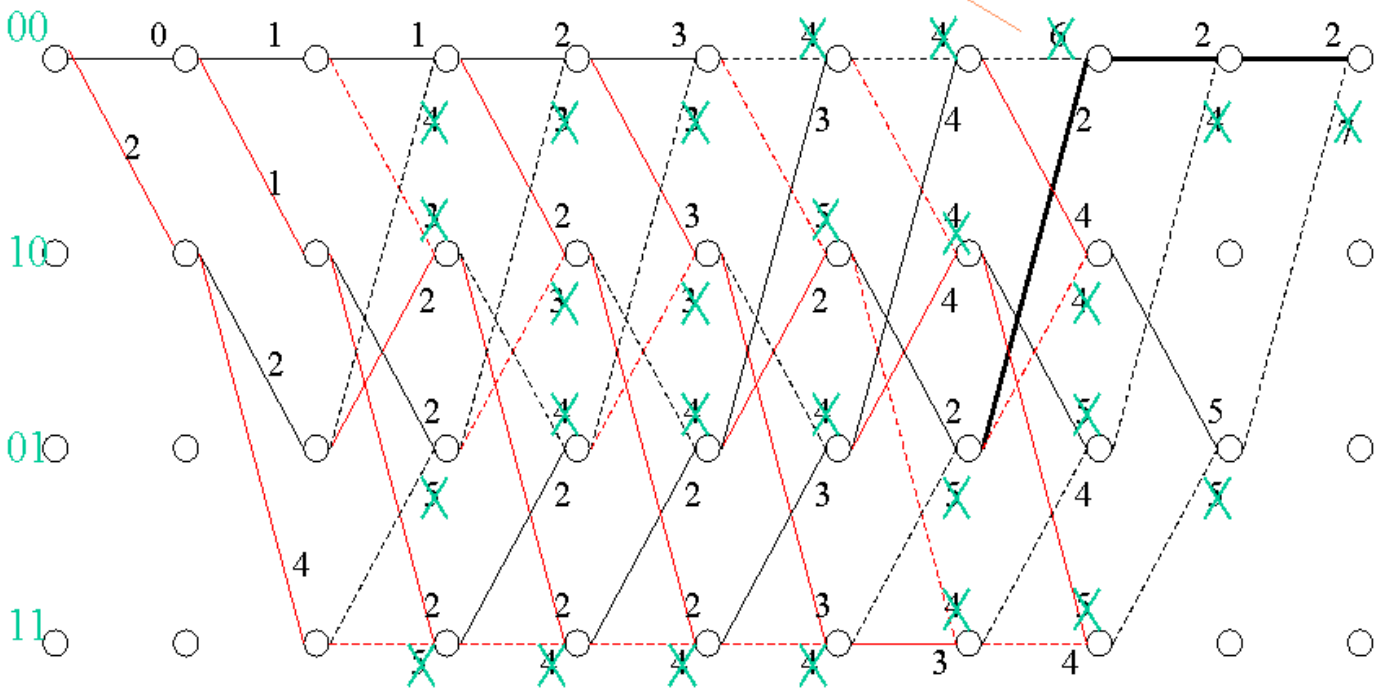
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

