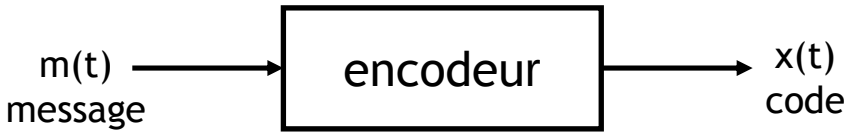
Code convolutif

M. Moussaoui

La convolution

Le problème du codage se schématise par :



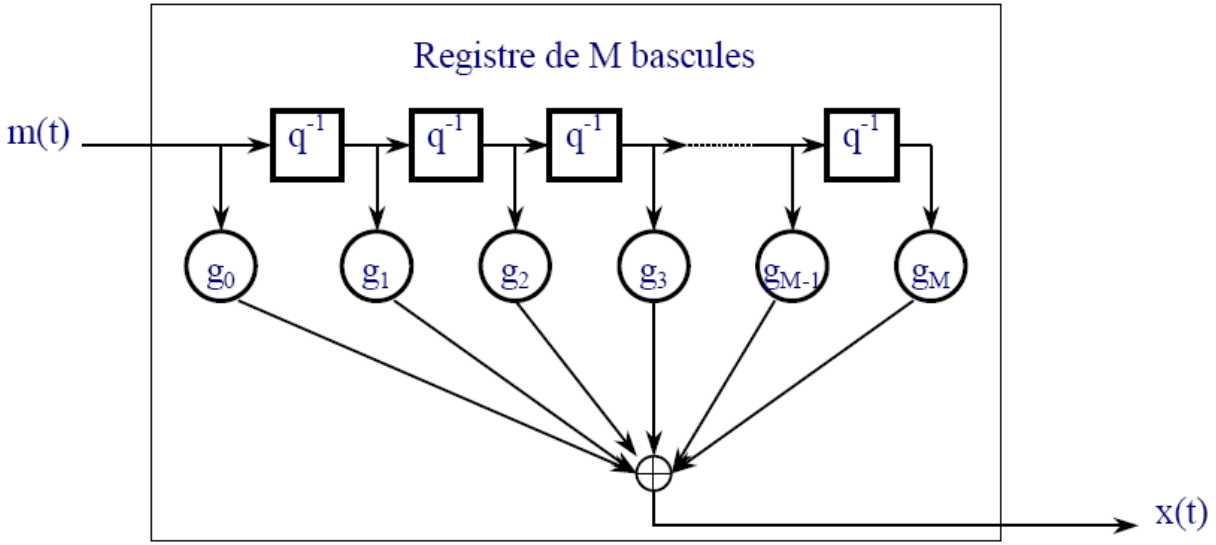
m(t) et x(t) sont des signaux discrets dont les échantillons sont binaires.

Si l'encodeur est un système linéaire, x(t) peut être généré par un produit de convolution

$$x_j = \sum_{i=0}^{M} g_i m_{j-i}$$

Schéma de principe

En binaire l'opération de convolution se réalise très simplement par :



En se rappelant que : gi = 0 ⇒ connexion absente

gi = 1 ⇒ connexion présente

Propriétés

Longueur de contrainte : paramètre K.

La longueur de contrainte est le nombre d'instants d'échantillonnage pendant lequel un bit du message participe à l'élaboration du mot de code (la sortie).
En regardant le schéma précédent il vient :

K = M + 1

Début du codage, initialisation du registre :

Par convention, nous prenons comme instant initial la présence du bit m_0 à l'entrée de l'encodeur

Par convention, les registres sont initialisés à "0"

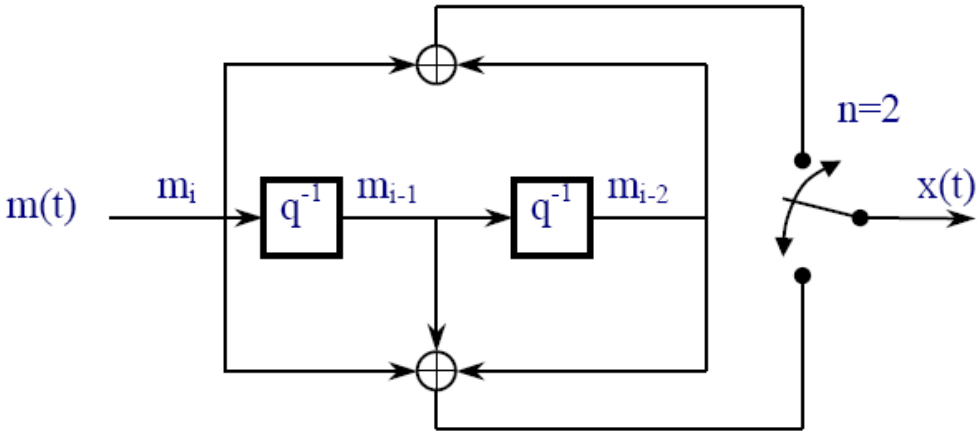
Fin du codage

Le codage est en cours tant que le dernier bit du message est actif sur l'élaboration de la sortie c'est-à-dire présent en sortie de la dernière bascule du registre.

