Turbo decoding algorithms

Kalle Ruttik

Communications Laboratory

Helsinki University of Technology

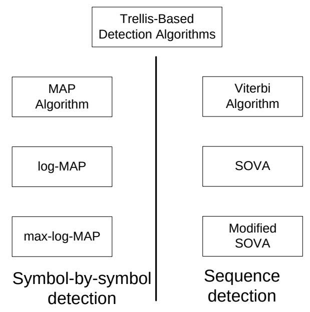
P.O.Box 2300, FIN-02015 HUT, FINLAND

Tel: +358 9 451235418; Fax: +358 9 4512359

e-mail: kalle.ruttik@hut.fi

February 4, 2005

Algorithms for Iterative (Turbo) Data Processing



Requirements

Accept soft-inputs in the form of a priori probabilities or log-likelihood ratios

Produce APP for output data

Soft-Input Soft-Output

- MAP: Maximum A Posteriori (symbol-by-symbol)
- SOVA: Soft Output Viterbi Algorithm

MAP decoding Algorithm

- The a posteriori estimation of the symbol is optimally made by the BCJR algorithm (Bahl, Cocke, Jalinek, Raviv)
- BCJR is a forward-backward MAP algorithm. In Turbo decoding purposes this algorithm is slightly modified.
- BCJR (MAP) algorithm finds the marginal probability that the received bit was 1 or 0.
- Since the bit 1 (or 0) could occur in many different code words, we have to sum over the probabilities of all these code words.
- The decision is made by using the likelihood ratio of these marginal distributions from 1 and 0.
- The calculation can be structured by using trellis diagram.
- For every state sequence there is a unique path trough the trellis and vice versa.
- The objective of the decoder is to examine states s and compute APPs associated with the state transitions.

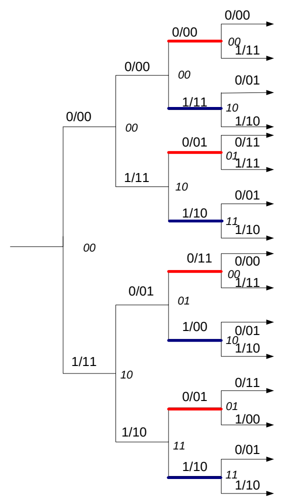


Figure 1: Calculation of the marginal probability in the code tree.

- The probability of the code words is visualised in the code tree.
- For independent bits the probability of one codeword is multiplication of probabilities of the individual bits in the codeword.
- The marginal probability from the code tree for some particular bit beeing 1 or 0 corresponds to the sum of probabilities over all the codewords where this bit is 1 or 0.
- The structured way to calculated the marginal probability can be done on the trellis.

Example of calculation of marginal probablities

We would like to calculate the marginal probability for $c_1 = 0$.

$p(c_1, c_2, c_3)$
$p(0,0,0)$
$p(0,0,1)$
$p(0,1,0)$
$p(0,1,1)$
$p(1,0,0)$
$p(1,0,1)$
$p(1,1,0)$
$p(1,1,1)$

$$p^{post}(c_2 = 0) = \frac{\sum\limits_{c_2=0} p(c_1,c_2,c_3)}{\sum\limits_{c_2=0} p(c_1,c_2,c_3) + \sum\limits_{c_2=1} p(c_1,c_2,c_3)}$$

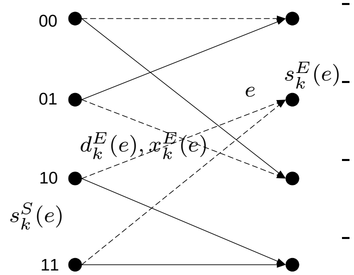
For the independent samples we can separate

$$p^{post}(c_2 = 0) = \frac{\sum\limits_{c_2=0} p(c_1|c_2)p(c_2)p(c_3|c_1,c_2)}{\sum\limits_{c_2=0} p(c_1|c_2)p(c_2)p(c_3|c_1,c_2) + \sum\limits_{c_2=1} p(c_1|c_2)p(c_2)p(c_3|c_1,c_2)}$$

