Trellis based decoding of linear block codes

- Previous discussion on trellis based decoding of convolutional codes applies directly to any code with a (moderately complex) trellis. We have considered the following algorithms:
  - Viterbi
  - SOVA
  - MAP (or BCJR)
  - Log-MAP
  - Max-log-MAP
- Linear block codes have irregular trellises
  - May be more difficult to design HW decoder

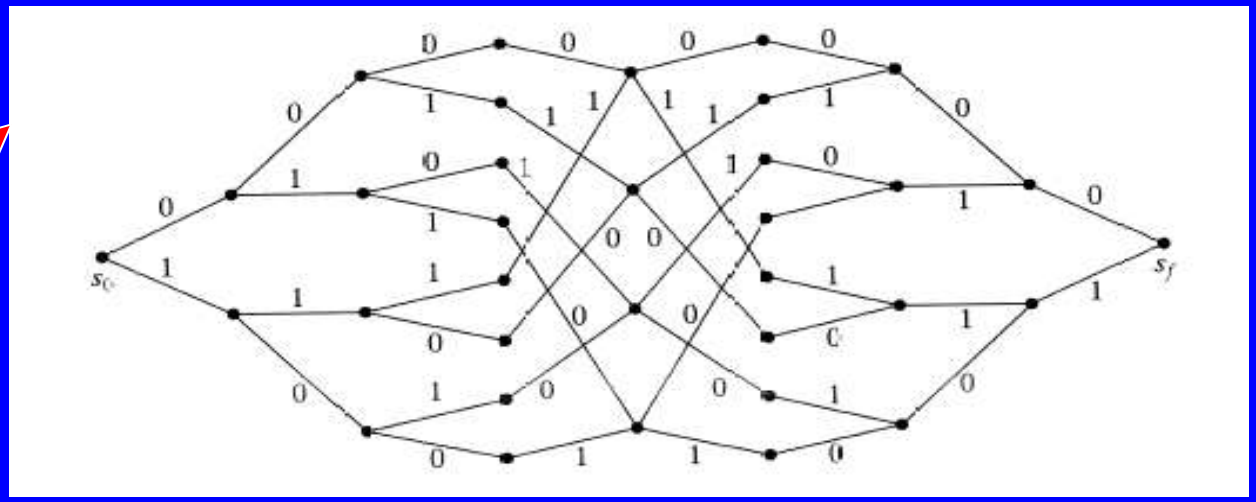
Computational complexity of Viterbi

- Let $I_i = 2$ if there is an information symbol corresponding to output bit i, $I_i = 1$ otherwise
- The branch complexity is the number of (one-symbol) branches
- Thus, it is also equal to the number of additions in the Viterbi algorithm N_a, and it follows that $N_a = \sum_{i=0\ldots n-1} I_i \cdot 2^{\rho_i}$
- Let $J_i = 0$ if there is only one branch entering each state at time $i+1$, $J_i = 1$ otherwise
- Then, the number of comparisons made in the Viterbi algorithm is $N_c = \sum_{i=0\ldots n-1} J_i \cdot 2^{\rho_{i+1}}$