Par convention des bits de queue sont des « 0 » qui pratiquement servent à réinitialiser les bascules du registre.

Généralisation de l'encodage de convolution

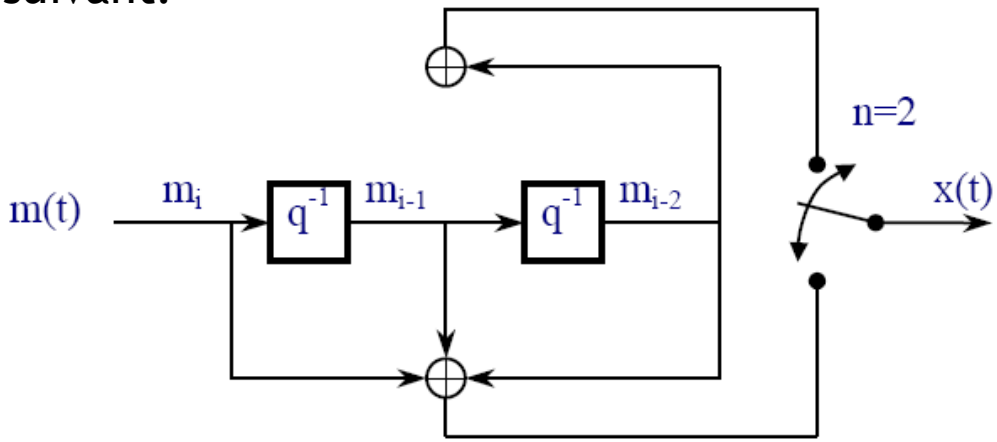
Pour améliorer le codage, il est possible de réaliser à partir du même message n séquences de convolution différentes qui seront entrelacées.



longueur de contrainte de K = 3 (2 bascules, M = 2) avec un entrelacement en sortie n = 2.

FONCTIONNEMENT TEMPOREL

Exemple pour lequel M= 2 génère deux séquences qui seront entrelacées selon le principe suivant:



$$x_j^{(1)} = \sum_{i=0}^{M} g_i^{(1)} m_{j-i} \qquad \text{avec} \qquad g_i^{(1)} = \left[g_0^{(1)} \; g_1^{(1)} \; g_2^{(1)} \right] = \left[0 \; 0 \; 1 \right]$$

$$x_j^{(2)} = \sum_{i=0}^{M} g_i^{(2)} m_{j-i} \qquad \text{avec} \qquad g_i^{(2)} = \left[g_0^{(2)} \; g_1^{(2)} \; g_2^{(2)} \right] = \left[1 \; 1 \; 1 \right]$$

La convolution se calcule sur K bits (ici K = 3) et tous les cas possibles peuvent se représenter dans un tableau : la table de convolution

FONCTIONNEMENT TEMPOREL

Combinaisons d'entrée	$[g_0^{(1)} g_1^{(1)} g_2^{(1)}] = [0\ 0\ 1]$	$[g_0^{(2)} g_1^{(2)} g_2^{(2)}] = [1\ 1\ 1]$
$[m_j \quad m_{j-1} \quad m_{j-2}]$	$x_j^{(1)}$	$x_j^{(2)}$
0 0 0	0	0
0 0 1	1	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	0	0
1 1 1	1	1

FONCTIONNEMENT TEMPOREL

Avec le message $m = [m_0\ m_1\ m_2\ \ldots\ldots m_{L-1}] = [\ 1\ 0\ 0\ 1\ 1\ ]$ (ici L= 5) :

j	remarque	$[\ m_j \quad m_{j-1} \quad m_{j-2}\ ]$	$x_j^{(1)}\ x_j^{(2)}$
0	Deux "0" d'initialisation	$[\ m_0\ 0\ 0\ ] = 1\ 0\ 0$	0 1
1	Un "0" d'initialisation	$[\ m_1\ m_0\ 0\ ] = 0\ 1\ 0$	0 1
2		$[\ m_2\ m_1\ m_0\ ] = 0\ 0\ 1$	1 1
3		$[\ m_3\ m_2\ m_1\ ] = 1\ 0\ 0$	0 1
4		$[\ m_4\ m_3\ m_2\ ] = 1\ 1\ 0$	0 0
5	Un "0" de queue	$[\ 0\ m_4\ m_3\ ] = 0\ 1\ 1$	1 0
6	Deux "0" de queue	$[\ 0\ 0\ m_4\ ] = 0\ 0\ 1$	1 1

Ceci fournit donc le mot de code : x = [0 1 0 1 1 1 0 1 0 0 1 0 1 1]

