Communications and Network, 2009, 20-24
doi:10.4236/cn.2009.11003 Published Online August 2009 (http://www.scirp.org/journal/cn)

Comparison and Design of Decoder in B3G Mobile Communication System

Mingxiang GUAN[1], Mingchuan YANG[2]
[1]*Department of Electronic Communication Technology, Shenzhen Institute of Information Technology, Shenzhen, China*
[2]*School of Electronic and Information Technology, Harbin Institute of Technology, Harbin, China*
Email: [1]*gmx2020@126.com*; [2]*yangmingchuan@hit.edu.cn*

Abstract: Turbo code has been shown to have ability to achieve performance that is close to Shannon limit. It has been adopted by various commercial communication systems. Both universal mobile telecommunications system (UMTS) TDD and FDD have also employed turbo code as the error correction coding scheme. It outperforms convolutional code in large block size, but because of its time delay, it is often only used in the non-real-time service. In this paper, we discuss the encoder and decoder structure of turbo code in B3G mobile communication System. In addition, various decoding techniques, such as the Log-MAP, Max-log-MAP and SOVA algorithm for non-real-time service are deduced and compared. The performance results of decoder and algorithms in different configurations are also shown.

Keywords: decoder, beyond 3G mobile communication system, decoding algorithm

1. Introduction

A turbo code can be thought as a refinement of the concatenated encoding structure and an iterative algorithm for decoding the associated code sequence [1]. The codes are constructed by applying two or more component codes to different interleaved versions of the same information sequence [2][3]. In literature [4], for any single traditional code, the final step at the decoder yields hard-decision decoded bits (or, more generally, decoded symbols). In order for a concatenated scheme such as a turbo code to work properly, the decoding algorithm should not limit itself to pass hard decisions among the decoders [5]. To best exploit the information learned from each decoder, the decoding algorithm must effect an exchange of soft rather than hard decisions [6]. For a system with two component codes, the concept behind turbo decoding is to pass soft decisions from the output of one decoder to the input of the other, and to iterate this process several times to produce better decisions [7].

2. Principle of Iterative Decoding

For the first decoding iteration of the soft input/soft output decoder in Figure 1, one generally assumes the binary data to be equally likely, yielding an initial a priori LLR value of L(d) =0 for the third term in [8]. The channel pre-detection LLR value, Lc(x) , is measured by forming the logarithm of the ratio of λ_1 and λ_2, seen in Figure l. The output $L(\hat{d})$ of the Figure 2 decoder is made up of the LLR from the detector, $L'(\hat{d})$, and the extrinsic LLR output. $L_e(\hat{d})$, representing knowledge gleaned from the decoding process. As illustrated in Figure 2, for iterative decoding the extrinsic likelihood is fed back to the decoder input, to serve as a refinement of the a priori value for the next iteration.

Consider the two-dimensional code (product code) depicted in Figure 2. The configuration can be described as

This paper supported by the 3rd natural science foundation of institute (No. LG-08010).

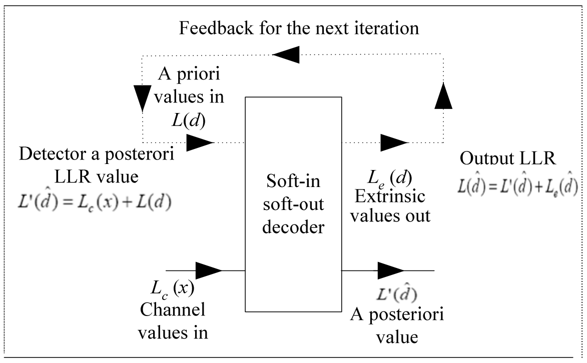
Feedback for the next iteration
A priori values in $L(d)$
Detector a posterori LLR value
$L'(\hat{d}) = L_c(x) + L(d)$
Soft-in soft-out decoder
$L_e(\hat{d})$ Extrinsic values out
Output LLR
$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d})$
$L_c(x)$ Channel values in
$L'(\hat{d})$ A posteriori value