Example: (8,4) RM

$N_a = 44$

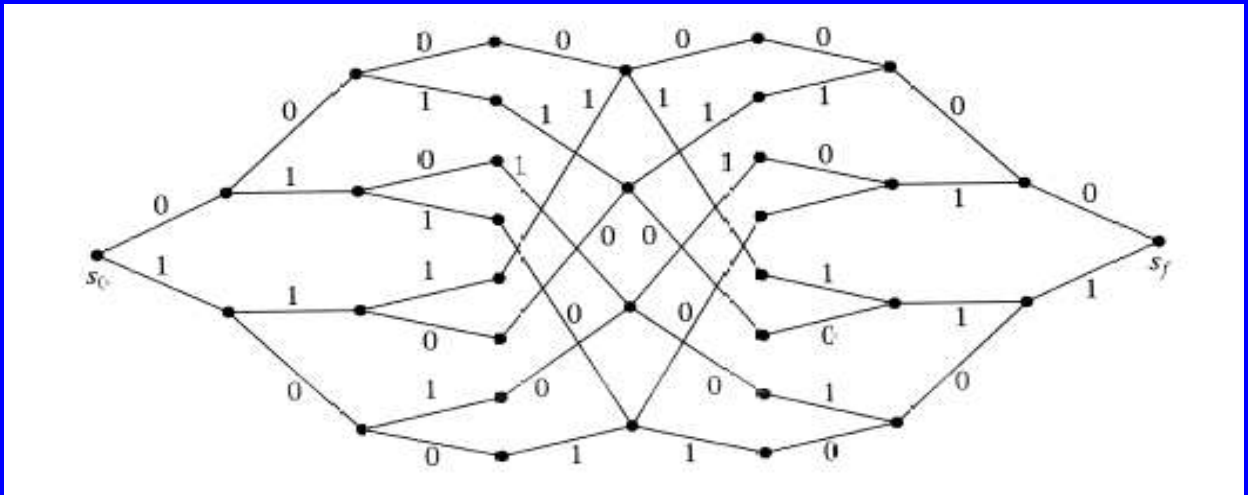
$N_c = 11$

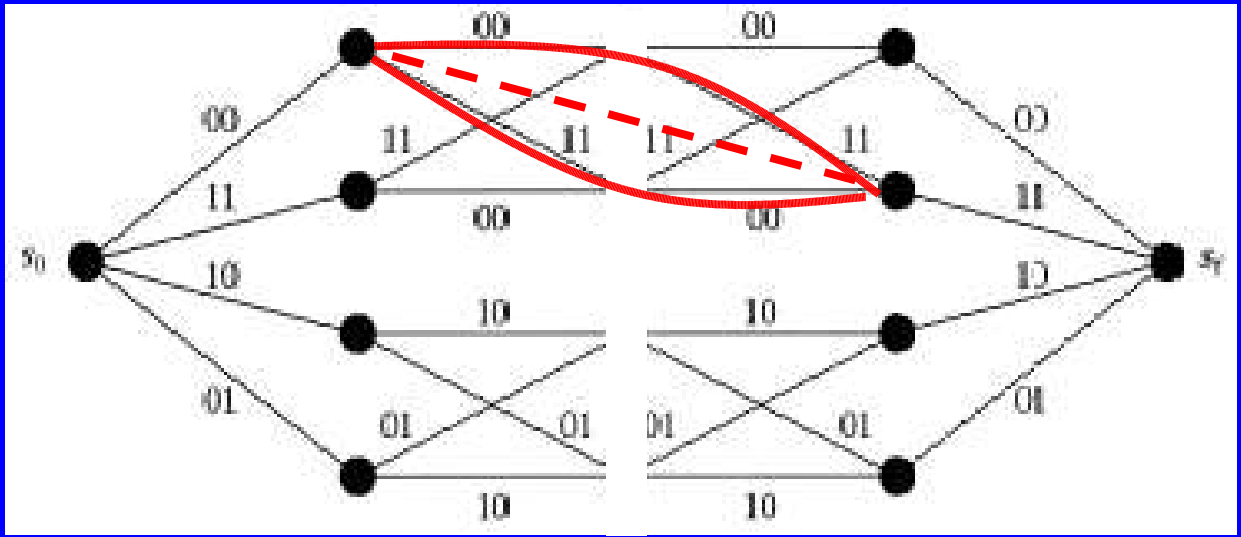


Trellis sectionalization

- Sectionalization combines adjacent bits and provides a trellis with fewer time instances in order to simplify the decoder
- Let $\Lambda=\{t_0, t_1, \ldots, t_\nu\}$ 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 the other at time t_{j+1} iff there was a path between these states in the original trellis. Label the new branches by the old path labels
- A set of parallel paths forms a composite branch

Trellis sectionalization: Example





Three trellis sections

Sectionalized trellis decoding

- For each composite branch:
  - Find the best single branch among those that form the composite branch. Make a note of who is the winner
  - Each composite branch is replaced by the winning branch
- The rest proceeds as the ordinary Viterbi algorithm

Complexity of sectionalized trellis decoding

- The complexity depends on the choice of section boundaries
- An optimum sectionalization is a sectionalization that requires the minimum number of additions + comparisons
- Let $\varphi(x, y)$ = the number of computations needed to process the trellis section from time x to time y in any sectionalized trellis with $x, y \in \Lambda$ and $x+1, x+2, \ldots, y-1 \notin \Lambda$
- Let $\varphi_{\min}(x, y)$ = the smallest number of computations needed to process the trellis section(s) from time x to time y in any sectionalized trellis with $x, y \in \Lambda$
- $\varphi_{\min}(0, y) = \min\{\varphi(0, y), \min_{0<x<y}\{\varphi_{\min}(0, x) + \varphi(x, y)\}\}$, for $1 < y \leq n$, and $\varphi(0, 1)$ for $y = 1$ (Lafourcade & Vardy)
- Algorithm:
  - Calculate $\varphi(x, y)$ for $0 \leq x < y \leq n$
  - Use these values to calculate $\varphi_{\min}(0, y)$ for successive values of y

