TS226
-
Codes convolutifs et codes concaténés associés

Romain Tajan

19 octobre 2018

Plan

QCM

Comment avez-vous trouvé le cours précédent ?

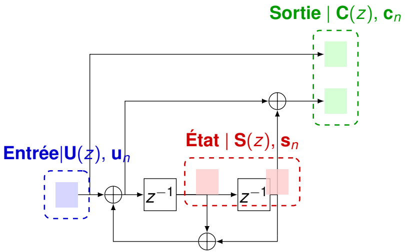
- A Très difficile
- B Difficile
- C Moyen
- D Simple
- E Très simple

#QDLE#S#ABCDE#30#

Plan

1. Previously on TS226 . . .
   - ▷ Rappels sur l'encodeur
   - ▷ Rappels sur le diagramme d'état
   - ▷ Rappels sur le treillis
   - ▷ Rappels sur les décodeurs
2. Décodage du Maximum de Vraisemblance des codes convolutifs
3. Turbo-Codes

Rappels / définitions



- **Entrée** : $\mathbf{U}(z) = [U^{(0)}(z),\ U^{(1)}(z)\ \dots\ U^{(n_b-1)}(z)]$

  $\rightarrow$ dans ce cours $n_b = 1 \Rightarrow \mathbf{U}(z) \rightarrow U(z)$

- **État** : $\mathbf{S}(z) = [S^{(0)}(z),\ S^{(1)}(z)\ \dots\ S^{(m-1)}(z)]$

  $\rightarrow$ m est appelé "mémoire du code"

  $\rightarrow$ $\nu = m + 1$ est appelé "longueur de contrainte"

  $\rightarrow$ dans l'exemple $m = 2$

  $\rightarrow$ 2^m : nombre d'états

- **Sortie** : $\mathbf{C}(z) = [C^{(0)}(z),\ C^{(1)}(z)\ \dots\ C^{(m-1)}(z)]$

  $\rightarrow$ n_s nombre de sorties

  $\rightarrow$ dans l'exemple $n_s = 2$

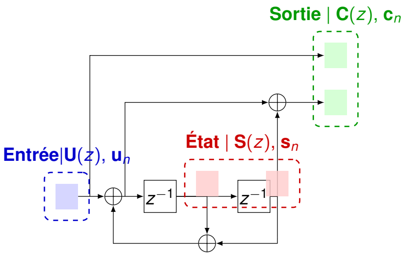
  $\rightarrow$ De façon générale : $\mathbf{C}(z) = U(z)\mathbf{G}(z)$

  $\rightarrow$ où $\mathbf{G}(z) = [\frac{A^{(1)}(z)}{B^{(1)}(z)}\ \dots\ \frac{A^{(n_s)}(z)}{B^{(n_s)}(z)}]$

- **Rendement du code** : $R = \frac{\#\text{bits d'info. en entrée}}{\#\text{bits codés en sortie}}$

  $\rightarrow$ Ici : $R = \frac{n_b}{n_s} = \frac{1}{2}$

Rappels / définitions



- **Code linéaire**

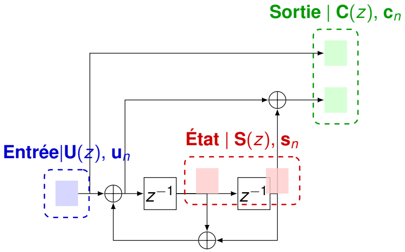
$\rightarrow$ Si $\mathbf{C}_1(z)$ et $\mathbf{C}_2(z)$ sont deux mots de codes, alors $\mathbf{C}_3(z) = \mathbf{C}_2(z) + \mathbf{C}_1(z)$ est aussi un mot de code.