REPRESENTATIONS GRAPHIQUES

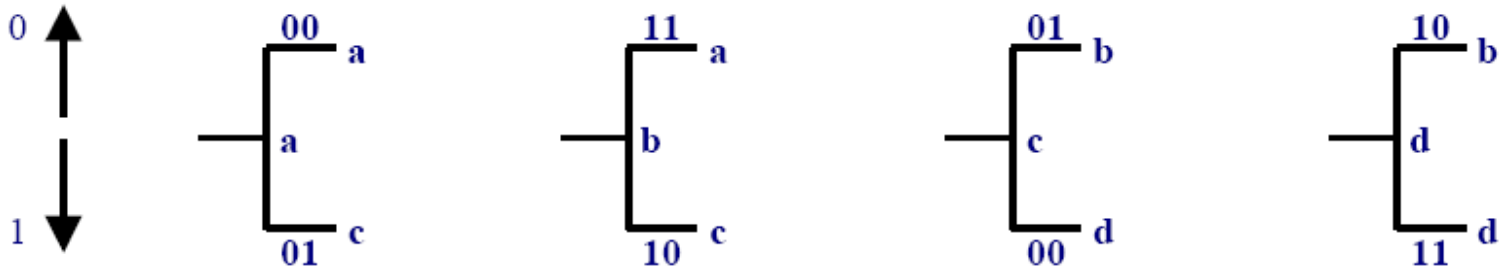
Etat présent – état suivant :

Etat suivant	Combinaisons d'entrée	Etat présent	$[g_0^{(1)} g_1^{(1)} g_2^{(1)}] =$ [1 0 1]	$[g_0^{(2)} g_1^{(2)} g_2^{(2)}] =$ [1 1 1]
	$[m_j \; m_{j-1} \; m_{j-2}]$		$x_j^{(1)}$	$x_j^{(2)}$
a	0 0 0	a	0	0
a	0 0 1	b	1	1
b	0 1 0	c	0	1
b	0 1 1	d	1	0
c	1 0 0	a	0	1
c	1 0 1	b	1	0
d	1 1 0	c	0	0
d	1 1 1	d	1	1

REPRESENTATIONS GRAPHIQUES

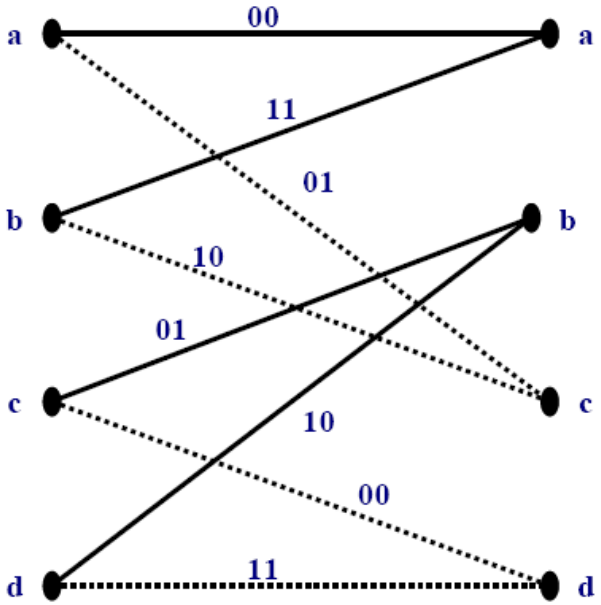
Nous pouvons en déduire les éléments de base du graphe avec pour convention :

m_j = 0, déplacement vers le haut
m_j = 1, déplacement vers le bas



Cellule élémentaire

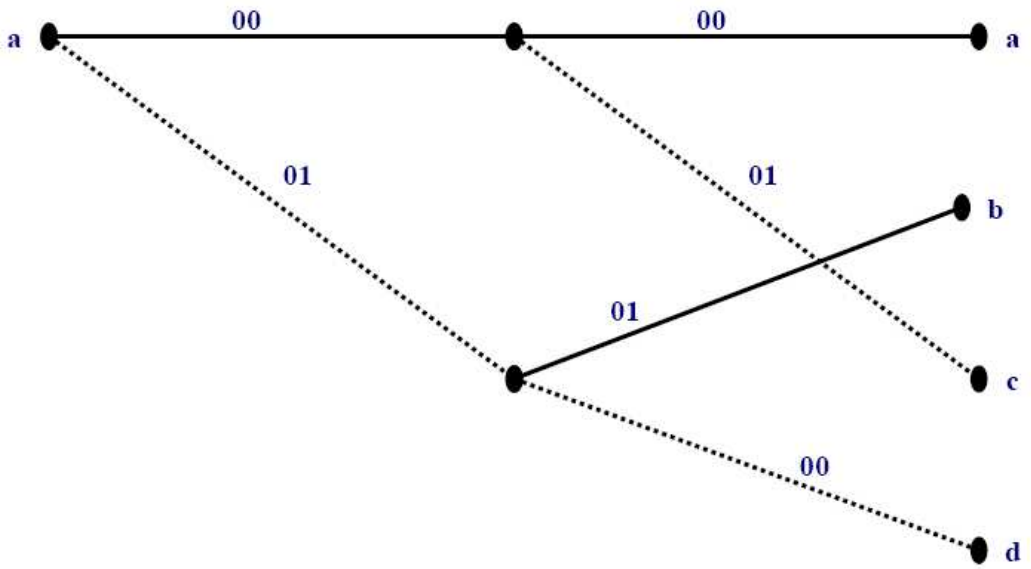
nous pouvons dessiner une cellule élémentaire du treillis reliant l'instant t à l'instant t+1. Cette cellule comporte 2^(K-1) entrées et sorties.