The likelihood ratio becomes

$$\begin{aligned}\frac{p^{post}(c_2=0)}{p^{post}(c_2=1)} &= \frac{\sum\limits_{c_2=0} p(c_1|c_2)p(c_2)p(c_3|c_1,c_2))}{\sum\limits_{c_2=1} p(c_1|c_2)p(c_2)p(c_3|c_1,c_2))} \\ &= \frac{p(c_2=0)}{p(c_2=1)} \cdot \frac{\sum\limits_{c_2=0} p(c_1|c_2)p(c_3|c_1,c_2))}{\sum\limits_{c_2=1} p(c_1|c_2)p(c_3|c_1,c_2))}\end{aligned}$$

Notation



- e edge.
- $s_k^s(e)$ starting stage of the edge e.
- $s_k^s(e)$ ending stage of the edge e.
- $d_k(e)$ the information word containing k_0 bits.
- u_i stands for individual information bits.
- $x_k(e)$ codeword containing n_0 bits.

In this notation $s_k^S(e) = s_k^E(e)$
We assume here that the received signal is $y_k = x_k + n$ (transmitted symbols + noise).

- The metric at time k is

$$\begin{aligned}M_k(e) &= p\left(s_k^E(e), y_k | x_k^S(e)\right) \\ &= \sum_{x_k} p\left(s_k^E(e) | s_k^S(e)\right) p\left(x_k | s_k^S(e)\right) p\left(y_k|x_k\right)\end{aligned}$$

$p\left(s_k^E(e) | s_k^S(e)\right)$ a-priori information of the information bit.
$p\left(x_k | s_k^S(e)\right)$ indicating the existence of connection between edges $s_k^E(e), s_k^S(e)$
$p\left(y_k|x_k\right)$ probability of receiving y_k if x_k was transmitted

- Let $A_k(\cdot)$ and $B_k(\cdot)$ be forward and backward path metrics.

$$A_k(s) = p\left(s_k^E(e) = s, y_1^k\right)$$
$$= \sum_{e:s_k^E(e)=s} A_{k-1}\left(s_k^s(e)\right) M_k(e),\ k = 1, \ldots, N-1$$
$$B_k(s) = p\left(y_1^k | s_{k+1}^S(e) = s,\right)$$
$$= \sum_{e:s_k^E(e)=s} B_{k+1}\left(s_{k+1}^S(e)\right) M_{k+1}(e),\ k = N-1, \ldots, 1$$

- Suppose the decoder starts and ends with known states.

$$A_0(s) = \begin{cases} 1, & s = S_0 \\ 0, & \text{otherwise} \end{cases}$$
$$B_N(s) = \begin{cases} 1, & s = S_N \\ 0, & \text{otherwise} \end{cases}$$

- If the final state of the trellis is unknown

$$B_N(s) = \frac{1}{2^m},\ \forall s$$

- The joint probability at time k is

$$\sigma_k(e) = p\left(e, y_1^N\right)$$
$$= A_{k-1}\left(s_k^S(e)\right) \cdot M_k(e) \cdot B_k\left(s_k^E(e)\right)$$

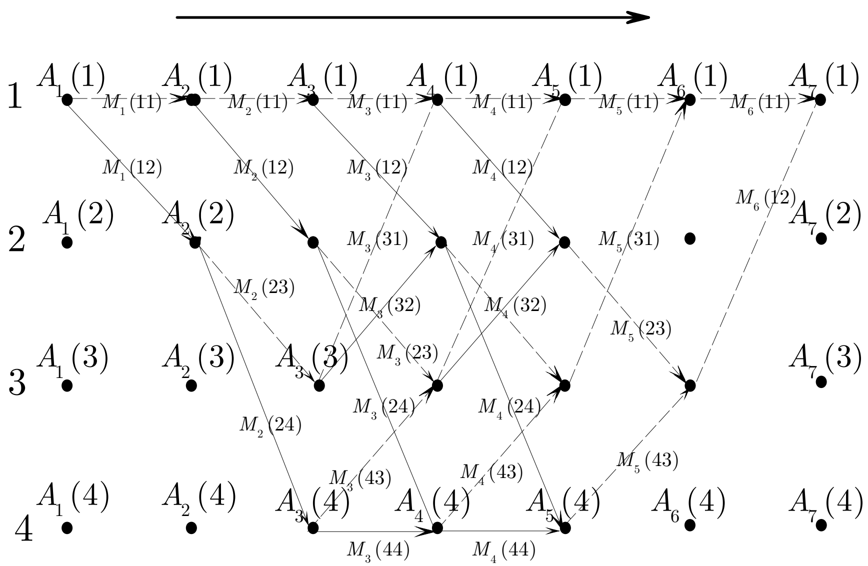


Figure 2: Forward calculation of $A(\cdot)$.

