Second Order Difference Aided CRC Check Stopping Criterion for Turbo Decoding

Xuying Zhao[1(✉)], Xiaoqin Wang[2], and Donglin Wang[2]

[1] University of Chinese Academy of Sciences, Beijing, China
zhaoxuying2012@ia.ac.cn
[2] Institute of Automation, Chinese Academy of Sciences, Beijing, China

Abstract. This paper presents an enhanced CRC (Cyclic Redundancy Check) check stopping criterion to improve the throughput of turbo decoders. Turbo decoder is applied to the 3G and LTE systems. CRC check stopping criterion is normally used by the turbo decoder, and it reduces the iteration times dramatically when transmission blocks are received correctly. However, when the quality of the received signals is very low, the receiving data cannot be decoded correctly even though the turbo decoder keeps working all the time. In these scenarios, the CRC check stopping criterion has no effects any more. To reduce the number of iterations when the quality of the received signals is low, the second order difference aided CRC check stopping criterion is proposed. Based on the second order difference of soft-bit information and/or hard-bit information, some bad situations are identified before CRC check stopping condition is satisfied. Through simulations, it is proved that this method greatly reduces the decoding iteration times in bad transmission environments. On the other hand, the whole throughput of the turbo decoder is improved.

Keywords: CRC check · Second order difference · Turbo decoding

1 Introduction

Turbo codes, introduced by Claude Berrou in 1993 [1] have been widely researched and used because of good decoding performance. Typically, turbo decoding algorithms set a maximum iteration number in case of endless computing. These methods are called "Fixed-Iteration-Number" (FIN) stopping criterion. FIN has poor flexibility especially when transmission quality is good. In this situation, the decoder can correct all the errors by iterating one or two times. Combined with the FIN, many other algorithms are proposed to improve the decoding performance after some sufficient iterations.

In 1996 Joachim Hagenauer proposed the Cross-Entropy (CE) method to stop iteration before it reached the maximum iteration number [2]. CE algorithm can significantly reduce the iteration numbers with very little performance loss, but the computational complexity is relatively large. Based on CE method, some other stopping critetions were created to reduce the complexity such as SCR (Sign-Change-Ratio) algorithm and HDA (Hard-Decision-Aided) algorithm [3]. Sign-Difference-Ratio algorithm was relative to the SCR and it resolved the problem of big data storage [4]. Improved HDA algorithm (IHDA) reduced the storage memory without degrading the performance compared to the HDA [5]. Later on, based upon the CE method, a kind of method called Yu criterion was proposed and its decoding performance was almost near the optimal [6]. Several other early termination methods were proposed afterwards. For example, Jia Hou presented the adaptive SNR (Signal Noise Ratio) algorithm [7], Fan-Min Li proposed the MOR (Measurement of Reliability) method in 2005 [8] and Wei Jiang proposed the algorithm about the mutual information between the logarithm likelihood ratio and the data bits [9]. In recent years, some other new algorithms for early termination have also been raised [10, 11].

In contrast with the above methods, the error correcting performance of CRC check stopping criterion is good and corresponds to the Genie method. Nevertheless, CRC check algorithm has one obvious drawback. When SNR is low or some outburst errors happen to the decoder, CRC check cannot pass and the decoder will iterate up to the maximum number. From simulations we can see that under this case the output of the decoder is almost the same as that with fewer iteration times, and the time consumption caused by excess iterations is invalid. A new method is proposed to solve this problem in this paper. At each iteration, we compute the correlation values of the two sub-decoders and the correlation values are called the first order difference. Then we compute the difference between two consecutive iterations and the difference is called the second order difference. The decoder stops iteration based on that the second order difference is positive or not. This method can reduce the number of iterations with almost no damage to the BLER (Block Error Rate) performance, especially in low SNR.

The remainder of the paper is organized as follows. In Sect. 2, the second order difference aided CRC check stopping criterion is presented. Simulation results and system complexity are analyzed in Sect. 3. Finally, Sect. 4 concludes the paper.

2 Second Order Difference Aided CRC Check Stopping Criterion

CRC check stopping criterion is rather reliable and early stopping hardly degrades the performance. On the basis of considerable performance, we can further decrease the average iteration times. Figure 2 shows that when SNR is low, the CRC check iteration times are high, whereas the BLER performance is still "bad". Next we will focus on this problem.

In discrete function, the difference between two continuous adjacent values is called the first order difference. If we define $x(k)$, then $y(k) = x(k + 1) - x(k)$ is the first order difference, and this value indicates the monotonicity of the discrete function. The second order difference is defined as $z(k) = y(k + 1) - y(k)$, and this indicates the speed of the change rate.

For turbo codes, we have analyzed the regularity of the internal data stream variation in the decoder. By tracking amount of intermediate process results, we discover that the difference between two sub-decoders is getting lower and lower as the iteration goes on. One case is shown in Table 1, where we compute the correlation values of the LLRs and hard bits from the two sub-decoders.