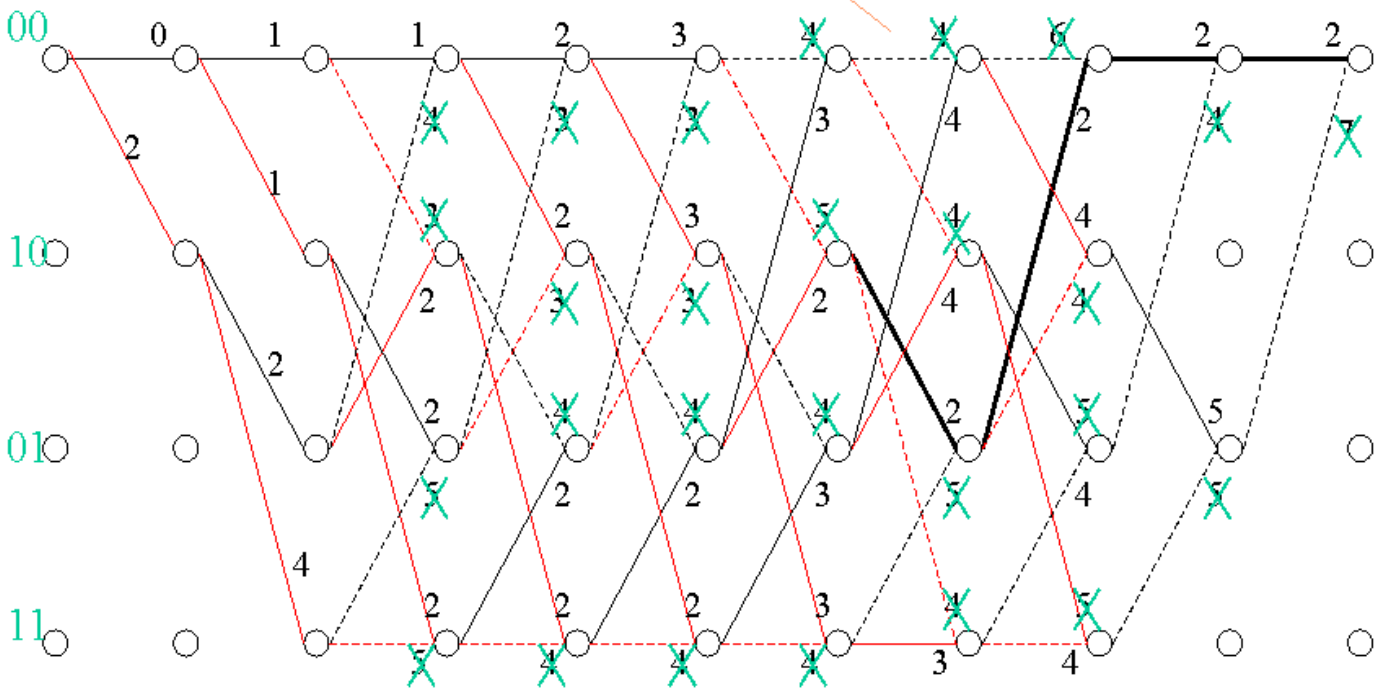
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)



e : 1, 0, 1, 1, 0, 1, 0, 0, 0, 0

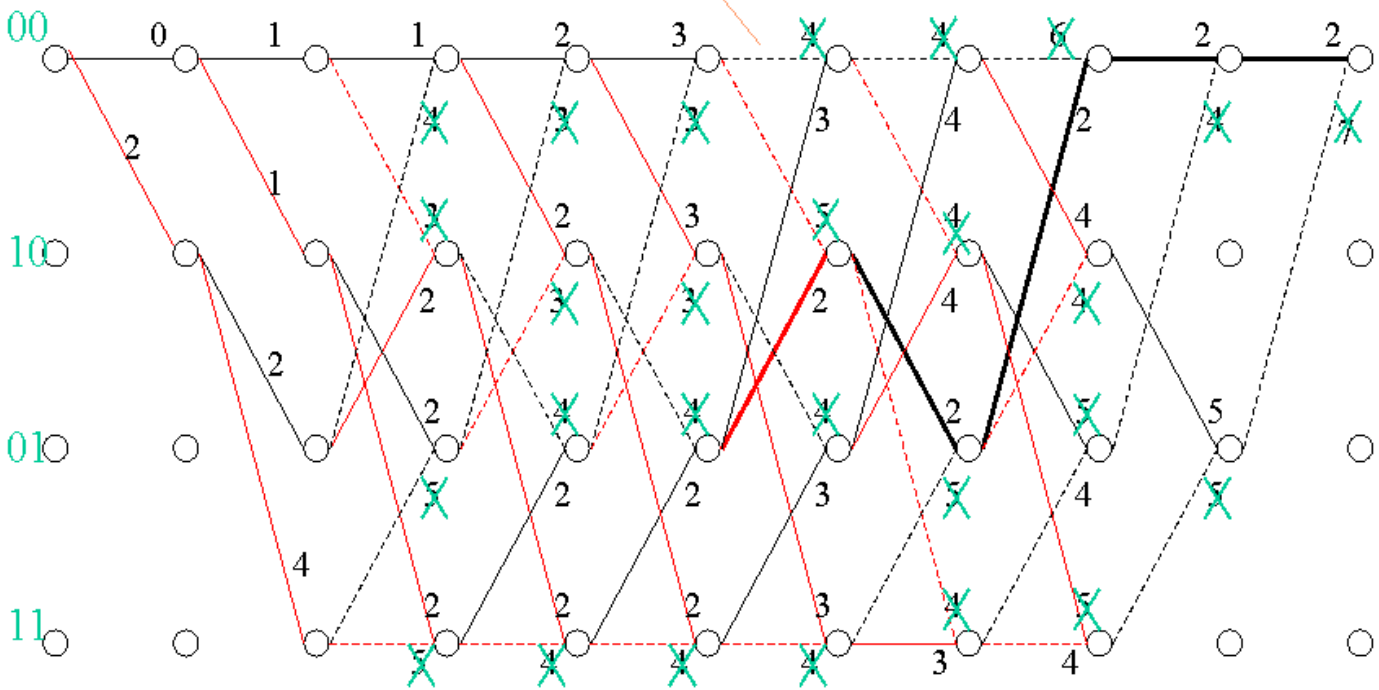
États : 00 | 01 | 10 | 01 | 11 | 10 | 01 | 10 | 00 | 00 | 00