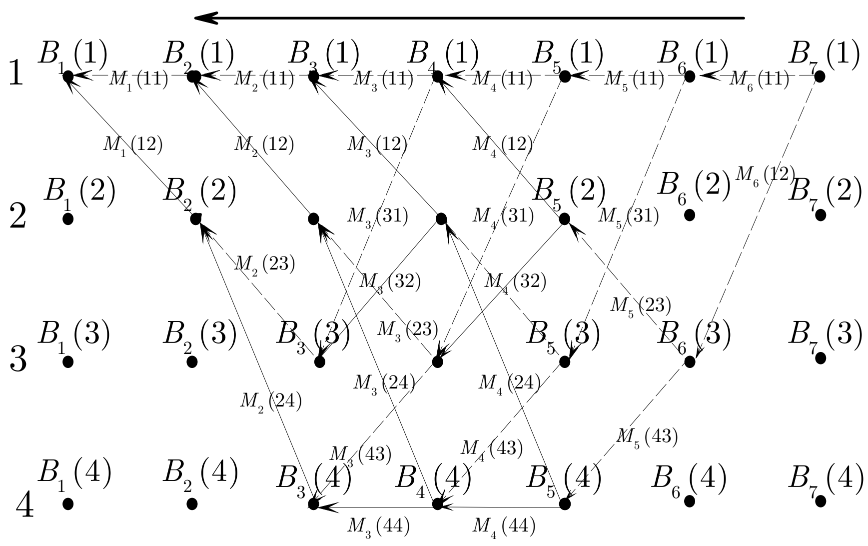


Figure 3: Backward calculation of $B(\cdot)$.

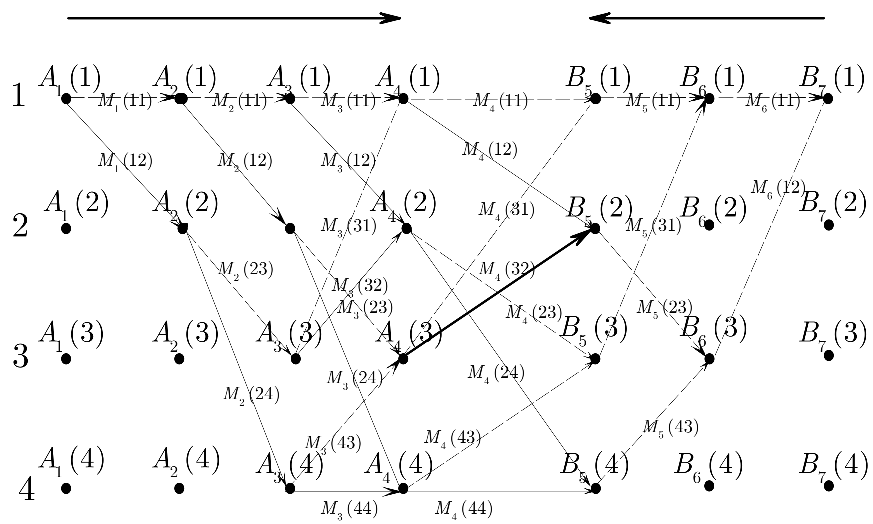


Figure 4: Caculation of the joint probability.

The a-posteriori probability can be expressed as:

$$
\begin{aligned}
p_k^A(u) &= p\left(u_k = u | Y_1^N\right) \\
&= \frac{1}{p\left(Y_1^N\right)} \sum_{e:u(e)=u} \sigma_k(e) \\
&= \frac{1}{p\left(Y_1^N\right)} \sum_{e:u(e)=u} A_{k-1}\left(s_k^S(e)\right) \cdot M_k(e) \cdot B_k\left(s_k^E(e)\right)
\end{aligned}
$$