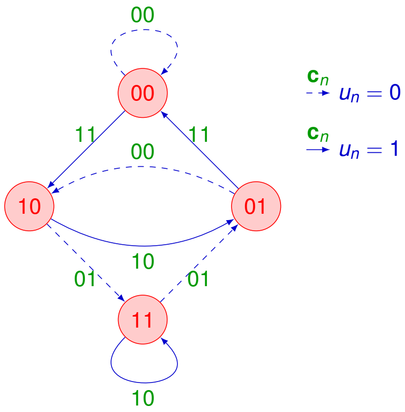
- **Encodeur récursif / non récursif**

- **Encodeur systématique / non systématique**

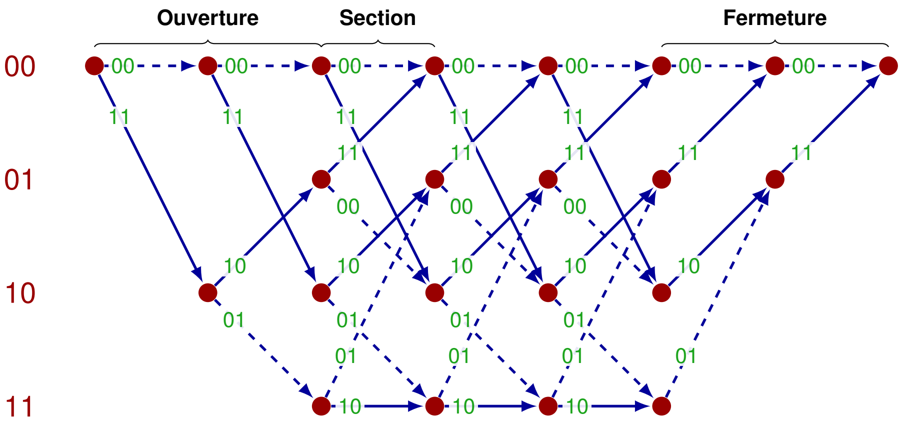
- **Notation octale**

Rappels / définitions

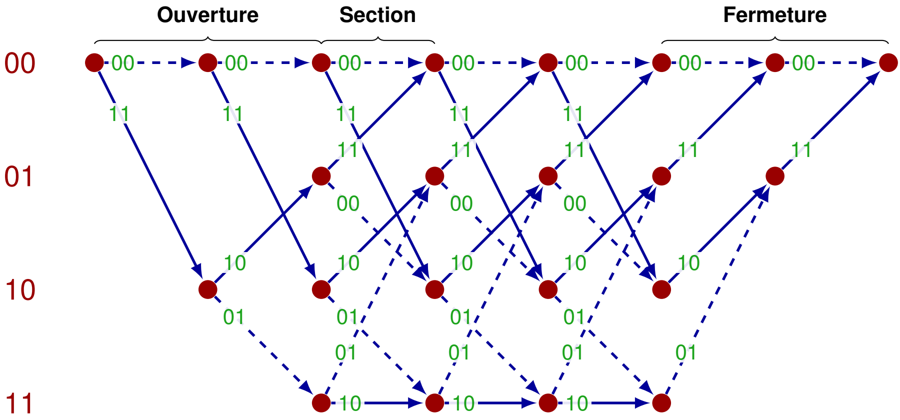




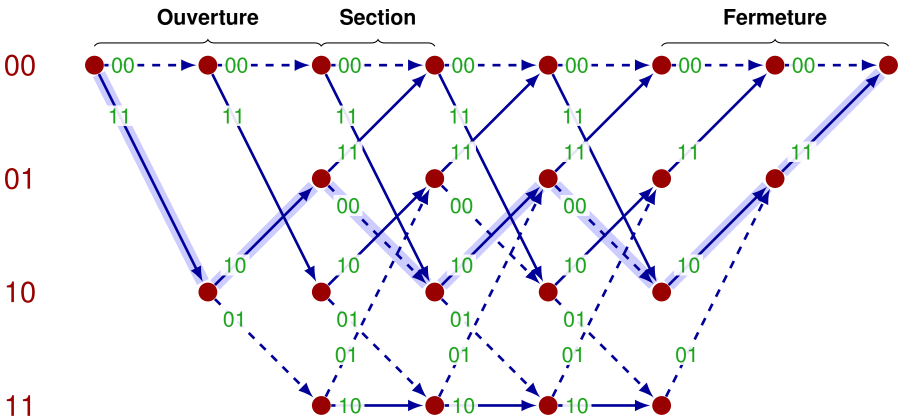
- Message ⇔ chemin dans le graphe
- Mot de code ⇔ étiquettes le long du chemin dans le graphe
- Nécessité de définir (au moins) un état initial



- Treillis $\Leftrightarrow$ machine à états faisant apparaître le temps explicitement.
- Fermer $\Rightarrow N = n_s(m+K) \Rightarrow R = \dfrac{K}{n_s(m+K)} \leq \dfrac{1}{n_s}$
- Fermer $\Rightarrow$ améliore la probabilité d'erreur

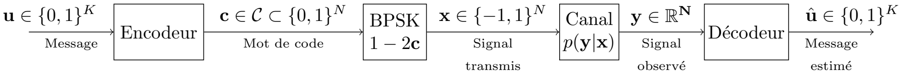


- On souhaite envoyer le message : $\mathbf{u} = [1,\ 1,\ 0,\ 1,\ 0]$



- On souhaite envoyer le message : $\mathbf{u} = [1,\ 1,\ 0,\ 1,\ 0]$
- On calcule le mot de code : $\mathbf{c} = [1\ 1,\ 1\ 0,\ 0\ 0,\ 1\ 0,\ 0\ 0,\ \underline{1\ 0},\ \underline{1\ 1}]$
- On envoie le signal : $\mathbf{x} = [-1\ -1,\ -1\ 1,\ 1\ 1,\ -1\ 1,\ 1\ 1,\ \underline{-1\ 1},\ \underline{-1\ -1}]$