Examples

Code	Viterbi based on 64-section trellis	Viterbi based on Lafourcade–Vardy algorithm
$RM(2, 6)$ $(64, 22)$	425,209	101,786
$RM(3, 6)$ $(64, 42)$	773,881	538,799
$RM(4, 6)$ $(64, 57)$	7,529	6,507

Recursive ML decoding (RMLD)

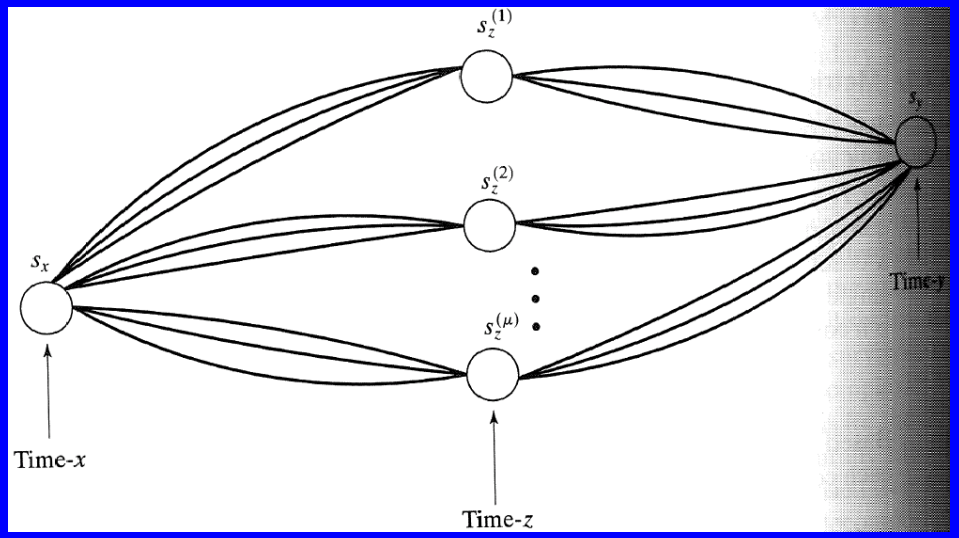
- Uses a sectionalized trellis and recursive combining of path metrics
- Start with a minimal trellis for an (n,k) linear block code
- Let $L(s_x, s_y)$ be the set of paths (forming one composite path) from state s_x at time x to state s_y at time y
- $L(s_x = \text{zero state}, s_y = \text{zero state})$ is the linear block code $C^{tr}_{x,y}$
- Each $L(s_x, s_y)$ is a coset of $C^{tr}_{x,y}$, i.e., a coset in $p_{x,y}(C) / C^{tr}_{x,y}$
- The number of distinct cosets is $2^{k(p_{x,y}(C)) - k(C_{x,y})}$
- Thus, each coset appears $2^{k - k(C_{0,x}) - k(C_{y,n}) - k(p_{x,y}(C))}$ times as a composite path
- Note that the expression in (9.19) on page 357 in the book is wrong! If the expression was correct, then each coset would appear only a single time as a composite path. (9.19) is correct when $i = 0$ and / or $j = n$

Composite path metric table (CPMT)

- Store, in the table $CPMT_{x,y}$, for each distinct composite path $L(s_x, s_y)$
  - The label of the best path within the composite path, denoted by $l(L(s_x, s_y))$
  - The metric of the best path within the composite path, denoted by $m(L(s_x, s_y))$
- The table $CPMT_{0,n}$ will contain just one path which is the ML path
- Construct the table $CPMT_{x,y}$ by

1. Computing metrics for each path within the composite paths. This requires $2^{k(C_{x,y})}(y\text{-}x\text{-}1)$ additions and $2^{k(C_{x,y})}\text{-}1$ comparisons, i.e., $2^{k(C_{x,y})}(y\text{-}x) - 1$ operations in total for each composite path. Use this technique only when $y\text{-}x$ is small
2. Combining tables $CPMT_{x,z}$ and $CPMT_{z,y}$. Use this technique when $y\text{-}x$ is larger

RMLD (cont.)