Cellule élémentaire

Phase initiale :

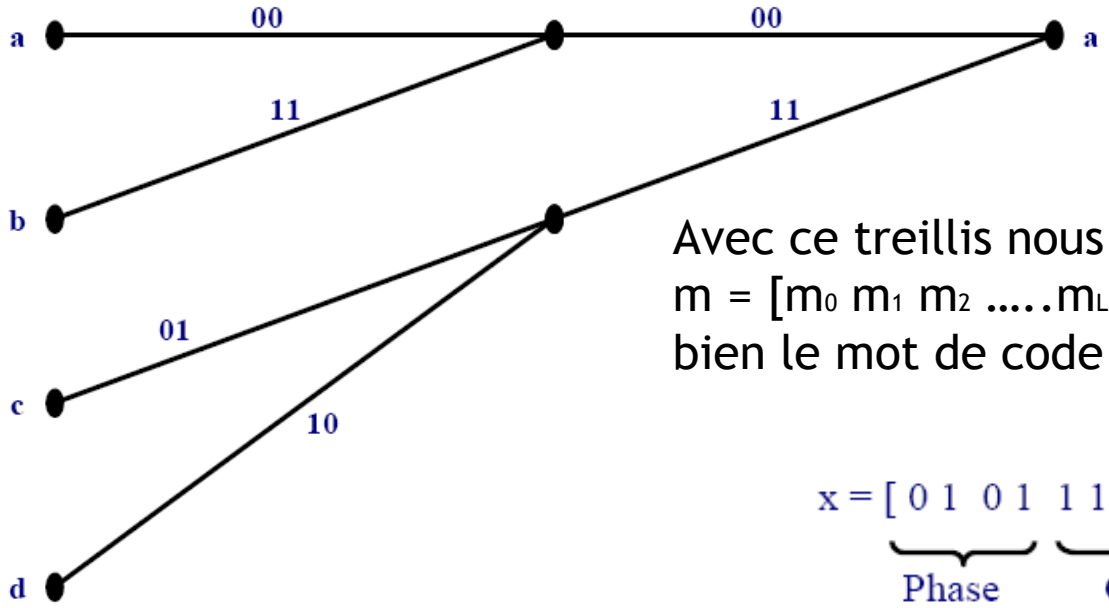
Pendant M = K-1 instants, une partie de l'état des bascules est constitué des "0" d'initialisation. Pendant tous ces instants, toutes les combinaisons ne sont pas possibles et tous les états ne sont pas accessibles.
Le treillis comporte donc à l'origine M cellules correspondant à cette phase initiale.



Cellule élémentaire

Phase finale :

A la fin du codage nous sommes amenés à introduire des "0" de queue. De ce fait, seules les transitions en trait plein deviennent possibles et le treillis se résume à M cellules ne comportant que ces transitions.



Avec ce treillis nous vérifions que le message $m = [m_0\ m_1\ m_2\ \ldots..m_{L-1}] = [\ 1\ 0\ 0\ 1\ 1\ ]$ génère bien le mot de code :

$$x = [\underbrace{0\ 1\ \ 0\ 1}_{\text{Phase initiale}}\ \underbrace{1\ 1\ \ 0\ 1\ \ 0\ 0}_{\text{Cellule élémentaire}}\ \underbrace{1\ 0\ \ 1\ 1}_{\text{Phase finale}}\ ].$$

DECODAGE DE CONVOLUTION

Maximum de vraisemblance :

Le schéma de la transmission est :

Message **m**→ code **x** → réception **y = x + e**

Rôle du décodeur :

Le décodeur doit faire la meilleure estimation de m qui correspond à la meilleure estimation de x (convolution = relation linéaire).

$$y \rightarrow \hat{x} \rightarrow \hat{m}$$

$\hat{x}$ correspond à la valeur qui rend maximale la probabilité conditionnelle p(y/x) soit

$$\hat{x} = \left[p(y / x) \right]_{\max}$$

Maximum de vraisemblance

Tous les messages sont équiprobables, et la fonction logarithme est une fonction monotone. Il est équivalent d'exprimer ce maximum de vraisemblance en optimisant le Ln d'où :

$$\hat{x} \rightarrow \max\left[Ln\left[p\left(y/x\right)\right]\right]$$

Cas du canal binaire symétrique :

Si le canal binaire symétrique, tous les bits du message sont statistiquement indépendants et :

$$p\left(y/x\right)=\prod_{i=1}^{N} p\left(y_i/x_i\right) \Rightarrow Ln\left[p\left(y/x\right)\right]=\sum_{i=1}^{N} Ln\left[p\left(y_i/x_i\right)\right]$$