Décodage du Maximum a Posteriori



Définition

- Le couple **encodeur/BPSK** sera vu comme une fonction (bijective) de $\mathbf{u}$:

$$\mathbf{x} = \Phi(\mathbf{u})$$

- Un **décodeur** est une fonction de $\mathbf{y}$ définie dans $\{0,1\}^K$:

$$\hat{\mathbf{u}} = \Psi(\mathbf{y})$$

- Le **décodeur** du **Maximum A Posteriori** (**MAP**) est la fonction de $\mathbf{y}$ définie par :

$$\Psi_{MAP}(\mathbf{y}) = \underset{\mathbf{u} \in \{0,1\}^K}{\operatorname{argmax}} \ \mathbb{P}(\mathbf{U} = \mathbf{u}|\mathbf{y})$$

- **Probabilité d'erreur trame** : $P_e = \mathbb{P}(\mathbf{U} \neq \hat{\mathbf{U}})$

Décodage du Maximum a Posteriori

$$\underset{\text{Message}}{\mathbf{u} \in \{0,1\}^K} \longrightarrow \boxed{\text{Encodeur}} \longrightarrow \underset{\text{Mot de code}}{\mathbf{c} \in \mathcal{C} \subset \{0,1\}^N} \longrightarrow \boxed{\begin{array}{c}\text{BPSK}\\ 1-2\mathbf{c}\end{array}} \longrightarrow \underset{\text{Signal transmis}}{\mathbf{x} \in \{-1,1\}^N} \longrightarrow \boxed{\begin{array}{c}\text{Canal}\\ p(\mathbf{y}|\mathbf{x})\end{array}} \longrightarrow \underset{\text{Signal observé}}{\mathbf{y} \in \mathbb{R}^{\mathbf{N}}} \longrightarrow \boxed{\text{Décodeur}} \longrightarrow \underset{\text{Message estimé}}{\hat{\mathbf{u}} \in \{0,1\}^K}$$

Définition

- Le couple **encodeur/BPSK** sera vu comme une fonction (bijective) de **u** :

$$\mathbf{x} = \Phi(\mathbf{u})$$

- Un **décodeur** est une fonction de **y** définie dans $\{0,1\}^K$:

$$\hat{\mathbf{u}} = \Psi(\mathbf{y})$$

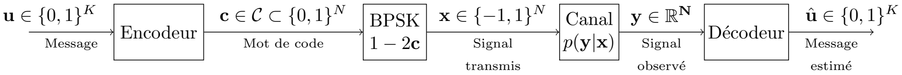
- Le **décodeur** du **Maximum A Posteriori** (**MAP**) est la fonction de **y** définie par :

$$\Psi_{MAP}(\mathbf{y}) = \underset{\mathbf{u} \in \{0,1\}^K}{\operatorname{argmax}} \mathbb{P}(\mathbf{U} = \mathbf{u}|\mathbf{y})$$

- **Probabilité d'erreur trame** : $P_e = \mathbb{P}(\mathbf{U} \neq \hat{\mathbf{U}})$

Le décodeur MAP minimise P_e !

Décodage du Maximum A Posteriori - Bit (MAP-Bit)



Définition

- Le **décodeur** du **Maximum A Posteriori - Bit** (**MAP-Bit**) est la fonction de **y** définie par :

$$\hat{u}_i = (\Psi_{MAP-Bit}(\mathbf{y}))_i = \underset{u_i \in \{0,1\}}{\operatorname{argmax}} \, \mathbb{P}(U_i = u_i|\mathbf{y})$$

- **Probabilité d'erreur binaire** : $P_b = \dfrac{1}{K} \displaystyle\sum_{k=0}^{K-1} \mathbb{P}(U_k \neq \hat{U}_k)$

Le décodeur MAP-Bit minimise la probabilité d'erreur binaire.

Décodage du Maximum de Vraisemblance (ML)

$$\xrightarrow[\text{Message}]{\mathbf{u}\in\{0,1\}^K} \boxed{\text{Encodeur}} \xrightarrow[\text{Mot de code}]{\mathbf{c}\in\mathcal{C}\subset\{0,1\}^N} \boxed{\begin{array}{c}\text{BPSK}\\ 1-2\mathbf{c}\end{array}} \xrightarrow[\text{Signal transmis}]{\mathbf{x}\in\{-1,1\}^N} \boxed{\begin{array}{c}\text{Canal}\\ p(\mathbf{y}|\mathbf{x})\end{array}} \xrightarrow[\text{Signal observé}]{\mathbf{y}\in\mathbb{R}^{\mathbf{N}}} \boxed{\text{Décodeur}} \xrightarrow[\text{Message estimé}]{\hat{\mathbf{u}}\in\{0,1\}^K}$$

Définition

- Le **décodeur** du **Maximum de vraisemblance** (**ML**) est la fonction de **y** définie par :

$$\Psi_{ML}(\mathbf{y}) = \underset{\mathbf{u}\in\{0,1\}^K}{\operatorname{argmax}}\ p(\mathbf{y}|\mathbf{u}) = \underset{\mathbf{x}=\Phi(\mathbf{u})|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmax}}\ p(\mathbf{y}|\mathbf{x})$$