- $L(s_x, s_y) = \cup_{s_z} L(s_x, s_z) \circ L(s_z, s_y)$
- $m(L(s_x, s_y)) = \max_{s_z}\{m(L(s_x, s_z)) + m(L(s_z, s_y))\}$
- $l(L(s_x, s_y)) = l(L(s_x, s^*_z)) \circ l(L(s^*_z, s_y))$, where s^*_z gives the maximum above
- This computation requires μ_z additions and μ_z -1 comparisons, where μ_z = the number of states at time z on some path from state s_x to state s_y
- In fact, $\mu_z = 2^{k(C_{x,y}) - k(C_{x,z}) - k(C_{z,y})}$
- Thus, forming the table $\mathrm{CPMT}_{x,y}$ requires $2^{k(p_{x,y}(C)) - k(C_{x,y})}(2\mu_z - 1)$ operations

Complexity comparisons

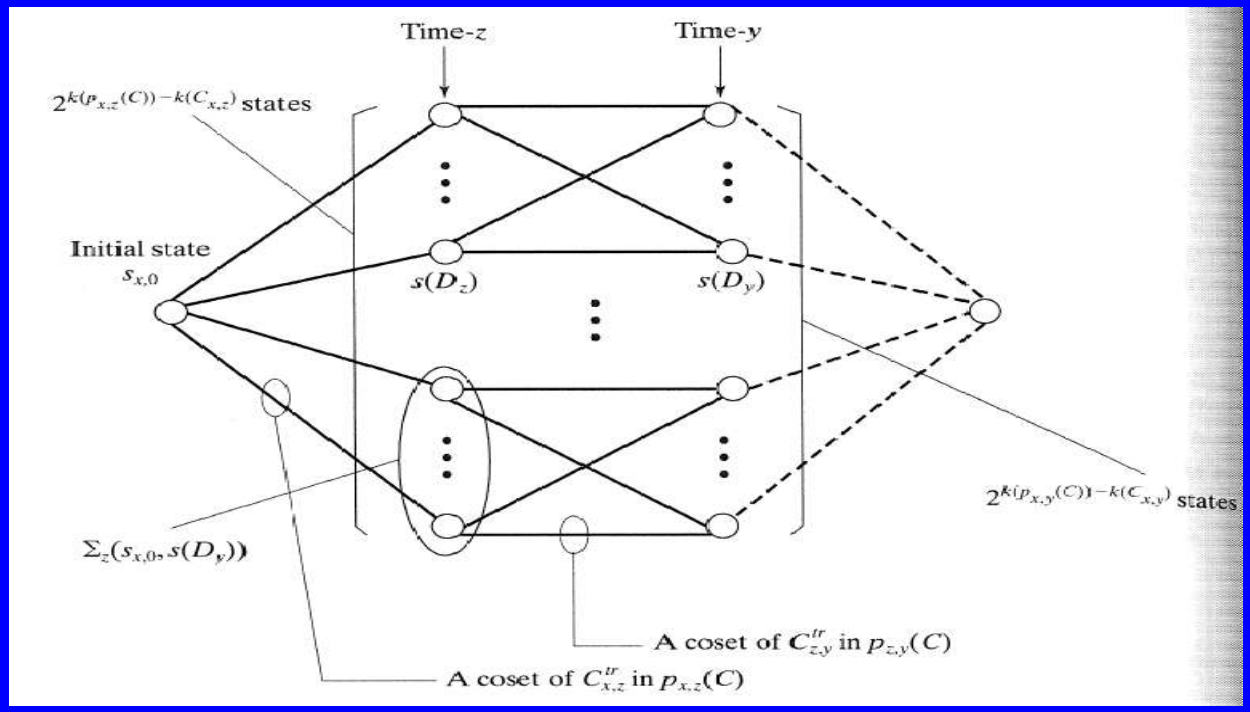
- The amount of computation for each composite path $L(s_x, s_y)$ is
  - $2^{k(C_{x,y})}(y\text{-}x) - 1$ when using the direct approach
  - $2\mu_z - 1 = 2^{k(C_{x,y}) - k(C_{x,z}) - k(C_{z,y}) + 1} - 1$ when combining tables
- Recursive computation is always faster if $y\text{-}x > 2$, independent of the value of z. Note that the complexity is the same if $y\text{-}x = 2$
- Thus, the direct approach should only be used at the beginning of the recursion process
- How do we combine metric tables efficiently?
  - We use a special two-section trellis for the code $p_{x,y}(C)$

