Trellis complexity and symmetry

a) Complexity measures
   - State complexity: Number of states
     - State space dimension profile $(\rho_0, \rho_1, \ldots, \rho_n)$
     - The maximum state space dimension ρ_{max}
   - Branch complexity: Number of branches

b) The Wolf bound
   - $\rho_{max} \leq \min\{k, n-k\}$

The trellis of the dual code

a) C is an $[n,k]$ linear code, C^* its dual code (an $[n,n-k]$ linear code)

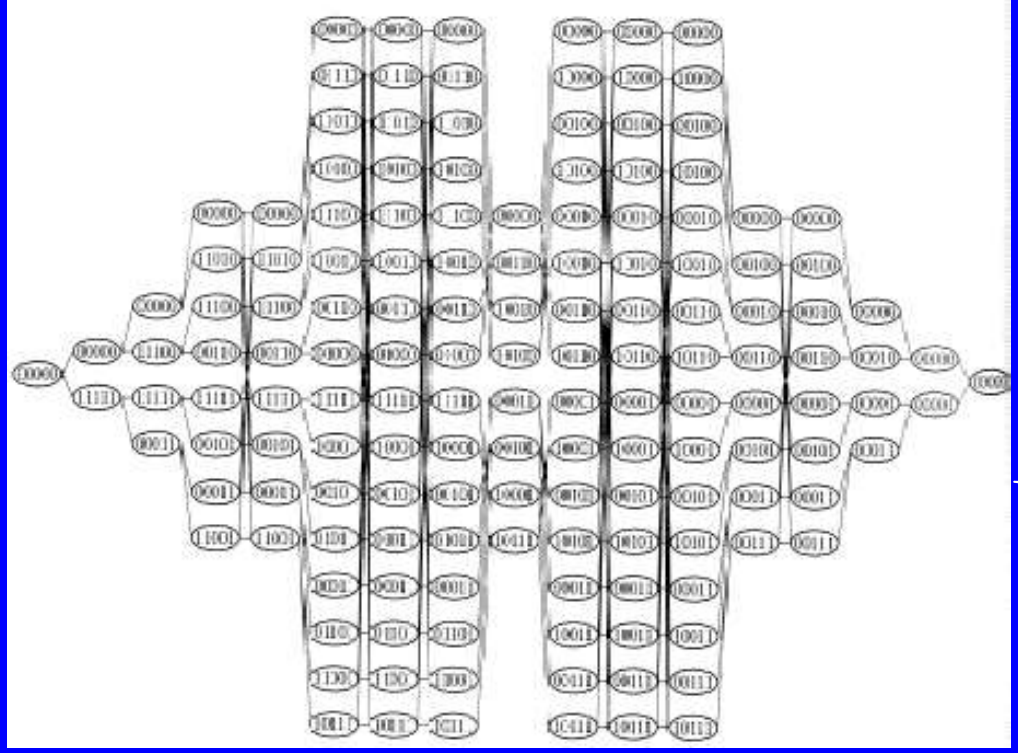
b) $\Sigma_i(C)$ and $\Sigma_i(C^*)$ are the state spaces at time i of C and C^*, resp.

c) Theorem: $|\Sigma_i(C)| = |\Sigma_i(C^*)|$ (i.e. $\rho_i = \rho_i^*$), for all i, $0 \le i \le n$