- **Le décodeur ML est équivalent au décodeur MAP si les messages sont équiprobables.**
- **Sur le canal AWGN sans mémoire, le décodeur ML est équivalent à la fonction suivante :**

$$\Psi_{ML}(\mathbf{y}) = \underset{\mathbf{x}=\Phi(\mathbf{u})|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmin}}\ ||\mathbf{y}-\mathbf{x}||_2^2 = \underset{\mathbf{x}=\Phi(\mathbf{u})|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmax}}\ \sum_{\ell=0}^{L-1} \mathbf{x}_\ell \mathbf{y}_\ell^T = \underset{\mathbf{c}\in\mathcal{C}|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmin}}\ \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$$

Plan

- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, où $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- Le message reçu : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}]$
- **Décodeur ML** : $\hat{\mathbf{u}} = \underset{\mathbf{c}\in\mathcal{C}|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$

- **Solution 1** : explorer tous les messages (eq. tous les mots de codes)

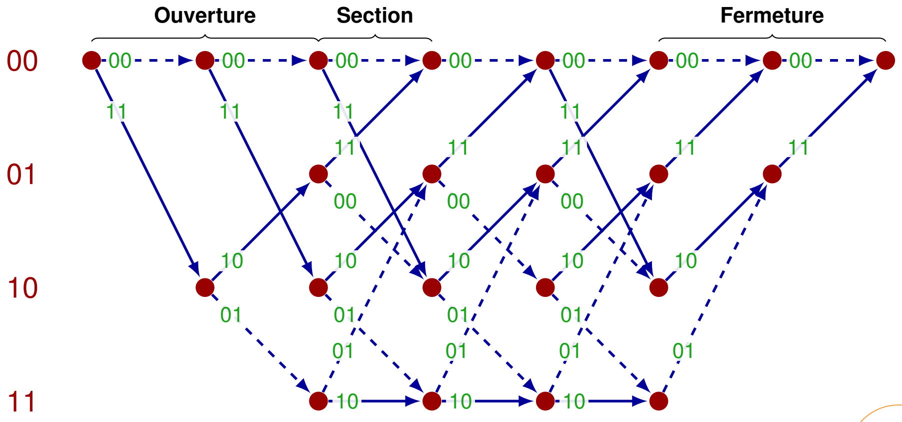
- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, où $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- Le message reçu : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}]$
- **Décodeur ML** : $\hat{\mathbf{u}} = \underset{\mathbf{c}\in\mathcal{C}|\mathbf{u}\in\{0,1\}^K}{\text{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$

- **Solution 1** : explorer tous les messages (eq. tous les mots de codes) $\rightarrow$ **il y en a** 2^K...

- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, où $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- Le message reçu : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}]$
- **Décodeur ML** : $\hat{\mathbf{u}} = \underset{\mathbf{c}\in\mathcal{C}|\mathbf{u}\in\{0,1\}^K}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$

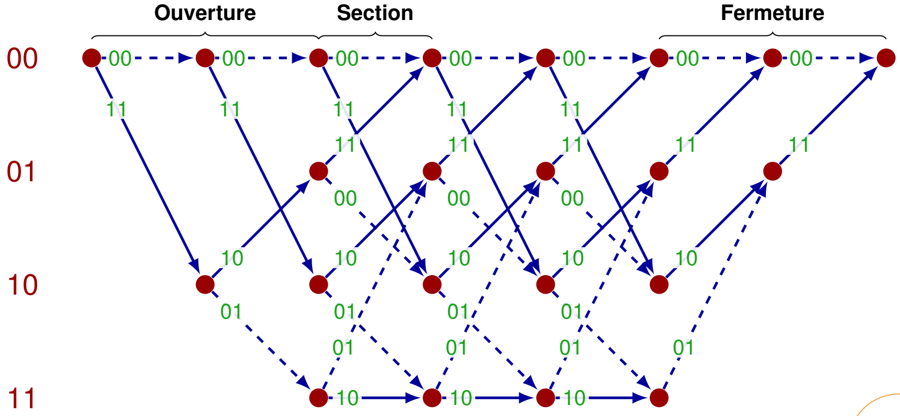
- **Solution 1** : explorer tous les messages (eq. tous les mots de codes) $\rightarrow$ **il y en a** 2^K...
- **Solution 2** : utiliser le treillis + la forme de la fonction de coût $\rightarrow$ **algorithme de Viterbi**

- Soit $\mathcal{S}_n$ l'ensemble des états possible du treillis à l'étage n
- Soit $\mathcal{C}(s_0 \to s_n)$ l'ensemble des chemins partant de s_0 arrivant à s_n dans le treillis
- Introduisons la fonction suivante $J_n(s_n) = \min_{\mathbf{c} \in \mathcal{C}(s_0 \to s_n)} \sum_{\ell=0}^{n-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$
- Le décodage ML est équivalent à trouver l'antécédent de $J_L(s_L)$) où $s_0 = s_L = 00$