Figure 1. Soft input/soft output decoder

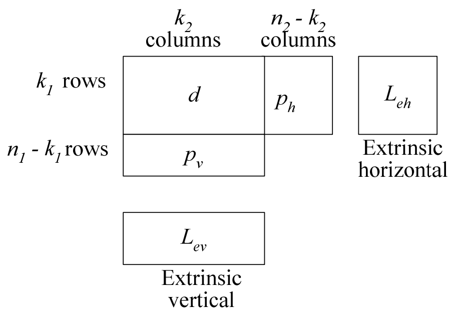

Figure 2. Two-dimensional product code

a data array made up of k_1 rows and k_2 columns. Each of the k_1 rows contains a code vector made up of k_2 data bits and n_2-k_2 parity bits. Similarly, each of the k_2 columns contains a code vector made up of k_1 data bits and n_1-k_1 parity bits. The various portions of the structure are labeled *d* for data, p_h for horizontal parity (along the rows), and p_v for vertical parity (along the columns). Additionally, there are blocks labeled L_{eh} and L_{ev} , which house the extrinsic LLR values learned from the horizontal and vertical decoding steps, respectively. Notice that this product code is a simple example of a concatenated code. Its structure encompasses two separate encoding steps, horizontal and vertical. The iterative decoding algorithm for this product code proceeds as follows:

1) Set the a priori information

$$L(d)=0 \tag{1}$$

2) Decode horizontally, obtain the horizontal extrinsic information as shown below:

$$L_{eh}(\hat{d}) = L(\hat{d}) - L_c(x) - L(d) \tag{2}$$

3) Set

$$L(d) = L_{eh}(\hat{d}) \tag{3}$$

4) Decode vertically, obtain the vertical extrinsic information as shown below:

$$L_{ev}(\hat{d}) = L(\hat{d}) - L_c(x) - L(d) \tag{4}$$

5) Set

$$L(d) = L_{ev}(\hat{d}) \tag{5}$$

6) If there have been enough iterations to yield a reliable 7 decision, go to step 7; otherwise, go to step 2;

7) The soft output is:

$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d}) \tag{6}$$

3. B3G Mobile System Decoder Design

The algorithms for decoders can be divided into two categories: (1) Log-MAP: Log-Maximum A Posteriori, (2) SOVA: Soft Output Viterbi Algorithm.

3.1. Log-MAP Algorithm

Before explaining the MAP decoding algorithm, we assume some notations:

$s_k^S(e)$ starting stage of the edge e.

$s_k^E(e)$ 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.

We assume here that the received signal is $y_k=x_k+n$ (transmitted symbols + noise).

The metric at time *k* is

$$\begin{aligned} M_k(e) &= p(s_k^E(e), y_k \mid x_k^E(e)) \\ &= \sum_{x_k} p(s_k^E(e) \,|s_k^S(e)) p(x_k \mid s_k^S(e)) p(y_k \mid x_k) \end{aligned} \tag{7}$$

$p(s_k^E(e) \mid s_k^S(e))$ a priori information of the information bit.

$p(x_k \mid s_k^S(e))$ indicating the existence of connection between edges $s_k^S(e)$, $s_k^E(e)$

$p(y_k \mid x_k)$ probability of receiving y_k if x_k was transmitted.

$A_k(.)$ and $B_k(.)$ is forward and backward path metrics.

$$\begin{aligned} A_k(s) &= p(s_k^E(e) = s, y_1^k) \\ &= \sum_{e:s_k^E(e)=s} A_{k-1}(s_k^S(e)) M_k(e), k = 1,...,N-1 \end{aligned} \tag{8}$$

$$\begin{aligned} B_k(s) &= p(y_1^k \mid s_{k+1}^S(e) = s) \\ &= \sum_{e:s_k^E(e)=s} B_{k+1}(s_{k+1}^S(e)) M_{k+1}(e), k = N,...,1 \end{aligned} \tag{9}$$