Pour le canal binaire symétrique de probabilité d'erreur de transmission p :

$$p\left(y_i/x_i\right)=p \;\; si \;\; y_i \neq x_i$$

$$p\left(y_i/x_i\right)=1-p \;\; si \;\; y_i = x_i$$

$$Ln\left[p\left(y/x\right)\right]=dLn\left(p\right)+\left(N-d\right)Ln\left(1-p\right)=dLn\left[\frac{p}{1-p}\right]+NLn\left(1-p\right)$$

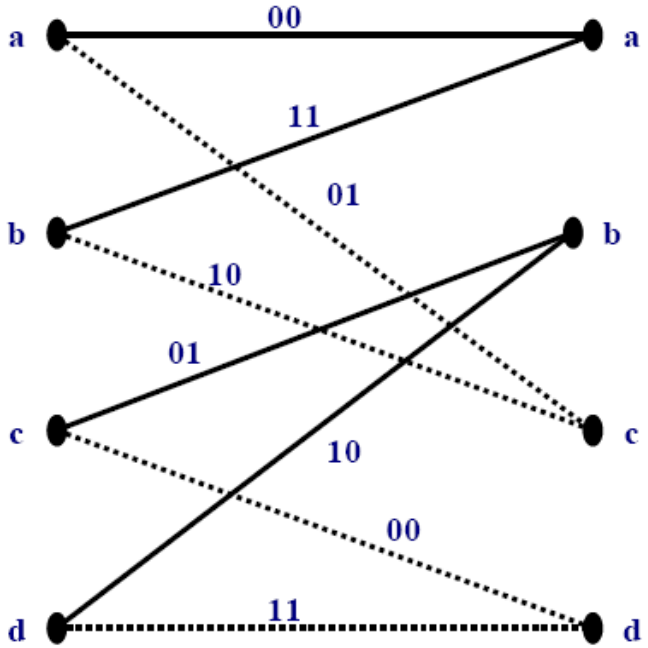
Pour un canal binaire symétrique, nous devons choisir xˆ de manière à minimiser la distance de Hamming entre x et y.

Maximum de vraisemblance

Pour cela, x est bien sûr inconnu mais nous avons à notre connaissance le treillis du code et, à la réception de y, nous chercherons à minimiser la distance de Hamming entre le code reçu et tous les codes possibles. Le plus "proche de y constituera l'estimation la plus vraisemblable.

Algorithme de Viterbi

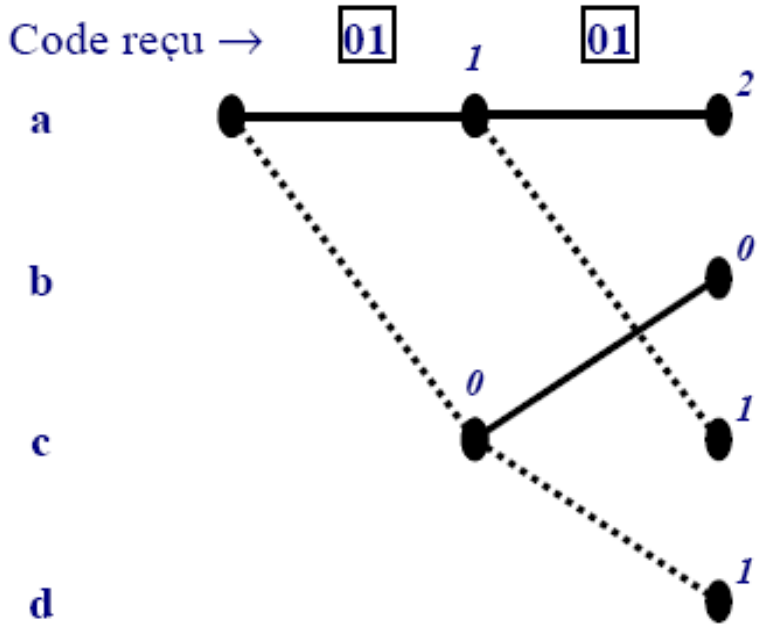
Nous l'expliquerons sur un exemple en utilisant le code précédent dont le treillis est rappelé ci-contre.



Avec ce treillis, nous avons obtenu le codage :
$m = [m_0\ m_1\ m_2\ .....m_{L-1}] = [\ 1\ 0\ 0\ 1\ 1\ ]$
$\rightarrow x = [\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ ]$
Supposons une erreur de transmission telle que :
$y = [\ 0\ 1\ 0\ 1\ \mathbf{0}\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ ]$