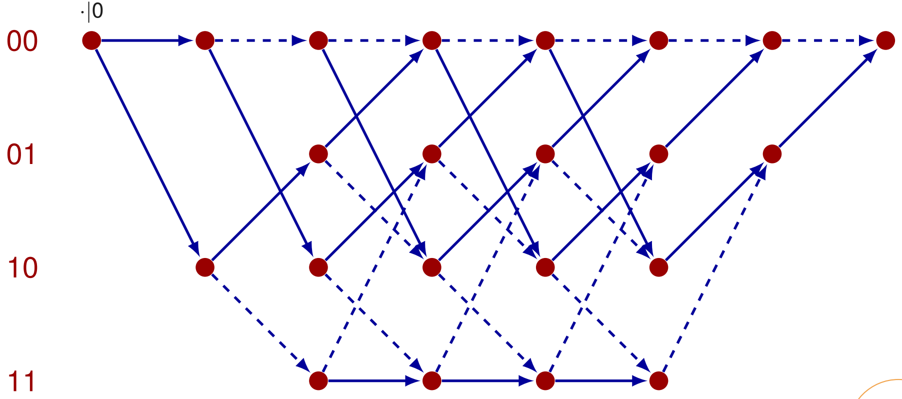
• **Remarque** : quelque soit $n \geq 0$ il existe au plus **deux transitions menant à** s_n, l'une correspond à $u_{n-1} = 0$ et l'autre $u_{n-1} = 1$. On notera $s_{n-1}^{(0)}$ et $s_{n-1}^{(1)}$ les états de départ de ces transitions et $\mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)$ la **sortie de l'encodeur** correspondante.

• **Montrons que** : $J_n(s_n) = \min_{u \in \{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T \right]$



Algorithme de Viterbi

1. Pour chaque s_n, calculer :
$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} | \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T \right]$$
2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

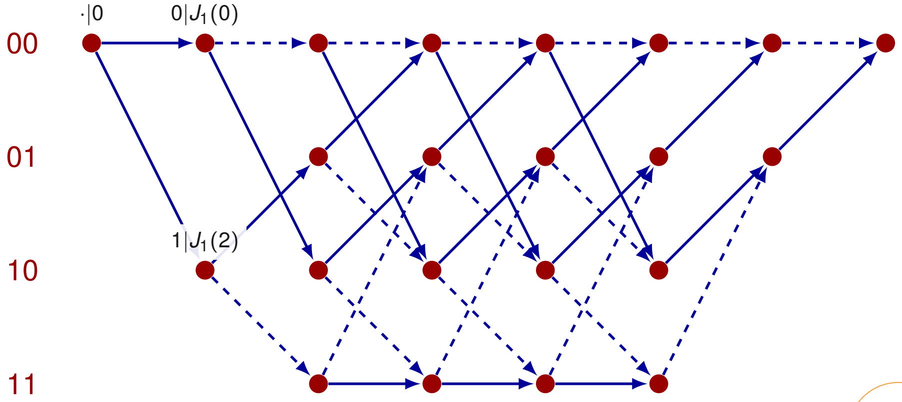


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\mathrm{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