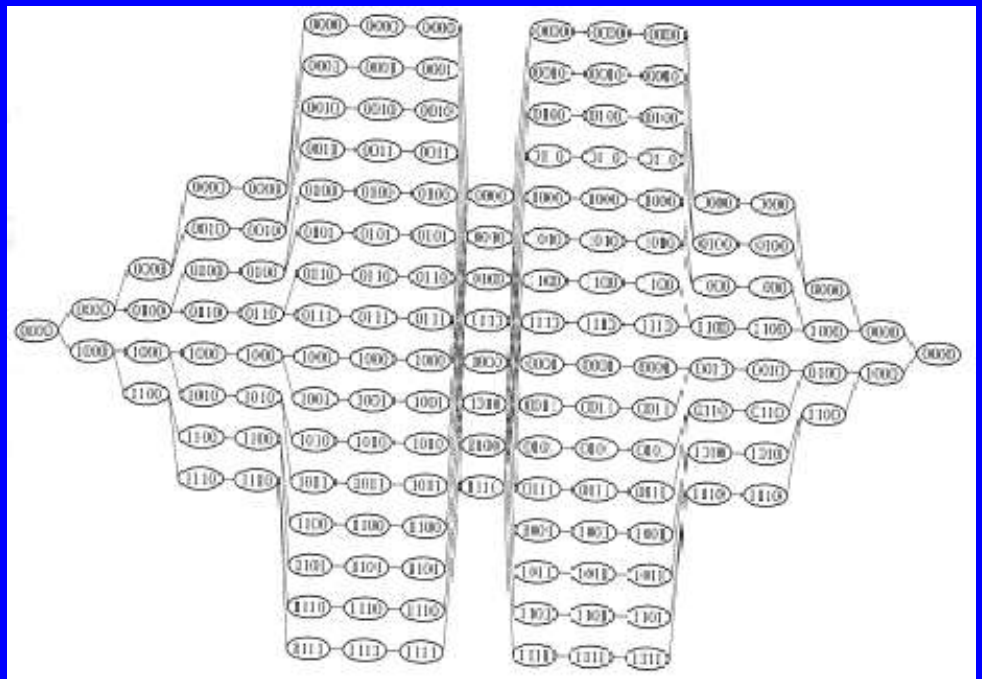
- Proof:
- States in $\Sigma_i(C^*)$ correspond cosets in the partition $p_{0,i}(C^*)/C^{*,\mathrm{tr}}_{0,i}$
- $\rho_i^* = k(p_{0,i}(C^*)) - k(C^{*,\mathrm{tr}}_{0,i})$
- $p_{0,i}(C^*)$ is generated by $\mathbf{H}_i$, the parity check matrix for $C^{\mathrm{tr}}_{0,i}$. Thus, $p_{0,i}(C^*)$ and $C^{\mathrm{tr}}_{0,i}$ are dual codes, and $p_{0,i}(C)$ and $C^{*,\mathrm{tr}}_{0,i}$ are also dual codes. Thus,
- $\rho_i^* = k(p_{0,i}(C^*)) - k(C^{*,\mathrm{tr}}_{0,i}) = (i - k(C^{\mathrm{tr}}_{0,i}) - (i - k(p_{0,i}(C)))$
  $= k(p_{0,i}(C)) - k(C_{0,i}) = k - k(C_{i,n}) - k(C_{0,i}) = \rho_i$

d) Note: State complexities equal; branch complexities different

Example



[16,11] RM



[16,5] RM

Minimal trellises

a) In principle there may be different trellises corresponding to the same code

b) A minimal trellis for a given code is a trellis such that its state complexity is $(\rho'_0, \rho'_1, \ldots, \rho'_n)$ and, for any other trellis for that code with state complexity $(\rho_0, \rho_1, \ldots, \rho_n)$, it holds that for all i, $0 \le i \le n$, $\rho'_i \le \rho_i$

c) A minimal trellis is unique up to graph isomorphism

d) In general, for a given code and for a given order of the coordinates, it is simple to find a minimal trellis (provided the trellis itself is simple). In fact, ρ'_i is given by the dimensions of the past and future subcodes, which is fixed for any given i

e) If we are free to play with the order of the coordinates, the problem of finding an optimum bit ordering is much more challenging

Bit ordering

Permuting the order of the coordinates does not change

- code rate or
- distance properties

However, it may change

- the dimensions of future and past subcodes at any time instant, and may thus change the trellis complexity

An optimum permutation

- Gives the smallest possible state space at each time instant
- Is hard to find
- Is known for some classes of codes, but unknown for others
- May not even exist for a given code