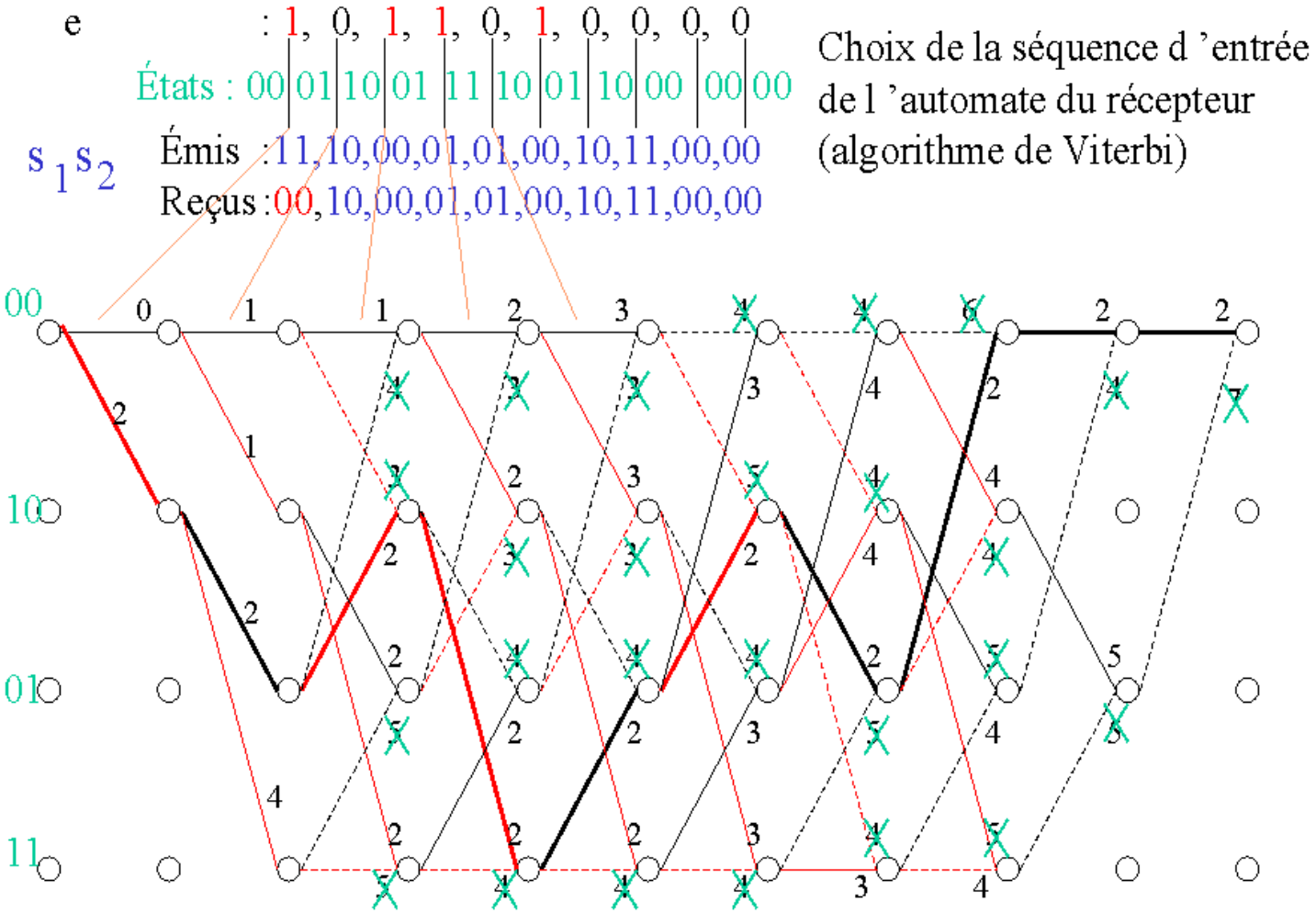
$s_1 s_2$

Émis : 11,10,00,01,01,00,10,11,00,00

Reçus : 00,10,00,01,01,00,10,11,00,00

Choix de la séquence d 'entrée de l 'automate du récepteur (algorithme de Viterbi)





Malgré l 'erreur sur les deux premières données, le chemin retrouvé correspond bien au messag émis

On vient de dérouler "à la main" l'algorithme de Viterbi. Remarque :

- le codeur et le décodeur possèdent le même treillis,
- le codeur et le décodeur s'entendent sur l'état de départ et l'état d'arrivée (plusieurs, un seul...),
- lors du décodage de Viterbi, **à chaque étape**, 2 informations sont calculées pour un état :
  - la distance minimum cumulée pour atteindre l'état,
  - le prédesseceur optimal pour atteindre l'état,
- Lorsque l'on implémente Viterbi, on optimise l'occupation mémoire. On conserve en memoire :
  - le tableau stockant pour chaque état les distances minimum cumulées à l'étape $l-1$,
  - le tableau stockant pour chaque état les distances minimum cumulée à l'étape l,