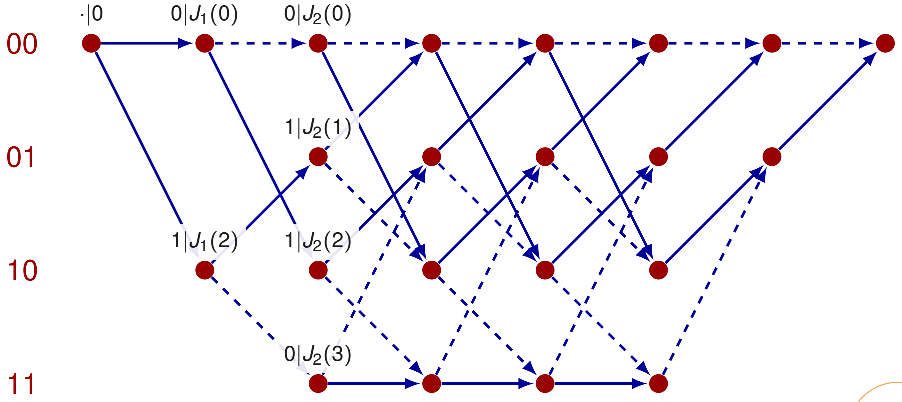


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

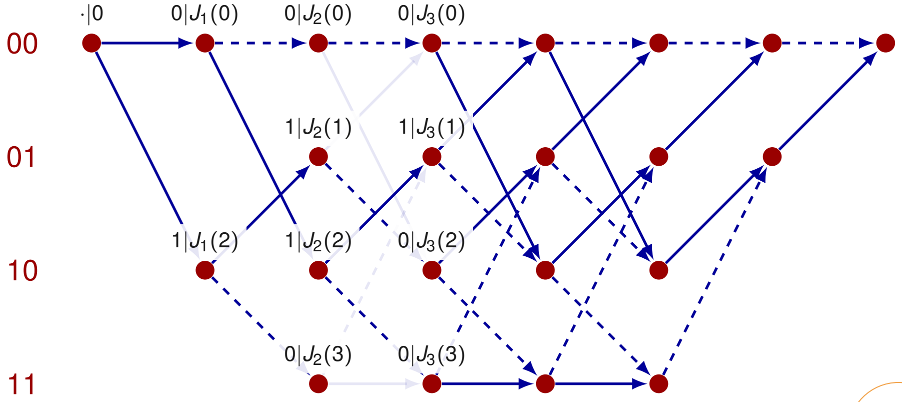


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

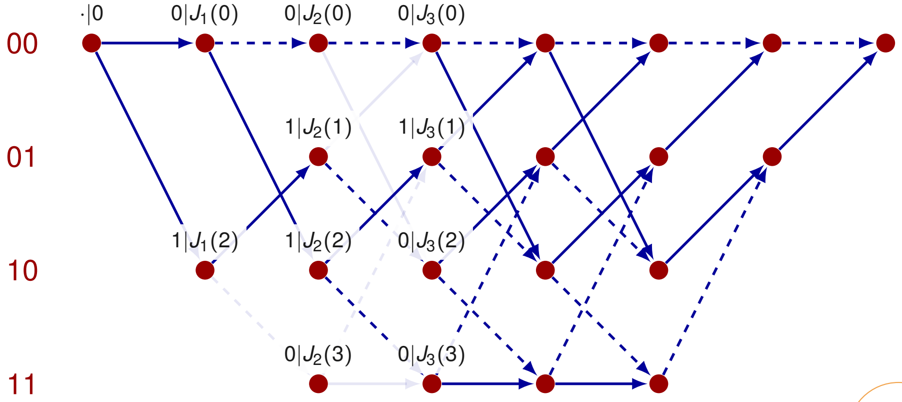


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \Big| \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

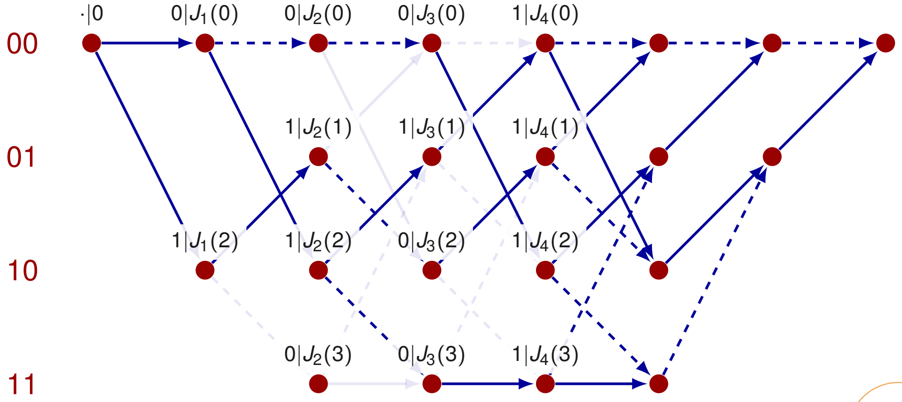


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T \right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

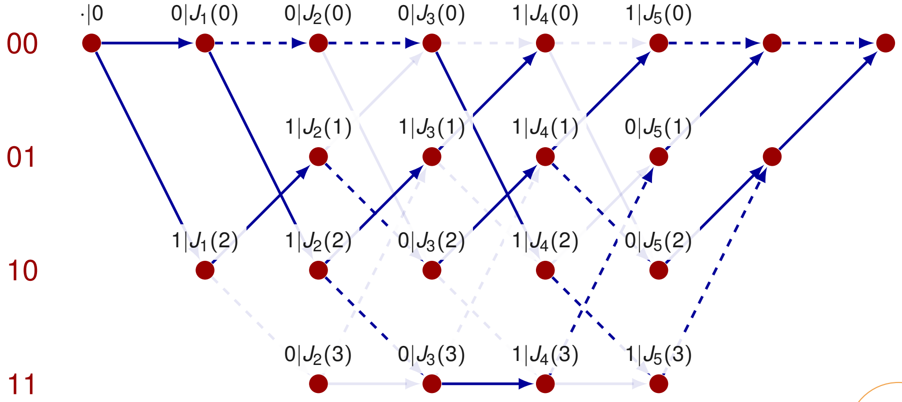


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

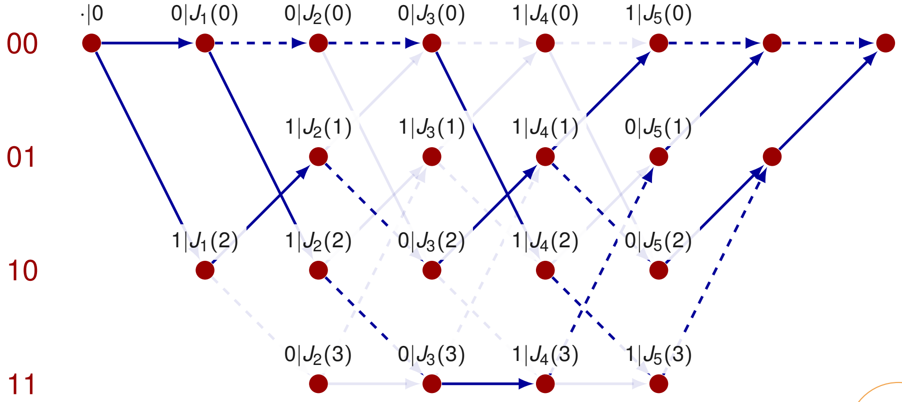
$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T \right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$