Branch complexity

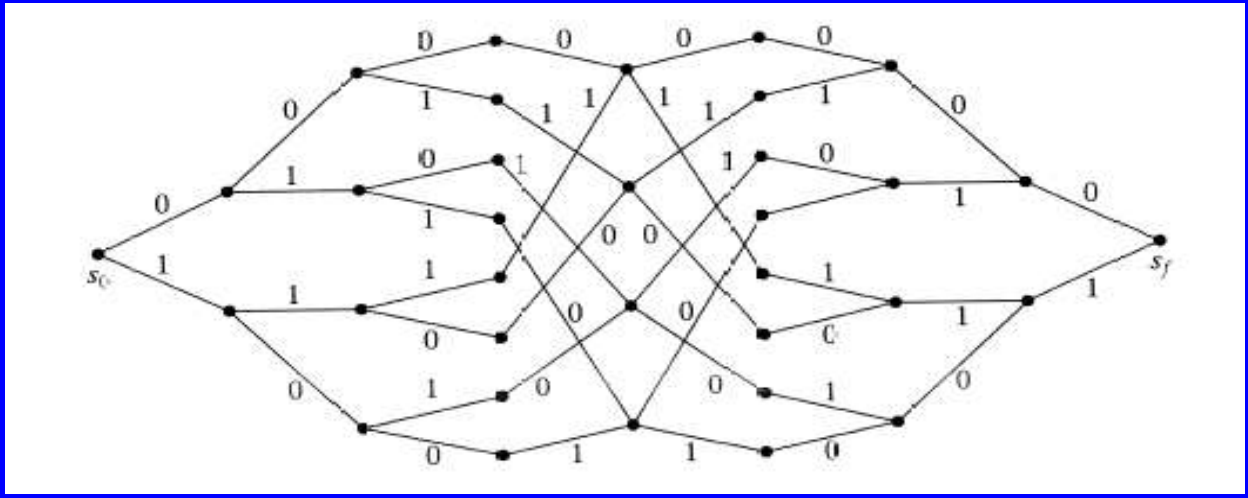
a) Total number of branches in the trellis is the branch complexity

b) Determines (to a large extent) the workload of a trellis based decoding algorithm

c) Let $I_i = 2$ if there is a current information bit a^* corresponding to output bit i, $I_i = 1$ otherwise

d) Then, the branch complexity is $\sum_i I_i \cdot 2^{\rho_i}$

e) A minimal trellis diagram has the smallest branch complexity



Cyclic codes

a) A linear code with the property that any rotational shift of a codeword is also a codeword is said to be cyclic

b) A cyclic code is generated by a generator polynomial

$$\mathbf{g}(x) = 1 + g_1 x + g_2 x^2 + \ldots + g_{n-k-1} x^{n-k-1} + x^{n-k}$$

c) The corresponding generator matrix is in TO form

$$\mathbf{G} = \begin{pmatrix} 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & & & \\ & 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & & \\ & & \ddots & & \ddots & & & & \ddots & \\ & & & 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 \end{pmatrix}$$

Cyclic codes (cont.)

$$\mathbf{G}=\begin{pmatrix} 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & & & \\ & 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & & \\ & & \ddots & & \ddots & & & & \ddots & \\ & & & 1 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 \end{pmatrix}$$

- Time span $\tau(\mathbf{g}_i) = [i, n-k+i+1]$, bit span $\phi(\mathbf{g}_i) = [i, n-k+i]$
- If $k > n-k$, then
  $(\rho_0, \rho_1, \ldots, \rho_n) = (0,1,\ldots,n-k-1,n-k,n-k,\ldots,n-k,n-k-1,\ldots,1,0)$
- If $k \le n-k$, then
  $(\rho_0, \rho_1, \ldots, \rho_n) = (0,1,\ldots,k-1,k,k,\ldots,k,k-1,\ldots,1,0)$
- Cyclic codes meet the Wolf bound with equality (worst possible)