Suppose the decoder starts and ends with known states

$$A_0(s) = \begin{cases} 1, s = S_0 \\ 0, otherwise \end{cases}, \; B_N(s) = \begin{cases} 1, s = S_N \\ 0, otherwise \end{cases} \tag{10}$$

If the final state of the trellis is unknown:

$$B_N(s) = \frac{1}{2^m}, \forall s \tag{11}$$

The joint probability at time k is:

$$\sigma_k(e) = p(e, y_1^N) = A_{k+1}(s_k^S(e)).M_k(e).B_k(s_k^E(e)) \tag{12}$$

3.2. Output Viterbi Algorithm (SOVA)

There are two modifications compared to the classical Viterbi algorithm. One is the path metric modified to account the extrinsic information. This is similar to the metric calculation in Log-MAP algorithm. The other is the algorithm modified to calculate the soft bit. For each state in the trellis the metric $M(\underline{s}_k^S)$ is calculated for both merging paths, the path with the highest metric is selected to be the survivor, and for the state (at this stage) a pointer to the previous state along the surviving path is stored. The information to give $L(u_k|\underline{y})$ is stored, the difference $\Delta_k^{s_i}$ between the discarded and surviving path. The binary vector containingδ+1 bits, indicating last δ+1 bits that generated the discarded path. After ML path is found the update sequences and metric differences are used to calculate $L(u_k|\underline{y})$. 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 δ+1 merging paths that follow stage k i.e. the $\Delta_k^{s_i}$ $i = k,\ldots, (k+\delta)$, 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 $\Delta_k^{s_i}$ is less than current $\Delta_k^{\min}$, we set $\Delta_k^{\min} = \Delta_k^{s_i}$.

$$L(u_k \mid \underline{y}) \approx u_k \min_{\substack{i=k\ldots k+\sigma \\ u_k^{ML} \neq u_k^i}} \Delta_i^{S_i} \quad \textbf{(13)}$$

4. Comparison of the Decoding Algorithms

SOVA the ML path is found. The recursion used is identical to the one used for calculating of α in Log-MAP algorithm. Along the ML path hard decision on the bit u_k is made. $L(u_k|\underline{y})$ 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 $L(u_k|\underline{y})$ calculations accordingly to Log-MAP 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. Compared to Log-MAP output (SOVA does not have bias), the output of SOVA is just much noisier. The SOVA and Log-MAP have the same output. The magnitude of the soft decisions of SOVA will either be identical of higher than those of Log-MAP. 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.

The forward recursion in Log-MAP and SOVA is identical but the trace-back depth in SOVA is either less than or equal to the backward recursion depth. Log-MAP is the slowest of the three algorithms, but has the best performance among these three algorithms. In our TD-CDMA simulation, we implemented the Log-MAP and SOVA decoder to get best performance (with Log-MAP) or fastest speed (with SOVA). Figure 3 gives the performance comparison between Log-MAP and SOVA and it shows that Log-MAP is 1.2dB better than SOVA at the BER of 10^{-2}, with the code block size of 260 bits and 7 iterations.

The code block size or interleaver size is also affect the decoding performance, the BER performance comparison of various code block size is shown in Figure 4. The longer is the code block, the better is the performance, but the big code block size makes more computing time, and the computing complexity increases by exponential.

Table 1. Comparison of complexity of different decoding algorithms

Operations	MAP	Log-MAP	SOVA
additions	$4\times2^M+6$	$12\times2^M+6$	$4\times2^M+9$
max-ops		$4\times2^M-2$	$2\times2^M-1$
multiplications	$10\times2^M+8$	8	4
look-ups	4(exp)	$4\times2^M-2$	