  - les 2^m (=nb d'états) listes stockant le chemin le plus court pour atteindre chaque état pour l'étape l.

Plan

Distribution de poids

Soit un code convolutif binaire m mémoires $1/n$ taux.

Matrice de transition d'état

Soit $\Omega_{ij}(x)$ la matrice $2^m \times 2^m$ défini par :

$$\Omega_{ij}(x) = \delta_{i,j} x^{h_{i,j}},$$

avec :

- $\delta_{ij} = 1$ s'il y a une transition entre l'état i et l'état j,
- $\delta_{ij} = 0$ sinon.

et h_{ij} le poids de Hamming du vecteur de sortie de longueur n.

Remarque : la variable x n'a pas vraiment de signification; ce qui nous intéresse ce sont les exposants cad les poids.

Illustration matrice de transition d'état

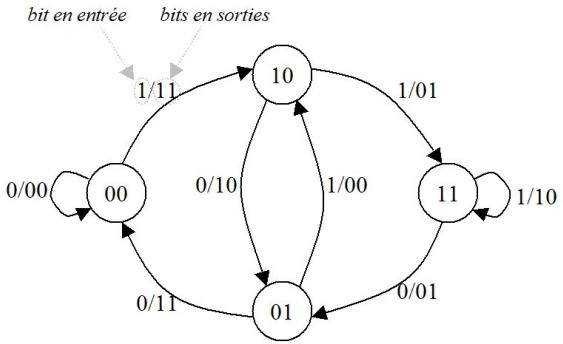


Figure: Diagramme d'état d'un codeur convolutif *2-mémoires 1/2-taux*

La matrice de transition d'état du codeur est :

$$\Omega(x) = \begin{pmatrix} 1 & 0 & x^2 & 0 \\ x^2 & 0 & 1 & 0 \\ 0 & x & 0 & x \\ 0 & x & 0 & x \end{pmatrix}$$