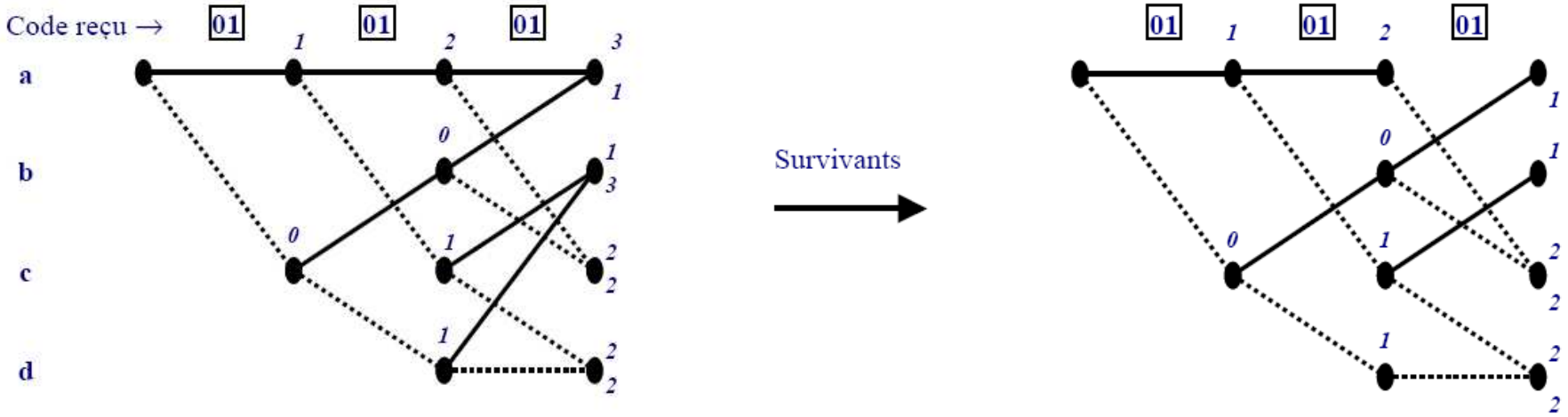
Phase initiale

Elle dure M coups et les états ne peuvent être atteints que d'une seule façon. Durant cette phase il n'y a donc pour un état suivant qu'une seule distance et aucun "trajet" ne sera rejeté. L'état initial est nécessairement l'état interne "0" soit **a** dans notre exemple.



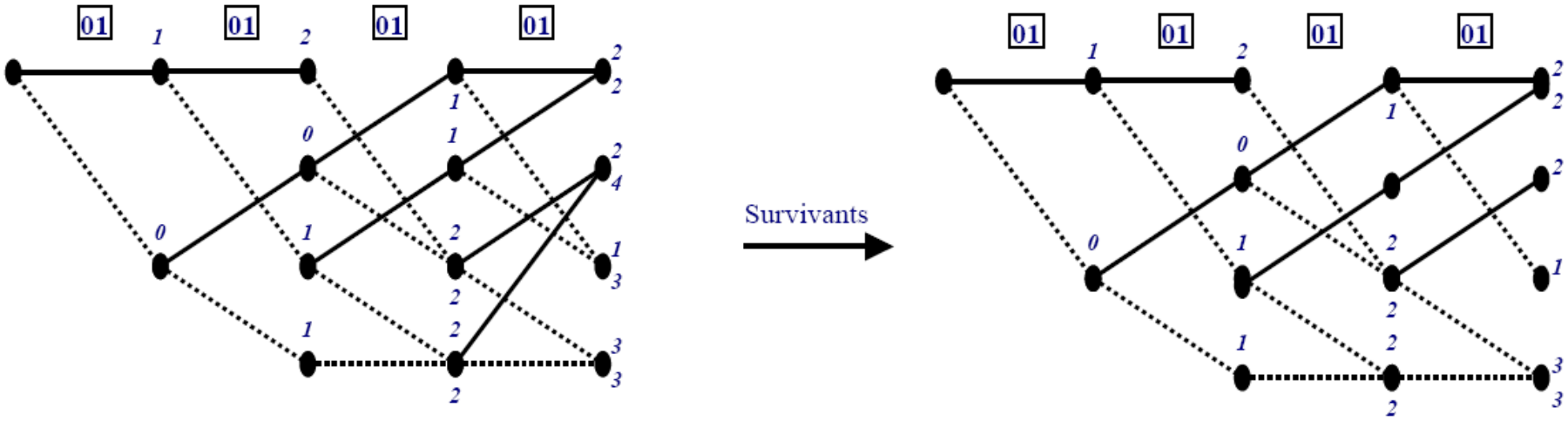
Phase centrale

Tous les états sont atteignables de deux manières et à chaque étape nous décidons de ne conserver que les trajets correspondants pour chaque état possible à une distance minimale : les survivants.