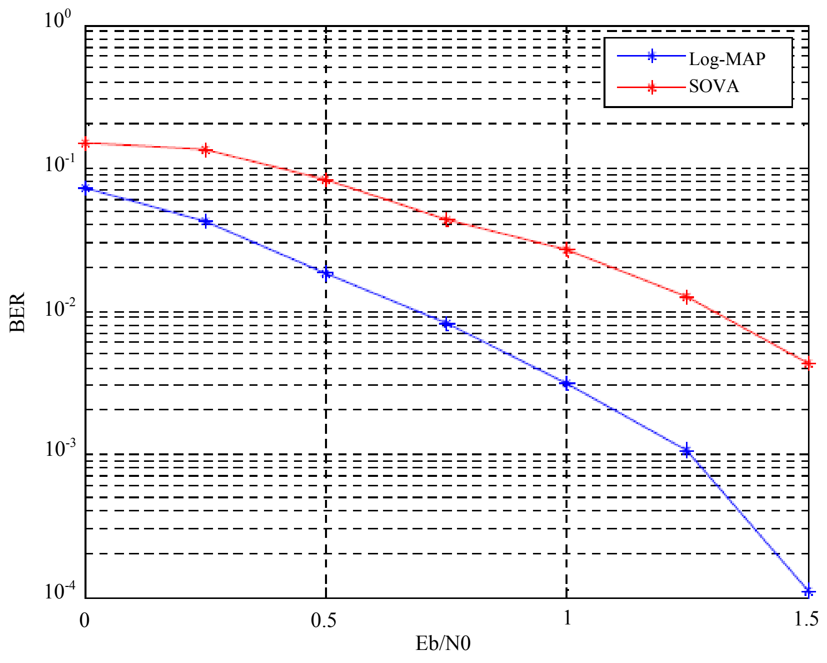

Figure 3. BER performance of Log-MAP algorithm VS SOVA algorithm

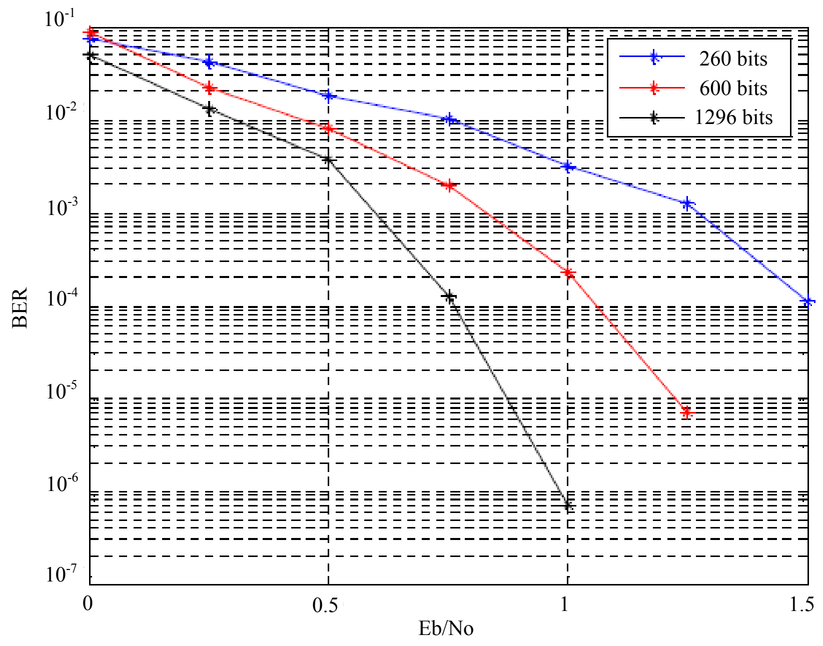

Figure 4. Measured BER performance of the Log-MAP algorithm for various block sizes