Distribution de poids

Élever la matrice de *transition d'état* à la puissance l revient à effectuer l transitions dans le treillis. On notera Ω^l la matrice $\Omega(x)$ élevé à la puissance l.

Distribution de poids d'un chemin

La distribution de poids d'un chemin du treillis qui débute dans l'état i et termine dans l'état j après l étapes de transition est donnée par le terme $\Omega^l_{ij}(x)$. La distribution donne la répartition de poids des différents mots de code que l'on peut obtenir en allant de l'état i à l'état j.

Remarque

Pour obtenir la distribution de poids, on peut également générer tous les mots de code et énumérer le nombre de mot de code pour les différentes valeurs de poids.

Illustration d'une distribution de poids

Soit la matrice génératrice du code linéaire $[6, 3, d_{DT}]$:

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

Calcul des poids de tous les mot de codes :

$info$	$mot-de-code$	w
000	000000	0
001	000011	2
010	001110	3
011	001101	3
100	111011	5
101	111000	3
110	110101	4
111	110110	4

La **distribution de poids** de ce code est $A(x) = 1 + x^2 + 3x^3 + 2x^4 + x^5$. La distance minimale de ce code est $d_{DT} = 2$ (car c'est le poids le plus petit $\neq 0$).

Dans le cas d'un code convolutif, pour obtenir la distribution $A(x)$ on peut également calculer la matrice de transition $\Omega(x)$, l'élever à la puissance l (=nombre de transitions) et ensuite prendre le ou les terme(s) de $\Omega^l(x)$ (cf. suite du cours...).

Plan

Remarque sur les codes convolutifs

Un code convolutif est l'ensemble des séquences produites par un codeur. En théorie, le nombre de séquences est infini. En pratique on force périodiquement le codeur à revenir dans un état connu. Les séquences ainsi produites sont d'une certaine manière en bloc.

fonctionnement classique d'un codeur

Il y a une connection entre code blocs et code convolutifs.

Il est de coutume de "casser" en blocs de tailles finie les séquences générées par un code convolutif. Pour cela, une séquence fixe de taille m est concaténée à la *séquence d'information*. Cette séquence doit être un mot unique et permet de synchroniser le décodeur et force le codeur à retourner dans un état connu.

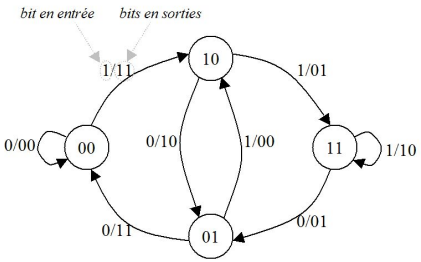
Plan

construction **en "zero-tail"**

Soit C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f.

À partir d'un vecteur d'information u, nous créons un **vecteur** u' composé de u suivi de m zéros. Soit L la taille de ce **vecteur** u'. Le codage convolutif de ce **vecteur** u' fait toujours aboutir le codeur dans l'état 0 (cf. treillis).

Le code ainsi obtenu est un code linéaire $[nL, (L-m), d_{zt}]$, avec d_{zt} la distance minimum. On notera ce code C_{ZT}.



construction **en "zero-tail"**

C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f et C_{ZT}, le code linéaire en bloc $[nL, (L-m), d_{zt}]$ issu du "zero-tail".