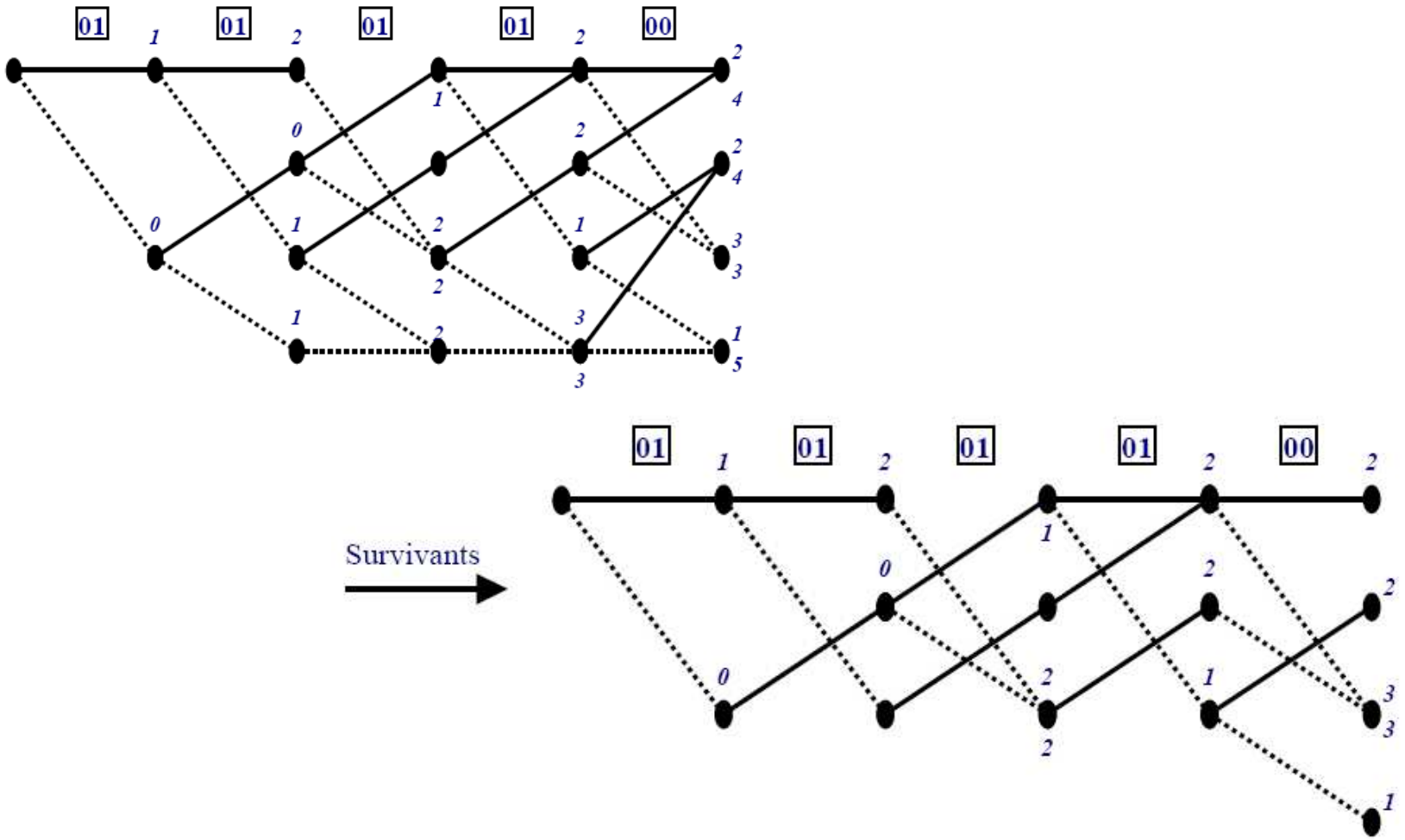
Phase centrale

étape suivante :



Phase centrale

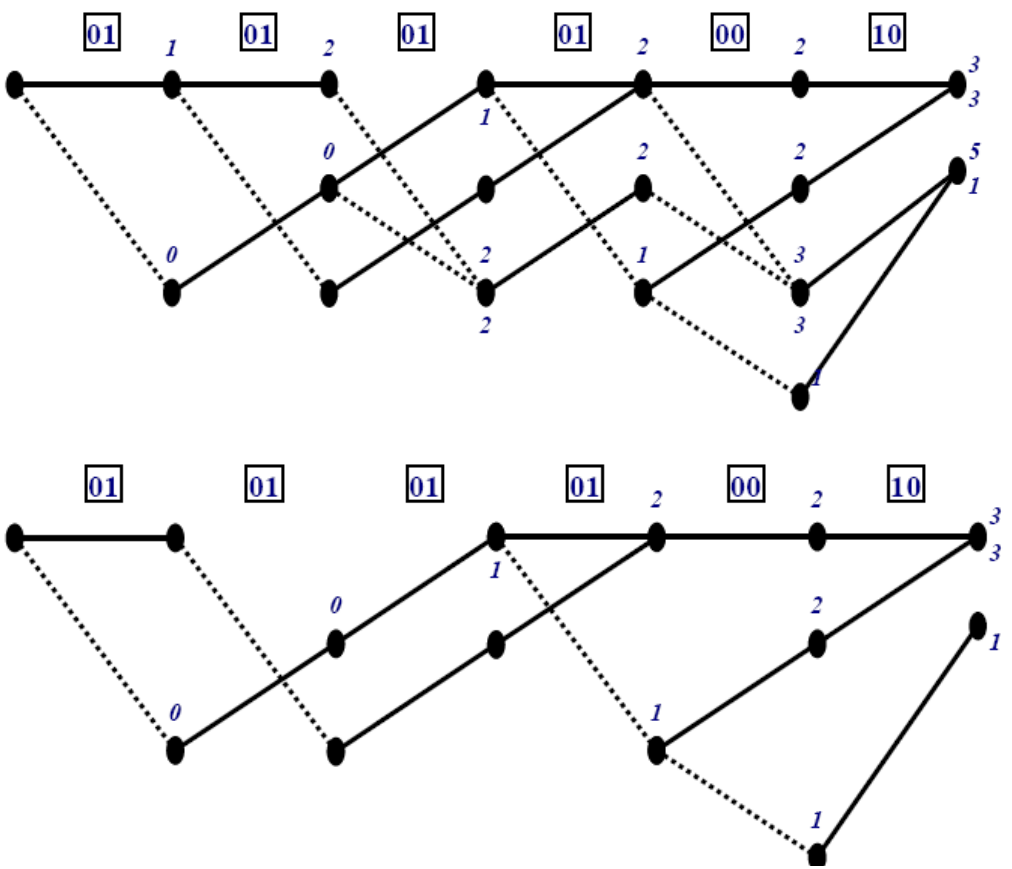
étape suivante :