Symmetry considerations

a) Assume that the TOGM **G** has the following symmetry property:
   - For each row **g** with bit span $\phi(\mathbf{g}) = [a,b]$, there is also a row **g'** with bit span $\phi(\mathbf{g'}) = [n\text{-}1\text{-}b, n\text{-}1\text{-}a]$
   - Thus, **g** is active at time i (i.e. $i \in [a+1,b]$) iff $n\text{-}i \in [n\text{-}b, n\text{-}1\text{-}a]$, i.e., $n\text{-}i$ is within the active time span of **g'**, and $\rho_i = \rho_{n-i}$

b) Put **G** in reverse trellis-oriented form and rotate the matrix 180 degrees counterclockwise to produce **G"**. Then, $\phi(\mathbf{g}_i'') = \phi(\mathbf{g}_i)$ and thus $\tau_a(\mathbf{g}_i'') = \tau_a(\mathbf{g}_i)$

c) The trellis possesses mirror symmetry
   - If n is even (odd), the last $n/2$ ($(n\text{-}1)/2$) trellis sections form the mirror image of the first $n/2$ ($(n\text{-}1)/2$) sections

d) The trellis of cyclic codes possesses mirror symmetry

Trellis sectionalization

a) So far we have considered n-section trellises, i.e., one code bit per trellis section

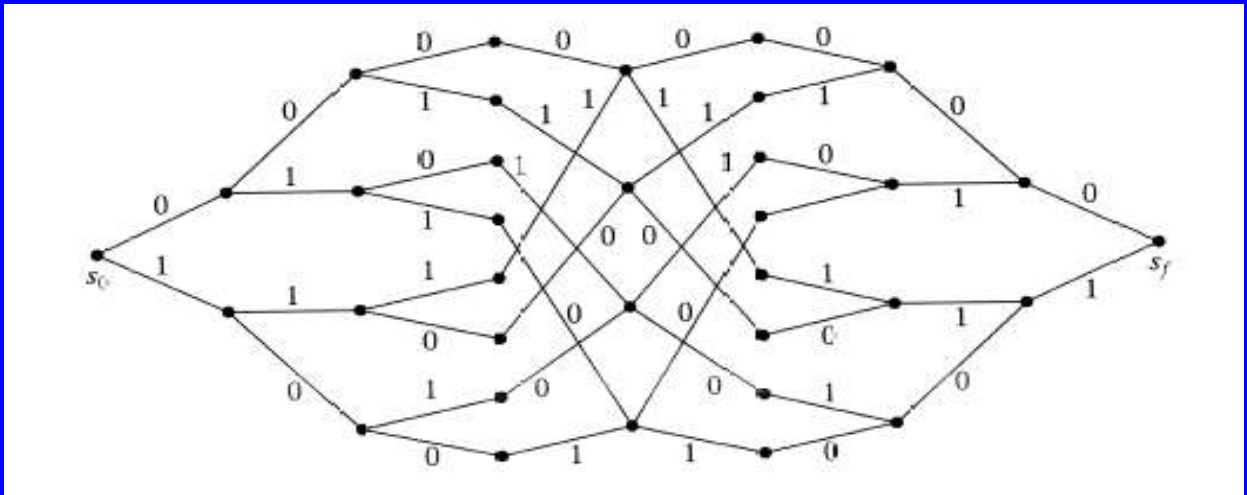
b) Sectionalization combines adjacent bits and provides a trellis with fewer time instances, in order to simplify the decoders