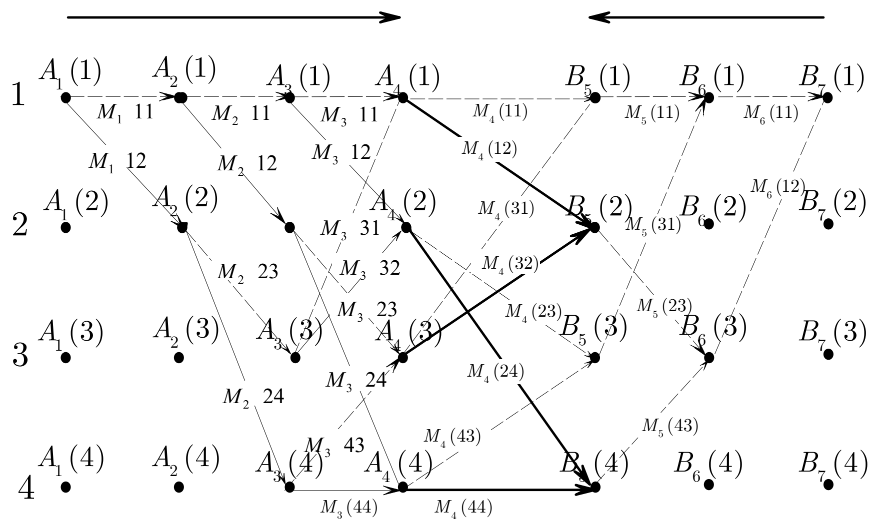


Figure 5: Caculation of the a-posteriori probability (APP).

Applying MAP algorithm to turbo codes

Encoder generates multiple encoded bit streams. These streams can be generated on different interleaved bit sequences.

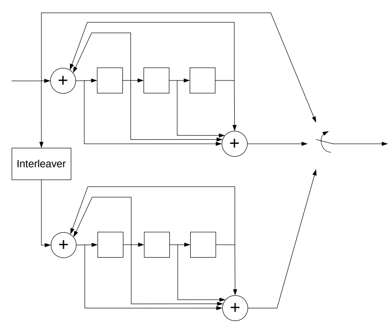


Figure 6: Example of a parallel concated convolutional codes

Iterative turbo decoding principle (PCCC)

The Transition probability in the trellis can be expressed by the bit probability from the other decoder and by the observed probability at the channel.

$$M_k\left(e\right) = C \cdot e^{(u_k L(u_k)/2} \cdot \exp\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{yl}\right)$$

One of the received bits corresponds to the systematic bits that is common for both coders.

$$\begin{aligned} \mathrm{M}_k\left(e\right) &= \mathrm{C}\cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\tfrac{L_c}{2} y_{kd} u_k\right) \cdot \exp\left(\tfrac{L_c}{2}\sum_{\substack{l=1\\ l\neq 1}}^{n} y_{kl}x_{yl}\right) \\ &= \mathrm{C}\cdot e^{(u_k L(u_k)/2)} \cdot \exp\left(\tfrac{L_c}{2} y_{kd} u_k\right) \cdot L_{e1} \end{aligned}$$

Where d is the index of the systematic bit in the codeword section k.
We assume that each section contains only one systematic bit.

The loglikelihood ratio of the estimated bit is the division of the probablity that $u_k = 1$ to probability that the bit was $u_k = 0$.

$$\begin{aligned} L\left(u_k|Y_1^N\right) &= ln\frac{p\left(u_k=0|Y_1^N\right)}{p\left(u_k=1|Y_1^N\right)} \\ &= \frac{\sum\limits_{e:u(e)=0} \sigma_k(e)}{\sum\limits_{e:u(e)=1} \sigma_k(e)} \\ &= ln\frac{\sum\limits_{e:u(e)=0} A_{k-1}\left(s_k^S(e)\right)\cdot M_k(e)\cdot B_k\left(s_k^E(e)\right)}{\sum\limits_{e:u(e)=1} A_{k-1}\left(s_k^S(e)\right)\cdot M_k(e)\cdot B_k\left(s_k^E(e)\right)} \\ &= ln\frac{\sum\limits_{\substack{e:u(e)=1\\ (mapped\ to\ -1)}} A_{k-1}\left(s_k^S(e)\right)\cdot e^{L(u_k)/2}\cdot e^{L_c y_{ks}/2}\cdot e^{\left(\frac{L_c}{2}y_{kd}u_k\right)}\cdot B_k\left(s_k^E(e)\right)}{\sum\limits_{\substack{e:u(e)=1\\ (mapped\ to\ -1)}} A_{k-1}\left(s_k^S(e)\right)\cdot e^{-L(u_k)/2}\cdot e^{-L_c y_{ks}/2}\cdot e^{\left(\frac{L_c}{2}y_{kd}u_k\right)}\cdot B_k\left(s_k^E(e)\right)} \\ &= L(u_k) + L_c y_{ks} + L_{e1}(u_k) \end{aligned}$$