The special trellis: Structural properties

- The special trellis $T(\{x, z, y\})$ is a trellis for $p_{x,y}(C)$ that has a single initial state $s_{x,0}$ at time x and multiple final states at time y
- There is one-to-one correspondence between states in Σ_z and the cosets in $p_{x,z}(C) / C^{tr}_{x,z}$. The composite path connecting $s_{x,0}$ to $s(D_z)$ is $L(s_{x,0}, s(D_z))$, where $s(D_z)$ is a state in Σ_z and $D_z = L(s_{x,0}, s(D_z))$ is a coset in $p_{x,z}(C) / C^{tr}_{x,z}$
- There is one-to-one correspondence between states in Σ_y and the cosets in $p_{x,y}(C) / C^{tr}_{x,y}$. The composite path connecting $s_{x,0}$ to $s(D_y)$ is $L(s_{x,0}, s(D_y))$, where $s(D_y)$ is a state in Σ_y and $D_y = L(s_{x,0}, s(D_y))$ is a coset in $p_{x,y}(C) / C^{tr}_{x,y}$
- If a state $s(D_z)$ at time z is connected to a state $s(D_y)$ at time y, then the connecting composite path $L(s(D_z), s(D_y))$ is a coset of $p_{z,y}(C) / C^{tr}_{z,y}$. Note that every coset in $p_{z,y}(C) / C^{tr}_{z,y}$ appears as a composite path between a state in Σ_z and a state in Σ_y
- For every state $s(D_y)$ at time y, there is a set of μ_z states in Σ_z that connect the initial state $s_{x,0}$ to state $s(D_y)$. We denote this set of states by $\Sigma_z(s_{x,0}, s(D_y))$

The special trellis: Structural properties

- $L(s_{x,0}, s(D_y)) = \cup_{s(D_z) \in \Sigma_z(s_{x,0}, s(D_y))} L(s_{x,0}, s(D_z)) \circ L(s(D_z), s(D_y))$
- $m(L(s_{x,0}, s(D_y))) = \max_{s(D_z) \in \Sigma_z(s_{x,0}, s(D_y))} \{m(L(s_{x,0}, s(D_z))) + m(L(s(D_z), s(D_y)))\}$
- Using the above expressions, the metric table $\mathrm{CPMT}_{x,y}$ can be constructed from the metric tables $\mathrm{CPMT}_{x,z}$ and $\mathrm{CPMT}_{z,y}$ using the special two-section trellis and ACS operations