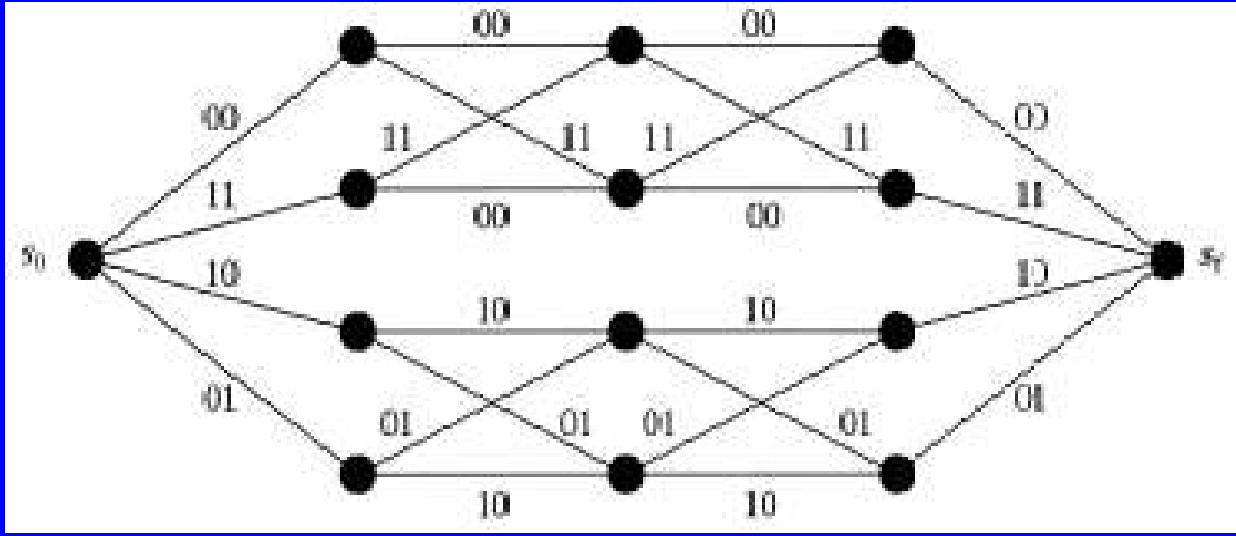
c) Let $\Lambda=\{t_0=0, t_1,\ldots, t_\nu=n\} \subseteq \Gamma$, for $\nu \leq n$. Delete all state spaces (and their adjacent branches) at time instances not in Λ, and connect every pair of states, one at time t_j and one at time t_{j+1}, iff there was a path between these states in the original trellis and label the new branches by the old path labels (parallel edges could occur in the new trellis)

d) Uniform sectionalization: All sections have the same length in bits

e) Sectionalization may
   - Reduce state complexity
   - Increase branch complexity

Sectionalization: Example





Parallel decomposition

a) The sectionalized trellis may be densely connected, making a hardware implementation difficult

b) The motivation for parallel decomposition is to select trellis subsections to consist of structurally identical components which are not interconnected (simplifies hardware implementation)

c) Code C with TOGM $\mathbf{G}$, let $\mathbf{g}$ be a row of $\mathbf{G}$, and let $\mathbf{G}_1 = \mathbf{G} \backslash \{\mathbf{g}\}$

d) $\mathbf{G}$ TOF $\Rightarrow \mathbf{G}_1$ TOF

e) $\rho_i(C_1) = \rho_i(C)$-1 for $i \in \tau_a(\mathbf{g})$, $\rho_i(C_1) = \rho_i(C)$ for $i \notin \tau_a(\mathbf{g})$