Algorithme de Viterbi

1. Pour chaque s_n, calculer :
$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$
2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

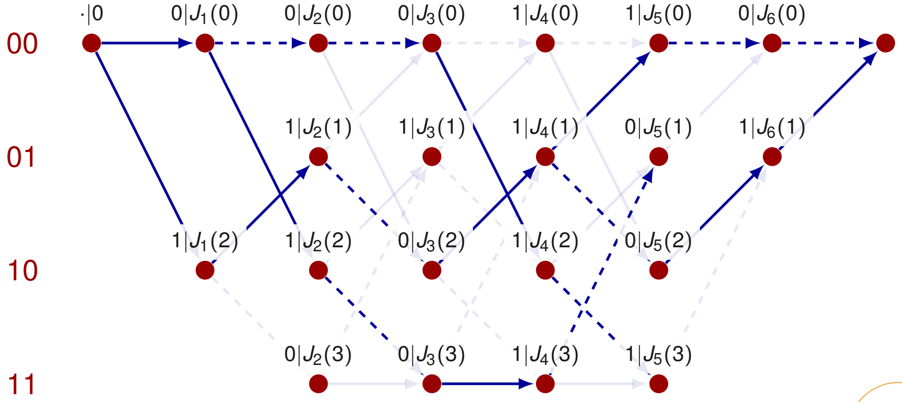


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\mathrm{argmin}} \,|\, \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

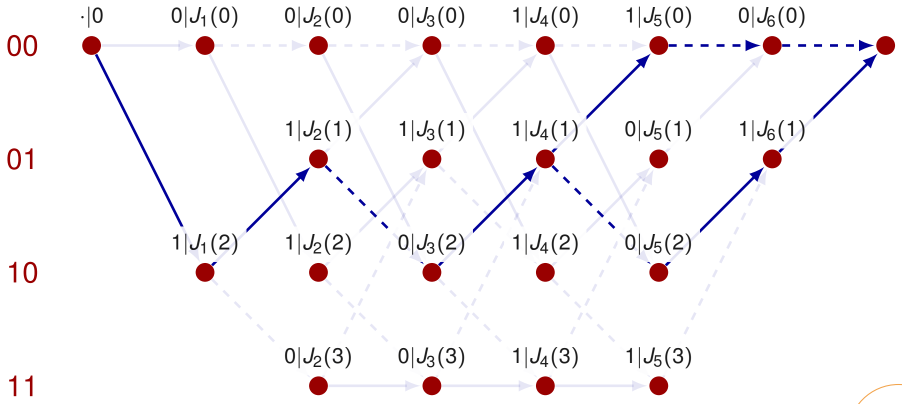


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\text{argmin}} \,|\, \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T \right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

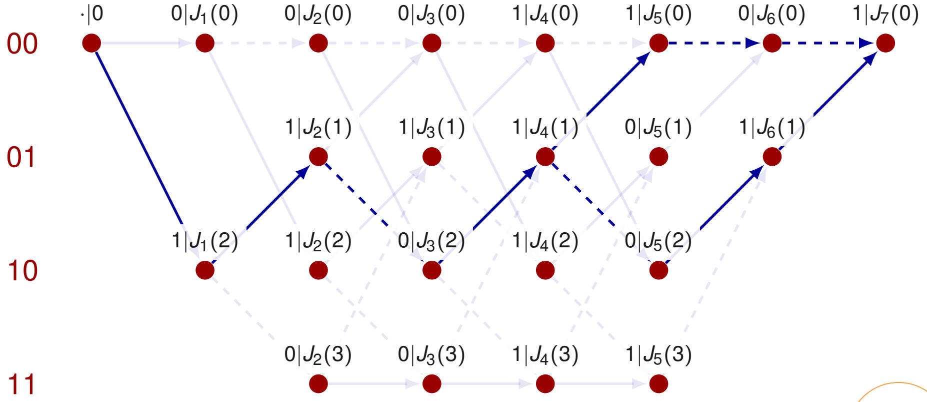


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\text{argmin}} \,|\, \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \rightarrow s_n)\mathbf{y}_{n-1}^T \right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