Phase finale

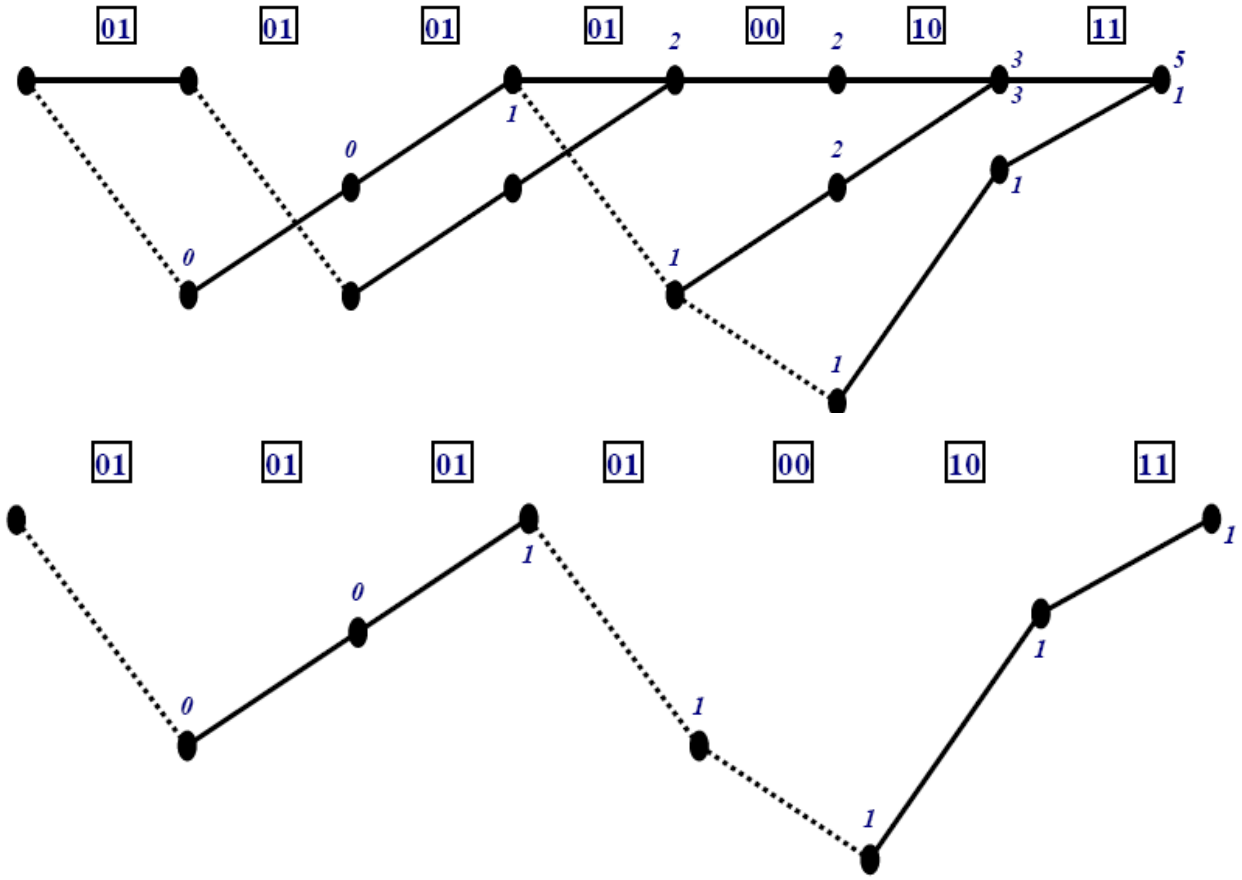
Nous savons que les M derniers blocs correspondent à la phase finale pendant laquelle seuls les "0« (trait plein) sont introduits.

Premier zéro de queue :



Phase finale

Deuxième zéro de queue :



Le dernier trajet survivant est le plus vraisemblable et correspond au message [1 0 0 1 1] . L'erreur de transmission a ainsi été corrigée.