f) $I_{\max}(C) = \{i: \rho_i(C) = \rho_{\max}(C)\}$

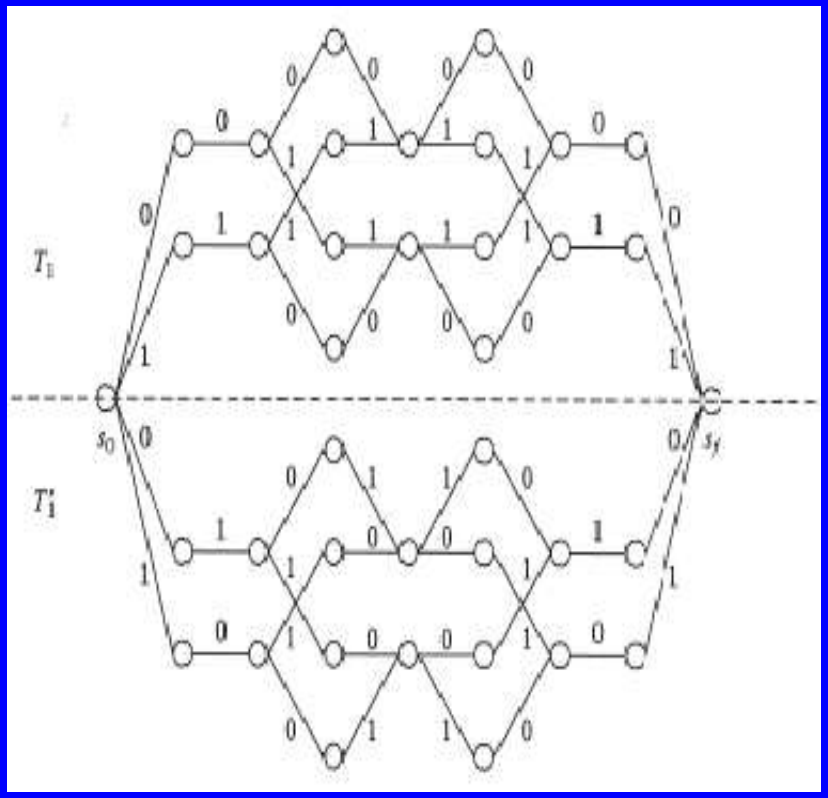
g) Thm 9.1: If there exists a row $\mathbf{g}$ such that $\tau_a(\mathbf{g}) \supseteq I_{\max}(C)$, then sectionalization produces two parallel and identically structured subtrellises, one for C_1 and one for $C_1 + \mathbf{g}$, such that

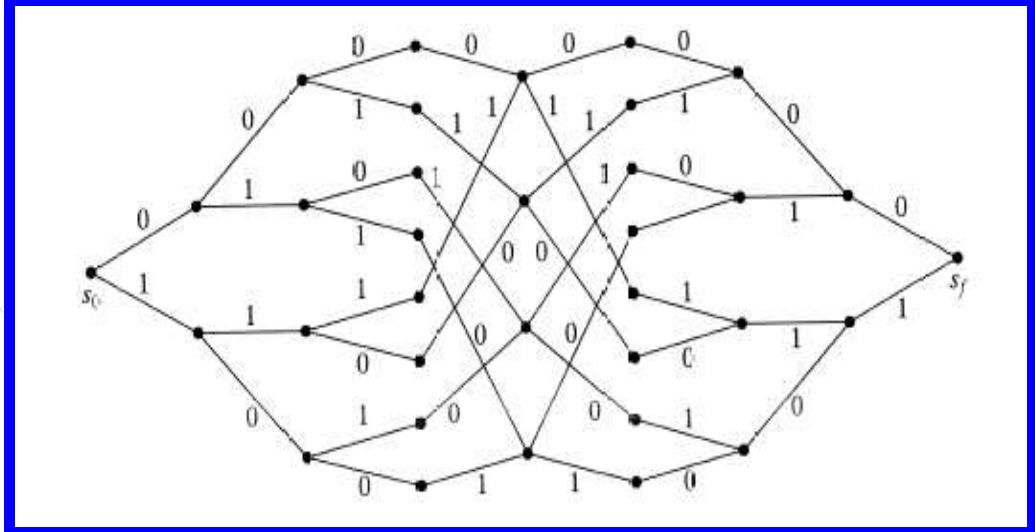
- $\rho_{\max}(C_1) = \rho_{\max}(C) - 1$, and
- $I_{\max}(C_1) = I_{\max}(C) \cup \{i: \rho_i(C) = \rho_{\max}(C) - 1, i \notin \tau_a(\mathbf{g})\}$