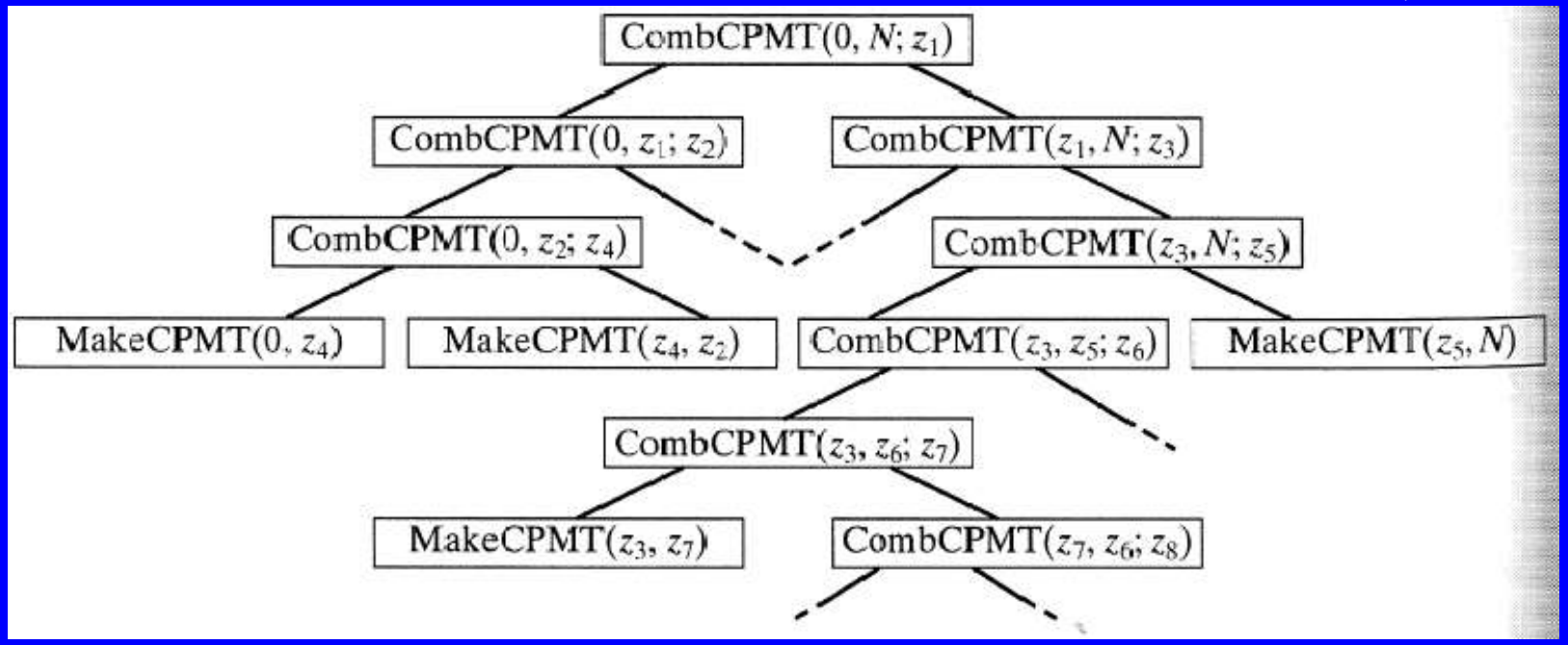
The special trellis: Construction

- Choose a basis $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{k(p_{x,y}(C))}\}$ for $p_{x,y}(C)$ such that the first $k(C_{x,y})$ vectors form a basis of $C^{\mathrm{tr}}_{x,y}$
- Let $n_{x,y} = y - x + k(p_{x,y}(C)) - k(C_{x,y})$
- Construct the matrix $\mathbf{G}(x, y)$ and a three level trellis for the corresponding code with section boundary locations at x, z, y, and $x + n_{x,y}$

$$\mathbf{G}(x, y) = \begin{bmatrix} \mathbf{g}_1 & \vdots & \\ \vdots & \vdots & \mathbf{0} \\ \mathbf{g}_{k(C_{x,y})} & \vdots & \\ \cdots & \cdots & \cdots \\ \mathbf{g}_{k(C_{x,y})+1} & \vdots & \\ \vdots & \vdots & I \\ \mathbf{g}_{k(p_{x,y}(C))} & \vdots & \end{bmatrix},$$

An RMLD-(I,V) algorithm

- Divide the code into very short sections
- Then, apply the MakeCPMT-I(x, y) procedure to these short sections
  - The MakeCPMT-I(x, y) procedure is ”brute-force”
- Then, apply the CombCPMT-V(x, y; z) procedure to larger sections
  - The CombCPMT-V(x, y; z) procedure uses the Viterbi algorithm (on the special trellis) to combine the tables $CPMT_{x,z}$ and $CPMT_{z,y}$



Optimum sectionalization

- How do we choose the values for z?
- A sectionalization (or the choices for z) that results in the smallest number overall computational complexity is called an optimum sectionalization
- To find an optimum sectionalization, we use the Lafourcade-Vardy algorithm in the same manner as before. See below
- Note: The RMLD-(I,V) algorithm with an optimum sectionalization is more efficient than the Viterbi algorithm based on the Lafourcade-Vardy's optimum sectionalization of the bit-level trellis

$$\Psi_{min}(x, y) \triangleq \begin{cases} \Psi_M^{(I)}(x, y), & \text{if} \quad y = x + 1, \\ \min\{\Psi_M^{(I)}(x, y), \min_{x<z<y}\{\Psi_{R,min}(x, y; z)\}, & \text{otherwise,} \end{cases}$$

where

$$\Psi_{R,min}(x, y; z) \triangleq \Psi_{min}(x, z) + \Psi_{min}(z, y) + \Psi_c^{(V)}(x, y; z).$$

Complexity comparisons

Code	Viterbi based on 64-section trellis	RMLD-(I,V)	Viterbi based on Lafourcade–Vardy algorithm
$RM(2, 6)$ $(64, 22)$	425,209	78,209	101,786
$RM(3, 6)$ $(64, 42)$	773,881	326,017	538,799
$RM(4, 6)$ $(64, 57)$	7,529	5,281	6,507

Note that the RMLD algorithm allows for parallel processing as well to speed up decoding