Propriété du code C_{ZT}

Le **taux** du code C_{ZT} est $\frac{L-m}{nL} = \frac{1}{n} - \frac{m}{nL}$ (*bit d'info/bit transmis*)et est plus faible que le taux du code convolutif $1/n$;

Par contre, si L (la taille du vecteur avant codage) est suffisamment grand et que $L > m$ on a alors d_{ZT} qui approche voir même vaut d_f. Le **pouvoir correctif** de C_{ZT} est donc similaire voir égal au code convolutif.

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux (rem : la distance libre $d_f = 5$).

Soit le vecteur u' composé de 3 bits d'information suivi de $m = 2$ zéros. Le code est alors tel que :

- les mots de code obtenus sont de taille $(3+m) \times n = (3+2) \times 2 = 10$ bits,
- la quantité d'information codée pour un tel code est de 3 bits,
- la distance minimum est égale à la distance libre.

La construction en "zero-tail" est donc un code linéaire en bloc [10, 3, 5].
La matrice génératrice est (cf transparent précédent):

$$G = \left(\begin{array}{cc|cc|cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right)$$

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux ($d_f = 5$). La construction en "zero-tail" est le code linéaire en bloc [10, 3, 5] de matrice génératrice :

$$G = \left(\begin{array}{cc|cc|cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right)$$

La **séquence de distribution de poids** de ce code est $A(x) = 1 + 3x^5 + 3x^6 + x^7$. Le poids le plus faible (en mettant à part le poids du vecteur $\underline{0}$) pour ce code est 5 ; la distance minimum (distance libre) est donc de 5.

Le taux pour ce code est de $3/10 = 0.3$ (*bit d'information/bits transmis*). Le taux pour le code convolutif intitial est meilleur ($1/2 = 0.5$ *bit d'information/bits transmis*).

Distribution de poids de C_{ZT}

Rappel : Matrice de transition d'état

Soit $\Omega_{ij}(x)$ la matrice $2^m \times 2^m$ défini par : $\Omega_{ij}(x) = \delta_{i,j} x^{h_{i,j}}$, avec h_{ij} le poids de Hamming du vecteur de sortie.

La **séquence de distribution de poids** du code C_{ZT} est $A(x) = \Omega^l_{0,0}(x)$, avec l le nombre de transitions et (0,0) les états de départ et d'arrivée dans le treillis.

Plan

Construction par troncature directe

Soit C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f.

Soit un vecteur d'information u de taille L. Le codage convolutif de ce **vecteur** u donne des mots de code qui font aboutir le codeur dans n'importe quel état.

Le code ainsi obtenu est un code linéaire en bloc $[nL, L, d_{DT}]$, avec d_{DT} la distance minimum de ce code ("Direct Truncation"). On notera ce code C_{DT}.

Construction par troncature directe

C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f et C_{DT}, le code linéaire en bloc $[nL, L, d_{DT}]$ issu de la troncature directe.

Propriété du code C_{DT}

Le **taux** de ce code C_{DT} est égal au taux du code convolutif $(1/n)$,
La distance minimum est plus petite : $d_{DT} < d_f$.

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux (rem : la distance libre $d_f = 5$).

Si le vecteur d'information est composé de 3 bits, le code linéaire obtenu est tel que :

- les mots de code obtenus sont de taille $3 \times n = 3 \times 2 = 6$ bits,
- la quantité d'information codée pour un tel code est de 3 bits.

La construction en troncature directe est donc un code linéaire en bloc [6, 3, d_{DT}]. La matrice génératrice est :

$$G = \left(\begin{array}{cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux ($d_f = 5$). La construction en troncature directe donne le code linéaire en bloc [6, 3, d_{DT}] de matrice génératrice :

$$G = \left(\begin{array}{cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

La **séquence de distribution de poids** de ce code est $A(x) = 1 + x^2 + 3x^3 + 2x^4 + x^5$. Le poids le plus faible (en mettant à part le poids du vecteur $\underline{0}$) pour ce code est 2 ; la distance minimum $d_{DT} = 2$.

Le taux pour ce code est de $3/6 = 1/2$ (*bit d'information/bits transmis*). Le taux pour le code convolutif initial est le même ($1/2 = 0.5$ *bit d'information/bits transmis*).