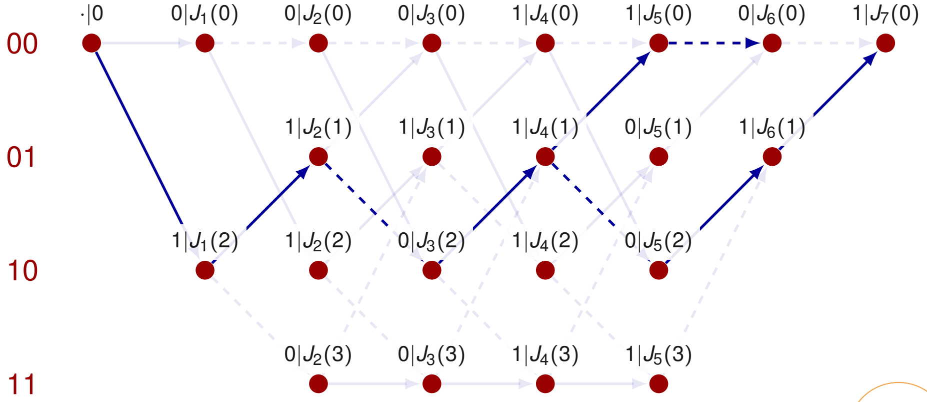


Algorithme de Viterbi

1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \to s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$

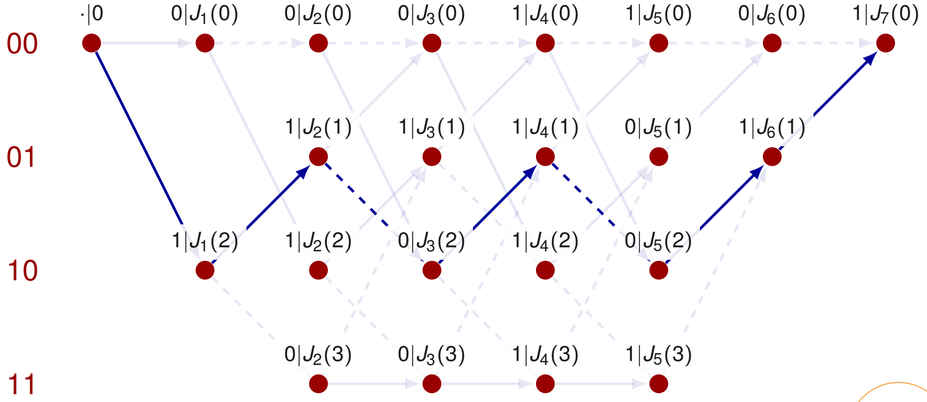


Algorithme de Viterbi

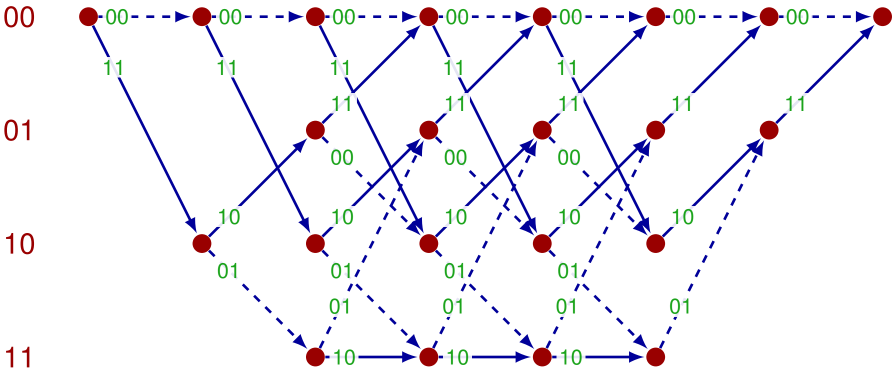
1. Pour chaque s_n, calculer :

$$\hat{u}_{n-1}(s_n)|J_n(s_n) = \underset{u\in\{0,1\}}{\operatorname{argmin}} \mid \min_{u\in\{0,1\}} \left[J_{n-1}(s_{n-1}^{(u)}) + \mathbf{c}_{n-1}(s_{n-1}^{(u)} \rightarrow s_n)\mathbf{y}_{n-1}^T\right]$$

2. Lorsque $n = L$, calculer $\hat{u}_{L-1} = \hat{u}_{L-1}(s_L)$, puis $\hat{u}_{n-1} = \hat{u}_{n-1}(s_n^{(\hat{u}_n)})$



Avec des valeurs



Soit le signal observé $\mathbf{y} = [-1,\ -1,\ -1,\ -1,\ 1,\ 1,\ -1,\ 1\ 1,\ 1,\ -1,\ 1,\ -1,\ -1]$, sachant que $\mathbf{u} = [1,\ 1,\ 0,\ 1,\ 0]$, reçoit-on le message sans erreur ?

Dernier QCM

Comment avez-vous trouvé ce cours ?

- A Très difficile
- B Difficile
- C Moyen
- D Simple
- E Très simple

#QDLE#S#ABCDE#30#

Plan

1. Previously on TS226 . . .
2. Décodage du Maximum de Vraisemblance des codes convolutifs
3. Turbo-Codes