$L(u_k)$ a-priori information.
$L_c y_{ks}$ loglikelihood of the systematic bit.
$L_{e1}(u_k)$ extrinsic information calculated for given decoder it describe information derived by imposing constraints of given code.
$L\left(u_k|Y_1^N\right)$ describes a posteriori information at the output of given decoder.

To the other decoder (for example to decoder 2) is given only extrinsic information (from decoder 1). That describes information derived from the other component code (code 1) and not othervise available to the next decoder (decoder 2).

Parallel concated convolutional codes (PCCC)

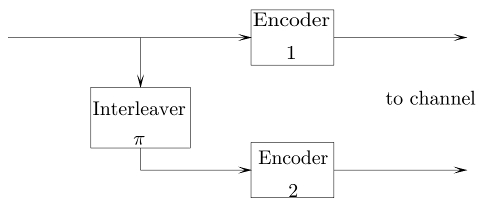


Figure 7: Encoder

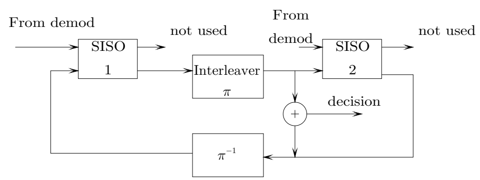


Figure 8: Decoder

Serial concated convolutional codes (SCCC)

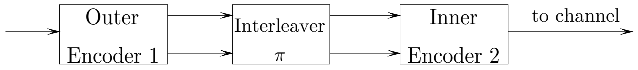


Figure 9: Encoder

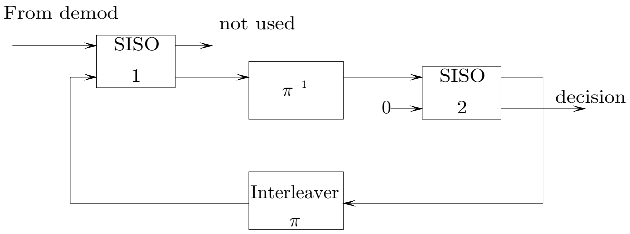


Figure 10: Decoder

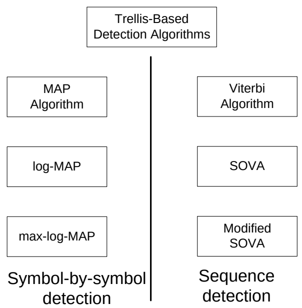
Parallel concated convolutional codes (PCCC)

- Encoder contains by two or more systematic convolutional encoders
- The constituent encoders code the same data stream that for different encoders are interleaved
- The systematic bits are transmitted only once
- In reciever the extrinsic information is calculated for the information bit and feed to the other code decoder

Serial concated convolutional codes (SCCC)

- Code is formed by concatenating two encoders
  - The output coded bit stream from the outer encoder is interleaved and feed to the inner encoder
- The decoder
  - Calculates the loglikelihoods of information bits at the output of the inner decoder and deinterleaver them
  - The outer decode is decoded and loglikelihoods for the coded bits are calculated
  - The coded bits loglikelihoods are interleaved and feed back to the inner decoder
  - The decision are made after the decoding iterations on the loglikelihoods of the information bits at the output of the outer decoder

Algorithms for Iterative (Turbo) Data Processing



Requirements

Accept soft-inputs in the form of a priori probabilities or log-likelihood ratios
Produce APP for output data
Soft-Input Soft-Output

- MAP: Maximum A Posteriori (symbol-by-symbol)
- SOVA: Soft Output Viterbi Algorithm

Modification of MAP algorithm

- MAP algorithm operates in probability domain.

$$p_k^A(u) = \frac{1}{p(Y_1^N)} \sum_{e:u(e)=u} A_{k-1}\left(s_k^S(e)\right) \cdot M_k(e) \cdot B_k\left(s_k^E(e)\right)$$

- When probablity is expressed by loglikelihood value we have to deal with numbers in very large range. (overflows in computers).