Example

a) Thus, it is possible to decompose the trellis in smaller independent components

$$G' = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$





Generalization of Thm 9.1

a) $R(C) = \{\mathbf{g}: \tau_a(\mathbf{g}) \supseteq I_{\max}(C)\}$

b) Theorem 9.2: Theorem 9.1 can be applied repeatedly

c) Corollary 9.2.1: Maximum number of parallel isomorphic subtrellises such that the new total state space dimension does not exceed $\rho_{\max}(C)$ is upper bounded by $2^{|R(C)|}$

d) Corollary 9.2.2: In a minimal ν-section trellis with section boundary locations in $\Lambda=\{t_0, t_1,\ldots, t_\nu\}$, the $\log_2$ of the number of parallel isomorphic subtrellises is equal to the number of rows in a TOGM with active time spans containing the section boundary locations in $\{t_1,\ldots, t_{\nu-1}\}$

Low-weight subtrellises

a) A code with (ordered) codeword weights 0, w_1, w_2,…, $w_m = n$

b) The w_t–weight subtrellis is obtained by pruning from the original trellis

- Every state not involved in a codeword of weight ≤ w_t
- Every branch not involved in a codeword of weight ≤ w_t
- Achieved using the Viterbi algorithm in a forward and a backward fashion

c) If necessary, split the zero state at each time instant into two states to avoid codewords of weights larger than w_t that merge with and diverge from the zero state in the original trellis

d) Such trellises are useful for devising a suboptimum (but simpler) decoding algorithm

Cartesian product

a) Trellis construction for codes designed by combining several component codes by
   - Interleaving
   - Product
   - Direct-sum
b) Trellis is constructed by taking the Cartesian product of component trellises

Interleaved codes

a) $C^{\lambda} = C_1 * C_2 * \ldots * C_{\lambda}$ formed by interleaving

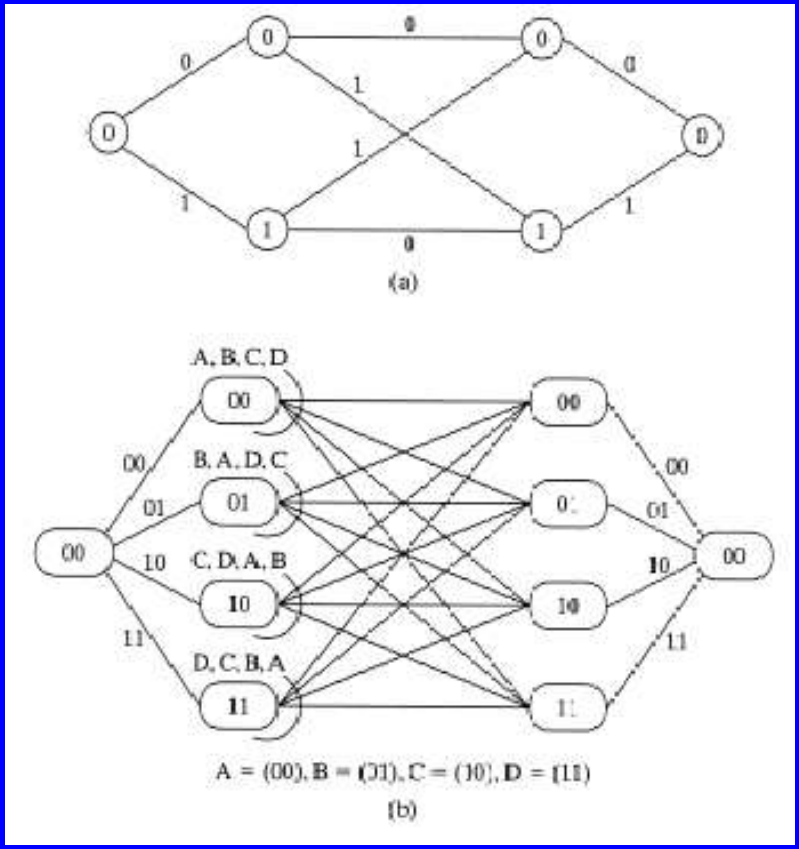
b) C^{λ} is a $(\lambda n, k_1 + \ldots + k_{\lambda})$ linear block code

c) We have n-section trellises for each component code

d) In the trellis for the interleaved code:

- The state space is the Cartesian product of the original component state spaces
- There is a branch in the new trellis if there is a branch in a component trellis linking corresponding component states for all component codes
- Label the branches by the concatenation of the component labels

Interleaved codes: Example



Note: This is utterly impractical – use the simpler trellises! Can decode each code individually (except on a burst channel?)

Product codes

a) $C = C_1 \times C_2$ formed by a product construction:

- Form a matrix where each of the k_2 n_1-bit rows $\in C_1$
- Then, append parity bits to the columns so that each $\in C_2$
- This is an $(n_2 n_1, k_2 k_1)$ linear block code

b) An n_1-section trellis for the product code can be formed by Cartesian product as follows:

– Regard the top k_2 rows as an interleaved code and construct a trellis for the interleaved code using Cartesian product

– Each k_2-tuple branch label is encoded into a codeword in C_2

c) Again totally impractical?...No; this code needs to be decoded in full

Direct-sum construction

a) Component codes C_1 and C_2 each of length n

b) $C_1 \cap C_2 = \{\mathbf{0}\}$

c) The direct-sum code C formed by

$$C = C_1 \oplus C_2 = \{\mathbf{u}+\mathbf{v}: \mathbf{u} \in C_1, \mathbf{v} \in C_2\}$$

is an $(n, k_1 + k_2)$ linear block code with minimum distance $\leq \min(d_1, d_2)$

d) Trellis: Cartesian product again, or more precisely, Shannon product

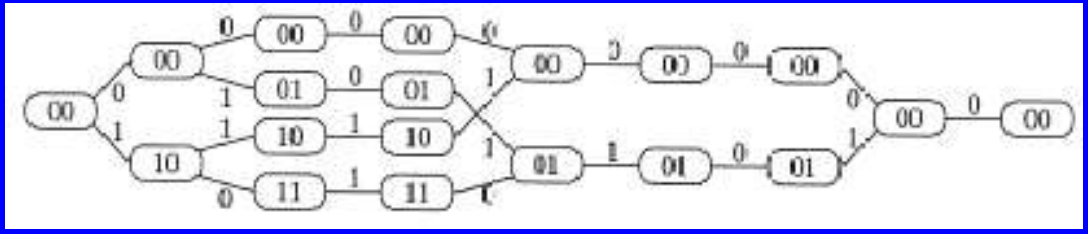
– Label the branches by the sum of the component labels

e) Can be generalized to m component codes, but the conditions stated at the bottom of page 386 are NOT sufficient for $K = K_1 + \ldots + K_m$

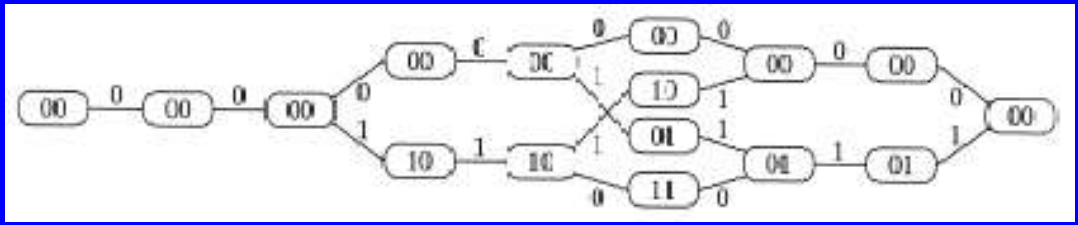
– The same codeword is represented by several paths in the trellis

f) The Shannon product is not necessarily the minimal n-section trellis for C. Conditions for minimality exist

Direct sum: Example



$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$