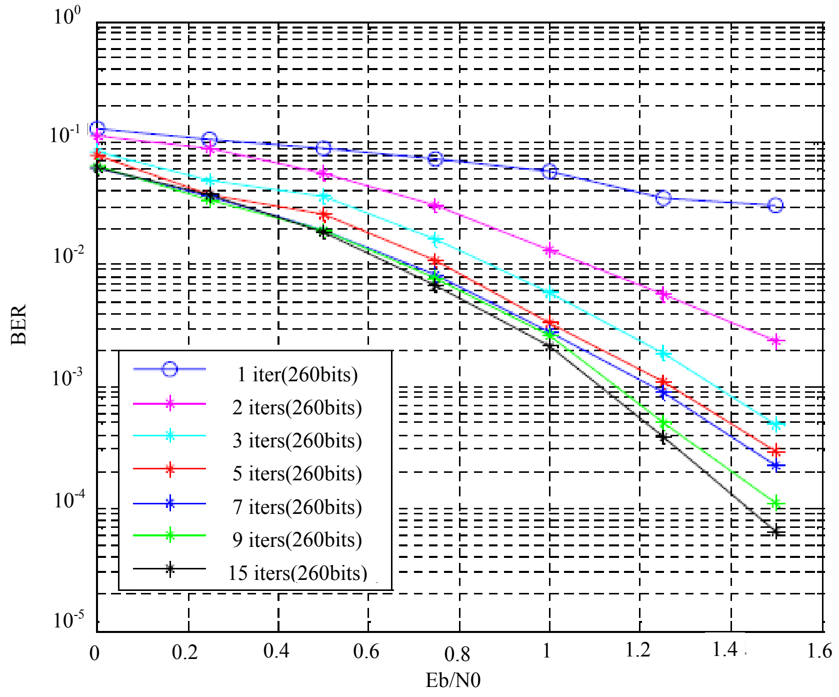

Figure 5. Measured BER performance of Log-MAP algorithm with increasing iteration count

Turbo decoder is the iterative decoder, and decoding performance is also impacted by the number of iteration. Figure 5 shows that the BER performance improves as the number of turbo iteration increased. However, the required computation also increases. It is shown that no significant performance improvement is observed after the sixth iteration.

5. Conclusions

We illustrate the turbo decoder principle, and the derivation of Log-MAP, and SOVA algorithms. Log-MAP algorithm is shown to achieve the best performance with good complexity tradeoff. SOVA algorithm has less computation complexity with about 1.2dB performance degradation compared with Log-MAP at the BER of 10^{-2}. We also demonstrate that the performance of turbo code is directly proportional to the interleaver size and number of iteration in the turbo decoder. Finally, it is also shown that the performance is affected by the scale of the input soft bits power but the effect is negligible when the scaling factor is larger than 0.8.

6. Acknowledgments

The author would like to thank Dr. Li Lu to help to design simulation platform, and Prof. Gu Xuemai for his revision of the text, and the editor and the anonymous reviewers for their contributions that enriched the final paper.

REFERENCES

[1] PIETROBON S S. Implementation and performance of a turbo/MAP decoder. Int. J. Satellite Communication, 1998, 16: 23-46.

[2] KAZA J, CHAKRABARTI C. Design and implementation of low-energy turbo decoders. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2004, 12(9): 968-977.

[3] HAGH M, SALEHI M, ET AL. Implementation issues in turbo decoding for 3GPP FDD receiver. Wireless Personal Communications, 2006, 39(2): 165-182.

[4] LEE D-S, PARK I.-C. Low-power log-MAP decoding based on reduced metric memory access. IEEE Transactions on Circuits and Systems, 2006, 53(6): 1244-1253.

[5] BOUTILLON E, GROSS W J, GULAK P G. VLSI architectures for the MAP algorithm. IEEE Transactions on Communications, 2003, 51(2): 175-185.

[6] ANANTHARAM V E Y. Iterative decoder architectures. IEEE Communications Magazine, 2003, 41(8): 132-140.

[7] KWON T-W, CHOI J-R. Implementation of a two-step SOVA decoder with a fixed scaling factor. IEICE Transactions on Communications, E86-B(6): Jun. 2003, 1893-1900.

[8] CHEN Y, PARHI K K. On the performance and implementation issues of interleaved single parity check turbo product codes. Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, 2005, 39(1): 35-47.