Simplification Log-MAP algorithm description

- Log-MAP algorithm is a transformation of MAP into logarithmic deomain.

- The MAP algorithm logarithmic domain is expressed with replaced computations
  - Multiplication is converted to addition.
  - Addition is converted to a $\max*(\cdot)$ operation.

  $max*(x,y) = \log(e^x + e^y) = \max(x,y) + \log\left(1 + e^{-|x-y|}\right)$
- The terms for calculating the probabilities in trellis are converted

$$\alpha_k(s) = \log A_k(s)$$
$$\beta_k(s) = \log B_k(s)$$
$$\gamma_k(s) = \log M_k(e)$$

- The complete logMAP algortihm is

$$\lambda_k^A(u) = \log \sum_{e:u(e)=1} A_{k-1}\left(s_k^S(e)\right) \cdot M_k(e) \cdot B_k\left(s_k^E(e)\right)$$
$$- \log \sum_{e:u(e)=0} A_{k-1}\left(s_k^S(e)\right) \cdot M_k(e) \cdot B_k\left(s_k^E(e)\right)$$
$$= \max_{e:u(e)=1}* \left(\alpha_{k-1}\left(s_k^S(e)\right) + \gamma_k(e) + \beta_k\left(s_k^E(e)\right)\right)$$
$$- \max_{e:u(e)=0}* \left(\alpha_{k-1}\left(s_k^S(e)\right) + \gamma_k(e) + \beta_k\left(s_k^E(e)\right)\right)$$
$$\alpha_k(s) = \log \sum_{e:s_k^E(e)=s} A_k\left(s_k^S(e)\right) \cdot M_k(e)$$
$$\beta_k(s) = \log \sum_{e:s_{k+1}^S(e)=s} B_{k+1}\left(s_{k+1}^S(e)\right) \cdot M_{k+1}(e)$$

Max-Log-MAP decoding Algorithm

- In summation of probabilities in Log-MAP algorithm we are using $max*(\cdot)$ operation.
- The $max*(\cdot)$ requires to convert LLR value into exponential and after adding 1 to move back into log domain.
- Simplifications
  - We can replace $\log\left(1 + e^{-|x-y|}\right)$ by a lookup table.
  - We can skip the term Max-Log-Map.

$$\alpha_n(s) = \log(A_n) = \log(\textstyle\sum_{s_k}[A_{n-1}(s_k^S)M_n(e)]$$

$$= \log(\textstyle\sum_{s_k}\exp[\alpha_{n-1}(s_k) + \gamma_n(s_k, s_k')])$$

$$= max * (\textstyle\sum_{s_k}\exp[\alpha_{n-1}(s_k) + \gamma_n(s_k, s_k')])$$

$$\approx \max(\alpha_{n-1}(s_k) + \gamma_n(s_k, s_k')) \qquad \Rightarrow \text{ Max-Log-MAP}$$

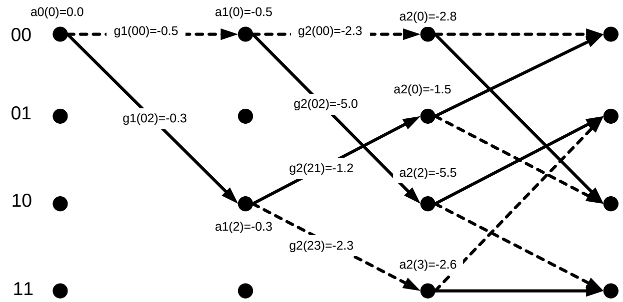
$$\beta_{n-1}(s) = \log(B_n) = \log(\textstyle\sum_{s_k}[B_n(s_k^S)M_n(e)]$$

$$= \log(\textstyle\sum_{s_k}\exp[\beta_n(s_k) + \gamma_n(s_k, s_k')])$$

$$= max * (\textstyle\sum_{s_k}\exp[\beta_n(s_k) + \gamma_n(s_k, s_k')])$$

$$\approx \max(\beta_n(s_k) + \gamma_n(s_k, s_k')) \qquad \Rightarrow \text{ Max-Log-MAP}$$