Distribution de poids de C_{DT}

Rappel : Matrice de transition d'état

Soit $\Omega_{ij}(x)$ la matrice $2^m \times 2^m$ défini par : $\Omega_{ij}(x) = \delta_{i,j} x^{h_{i,j}}$, avec h_{ij} le poids de Hamming du vecteur de sortie.

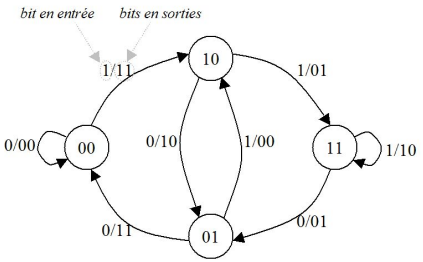
La **séquence de distribution de poids** du code C_{DT} est $A(x) = \sum_{j=0}^{2^m-1} \Omega_{0,j}^{l}(x)$, avec l le nombre de transitions, 0 l'état de départ et $j \in [0, 2^m - 1]$ l'un des états d'arrivée dans le treillis.

Plan

Construction "tail-biting" ("queue-entrante")

Soit C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f.

Soit un vecteur d'information $\overline{u}$ de taille L. Le codage convolutif de ce **vecteur** u nécessite de se **placer dans l'état** désigné par les **m derniers bits** de u (lecture droite vers gauche). Le codage convolutif donne des mots de codes font aboutir le codeur dans l'état de départ.
Le code ainsi obtenu est un code linéaire en bloc $[nL, L, d_{DT}]$, avec d_{TB} la distance minimum de ce code ("tail-biting"). On notera ce code C_{TB}.



Construction "tail-biting"

C un code convolutif m-mémoires $1/n$-taux, sous forme non systématique (FIR), de distance libre d_f, et C_{DT} le code linéaire en bloc $[nL, L, d_{TB}]$ issu du "tail-biting".

Propriété du code C_{TB}

Le **taux** de ce code C_{TB} est égal au taux du code convolutif $(1/n)$,
La distance minimum est $d_{TB} \leq d_f$.

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux (rem : la distance libre $d_f = 5$).

Si le vecteur d'information est composé de 5 bits, le code linéaire obtenu est tel que :

- les mots de code obtenus sont de taille $5 \times n = 5 \times 2 = 10$ bits,
- la quantité d'information codée pour un tel code est de 5 bits.

La construction en troncature directe est donc un code linéaire en bloc [10, 5, d_{TB}]. La matrice génératrice est :

$$G = \left(\begin{array}{cc|cc|cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

Illustration avec le codeur 2-mémoires 1/2-taux

Soit C un code convolutif 2-mémoires 1/2-taux ($d_f = 5$). La construction en "tail-biting" donne le code linéaire en bloc [10, 5, d_{TB}] de matrice génératrice :

$$G = \left(\begin{array}{cc|cc|cc|cc|cc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

La **séquence de distribution de poids** de ce code est $A(x) = 1 + 5x^3 + 5x^4 + 6x^5 + 10x^6 + 5x^7$. Le poids le plus faible (en mettant à part le poids du vecteur $\underline{0}$) pour ce code est 3 ; la distance minimum $d_{DT} = 3 < d_f = 5$.

Le taux pour ce code est de $3/6 = 1/2$ (*bit d'information/bits transmis*). Le taux pour le code convolutif intitial est le même ($1/2 = 0.5$ *bit d'information/bits transmis*).

Distribution de poids de C_{TB}

Rappel : Matrice de transition d'état

Soit $\Omega_{ij}(x)$ la matrice $2^m \times 2^m$ défini par : $\Omega_{ij}(x) = \delta_{i,j} x^{h_{i,j}}$, avec h_{ij} le poids de Hamming du vecteur de sortie.

La **séquence de distribution de poids** du code C_{TB} est $A(x) = \sum_{j=0}^{2^m-1} \Omega^l_{j,j}(x)$, avec l le nombre de transitions, $j \in [0, 2^m - 1]$ l'état d'arrivée et de départ dans le treillis.