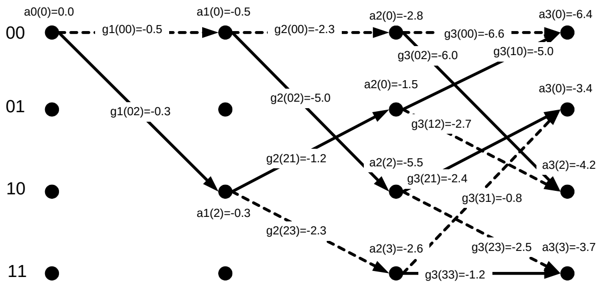
Log-MAP Algorithm (Forward Recursion 1)



$$\alpha_{n+1}(s_k') = \log(\textstyle\sum_{s_k}\exp[\alpha_n(s_k) + \gamma_{n+1}(s_k, s_k')])$$

$$\log(\exp[x] + \exp[y]) \approx \max(x, y) + \underbrace{\log(1 + \exp[-|x-y|])}_{tab\Delta} = \max{}^*(x, y)$$

$$\alpha_1(s_1) = \log(\exp[\alpha_0(s_0) + \gamma_1(s_0, s_1)]) \Rightarrow \log(\exp[0.0 + (-0.3)]) = -0.3$$

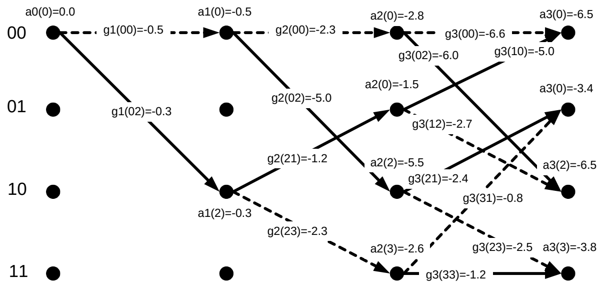


$$\alpha_{n+1}(s_k') = \log(\textstyle\sum_{s_k}\exp[\alpha_n(s_k) + \gamma_{n+1}(s_k, s_k')])$$

$$\alpha_3(s_2) = \log\left(e^{(\alpha_2(s_0)+\gamma_3(s_0,s_2))} + e^{(\alpha_2(s_1)+\gamma_3(s_1,s_2))}\right)$$

$$\Rightarrow \ln\left(e^{(-2.8-6.0)} + e^{(-1.5-2.7)}\right) \approx \max *(-8.8, -6.5) = -6.4$$

Max-Log-MAP approximation



$$\alpha_{n+1}(s_k') = \log(\textstyle\sum_{s_k}\exp[\alpha_n(s_k) + \gamma_{n+1}(s_k, s_k')])$$

$$\alpha_3(s_2) = \log\left(e^{(\alpha_2(s_0)+\gamma_3(s_0,s_2))} + e^{(\alpha_2(s_1)+\gamma_3(s_1,s_2))}\right)$$

$$\approx \max\left(e^{(-2.8-6.0)} + e^{(-1.5-2.7)}\right) = \max(-8.8, -6.5) = -6.5$$

Soft-output Viterbi algorithm

- Two modifications compared to the classical Viterbi algorithm
  - The path metric is modified to acoount the extrinsic information. This is similar to the metric calculation in Max-Log-MAP algorithm.
  - The algorithm is modified to calculate the soft bit.

Figure 5.16 - 5.17 from the book.

SOVA

- For each state in the trellis the metric $M\left(\underline{s}_k^s\right)$ is calculated for both merging paths.
- The path with the highest metric is selected to be the survivor.
- For the state (at this stage) a pointer to the previous state along the surviving path is stored.
- The information to give $L\left(u_k|\,\underline{y}\right)$ is stored.
  - The difference Δ_k^s between the discarded and surviving path.
  - The binary vector containing $\delta+1$ bits, indicating last $\delta+1$ bits that generated the discarded path.
- After ML path is found the update sequences and metric differences are used to calculate $L\left(u_k|\,\underline{y}\right)$.

Calculation of $L\left(u_k|\,\underline{y}\right)$.

- For each bit u_k^{ML} in the ML path we try to find the path merging with ML path that had compared to the u_k^{ML} in ML different bit value u_k at state k and this path should have minimal distance with ML path.
- We go trough $\delta+1$ merging paths that follow stage k i.e. the $\Delta_i^{s_i}$ $i=k...(k+\delta)$
- For each merging path in that set we calculate back to find out which value of the bit u_k generated this path.
- If the bit u_k in this path is not u_k^{ML} and $\Delta_i^{s_i}$ is less than current Δ_k^{min} we set $\Delta_k^{min}=\Delta_i^{s_i}$
- $L\left(u_k|\,\underline{y}\right)\approx u_k \min\limits_{\substack{i=k\ldots k+\sigma \\ u_k^{ML}\neq u_k^i}} \Delta_i^{s_i}$

Comparison of the computing principles

MAP

- The MAP algorithm is the optimal component decoder algorithm.
- It finds the probability of each bit u_k of either being a +1 or -1 by summing the probabilities of all the codewords where the given bit is +1 and where the bit is -1.
- Extremely complex.
- Because of the exponent is probability calculations in pracitce the MAP algrithm often suffers of numerical problems.

LogMAP

- LogMAP theorectically identical to MAP the calculation only are made in logarithmic domain.
- Multiplications are replaced with addition and summation with $\max*(\cdot)$ operation.
- Numerical problems that occure in MAP are cirmcumvented.

Max-Log-MAP

- Similar to LogMAP but replaces the maxlog operation ($\max*(\cdot)$) with taking maximum.
- Because at each state in forward and backward calcualtions only the path with maximum value is considered the probabilities are not calculated over all the codewords.
  - In recursive calculation of α and β also only the best transition is considered.
  - The algorithm gives the logarithm of the probability that only the most likely path reaches the state.
- In the MaxLogMAP $L\left(u_k|\underline{y}\right)$ is comparison of probability of most likely path giving $u_k = -1$ to the most likely path giving $u_k = +1$.
  - In calcualtions of loglikelihood ratio only two codewords are considered (two transitions): The best transition that would give +1 and the best transition that would give -1.
- MaxLogMAP performs worse than MAP or LogMAP

SOVA

- SOVA the ML path is found.
  - The recursion used is identical to the one used for calcuating of α in MaxLogMAP algorithm.
- Along the ML path hard decision on the bit u_k is made.
- $L\left(u_k|\underline{y}\right)$ is the minimum metric difference between the ML path and the path that merges with ML path and is generated with different bit value u_k.
  - In $L\left(u_k|\underline{y}\right)$ calcualtions accordingly to MaxLogMAP one path is ML path and other is the most likely path that gives the different u_k.
  - In SOVA the difference is calculated between the ML and the most likely path that merges with ML path and gives different u_k.
    This path but the other may not be the most likely one for giving different u_k.

- The output of SOVA just more noisy compared to MaxlogMAP output (SOVA does not have bias).
- The SOVA and MaxLogMAP have the same output
  - The magnitude of the soft decisions of SOVA will either be identical of higher than those of MaxLogMAP.
  - If the most likely path that gives the different hard decision for u_k has survived and merges with ML path the two algorithms are identical.
  - If that path does not survive the path on what different u_k is made is less likely than the path which should have been used.

Table 1: Comparison of complexity of different decoding algorithms

Operations	maxlogMAP	logMAP	Sova
max-ops	$5 \times 2^M - 2$	$5 \times 2^M - 2$	$3\,(M+1) + 2^M$
additions	$10 \times 2^M + 11$	$10 \times 2^M + 11$	$2 \times 2^M + 8$
mult. by ± 1	8	8	8
bit comps			$6\,(M+1)$
look-ups		$5 \times 2^M - 2$	

M is the length of the code memory.

Table accordingly to reference [1]

If to assume that each operation is comparable we can calculate the totat amount of operations per bit every algorithm demands for decoding one code in one iteration.

Table 2: Number of reguired operations per bit for different decoding algorithms

memory (M)	MaxLogMAP	LogMAP	Sova
2	77	95	55
3	137	175	76
4	257	335	109
5	497	655	166

Performance of the Turbo Decoding algorithms

- Component decoding algorithms.
- Number of iterations used.
- Frame length impact to performance.
- Interleavers.
- Channel reliability values

Figures from the book pages 150 - 171.

References

1 P. Robertson, E. Villebrun, P. Hoeher, "Comparison of Optimal and Suboptimal MAP decoding algorithms", ICC, 1995 page 1009-1013.