Table 1. The regularity of the data stream changes as the iteration goes on

Iteration number	First order difference	Second order difference
0	−0.47484	
1	−0.791836	0.316997
2	−0.981215	0.189379
3	−1.200226	0.219011
4	−1.375428	0.175202
5	−1.681867	0.306439
6	−1.958921	0.277054
7	−2.659726	0.700806

The above table shows that the expected hard bits of the two sub-decoders are approaching to each other as the number of iterations increases. The values of second order difference are all positive, which indicates the "approaching" tendency. Normally, when the iteration times get bigger, the first order difference will be smaller and the decreasing rate is growing. That is, the second order difference keeps positive. Once the decoder works abnormally, the first order difference becomes larger and the second order difference turns to be negative. As is shown in Table 2, at the last iteration 4, the first order difference becomes larger than −1.674556 at iteration 3, indicating that the decoder meets some errors. Correspondingly, the second order difference changes into a negative value −0.062031.

Table 2. Abnormal work of the decoder, the decoded bits of the decoder are not the same with the original information bits

Iteration number	First order difference	Second order difference
0	−0.564011	
1	−0.9835	0.419489
2	−1.244495	0.260995
3	−1.674656	0.430161
4	−1.612625	−0.062031

Based on the above analysis, we propose the second order difference aided CRC check stopping criterion. When some accidental errors happen to the decoder, the errors will pass over iteration by iteration, and the extra iterations work uselessly or even worse. The second order difference aided CRC check algorithm is as follows,

(1) Set $i = 0$.
(2) At iteration i, compute the correlation values of the two sub-decoders and save them, where the correlation values are corresponding to the first order difference. At the same time, CRC check is processing. If CRC check is passed, stop the iteration; otherwise, go to step (3).
(3) At iteration $i + 1$, compute the correlation of the two sub-decoders, then calculate the second order difference along with the previous difference values. If the second order difference is positive, go on CRC checking and save the first order difference

values; otherwise, stop the iteration. If the CRC check is passed, stop the iteration; else set $i = i + 1$ and go to step (4).

(4) If i is less than the preset maximum iteration number, go to step (3); else stop the iteration.

3 Simulation Results and System Complexity Analysis

There are several schemes to compute the first order difference. The output LLRs of each sub-decoder can be used to calculate these values. To some extent, the performance of soft information is better than the hard bits. While the arithmetic units for floating points are complex and time-consuming, especially the multiplication arithmetic. On the other hand, as the CRC check is based on the hard bits, we can take this advantage and adopt the hard bits to compute the first order difference. However, completely hard decision will damage a part of performance. Considering the system complexity and decoding performance, we can have a trade-off. A multiplier for floating point and hard bit is corresponding to a multiplexer, which is less complex than the floating-point multiplier. For simplicity, we call the soft information and hard bits assisted CRC check "CRC-SHA", and the hard bits assisted CRC check "CRC-HARD". The CRC-SHA architecture is depicted in Fig. 1.

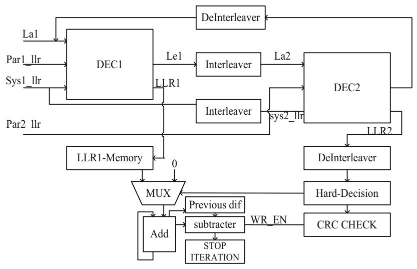

Fig. 1. CRC-SHA architecture

In this architecture, we need an extra memory to store the LLRs of the first sub-decoder. After the second sub-decoder outputs the LLRs and makes a hard decision, the computing of the first order difference begins. The process is like that the soft information is to be multiplexed and goes into an accumulator. After finishing the whole code block, the first order difference will be stored and do subtraction with the previous saved ones. Then we can get the second order difference, and this value will control the CRC check module and decide whether to stop the iteration.

There is a situation that the decoder works abnormally on occasion, but it can still repair itself to obtain the correct results. If we stopped the iteration early, the BLER performance might be a little "bad". This paper proposes a solution to reduce the decoder's accidental errors. Here a parameter "ErrFlag" is set to indicate the error detected numbers. We simulate the cases "$ErrFlag = 1$" and "$ErrFlag = 2$". The BLER performance and the average iteration times are described in Fig. 2.

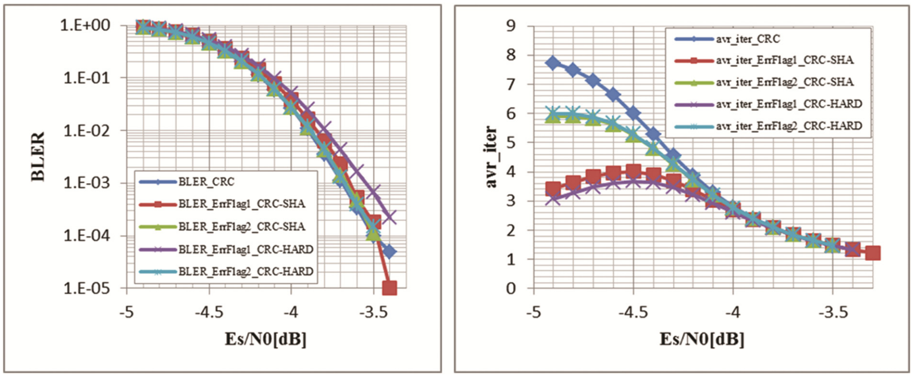

Fig. 2. The BLER and average iteration numbers of CRC-SHA and CRC-HARD

The above figures show that CRC check stopping criterion has the lowest BLER, while the average iteration times are highest. In the case *ErrFlag* = 2, the BLER performance of CRC-SHA is nearly to the CRC check, and the average iteration times are also decreased. Figure 2 shows that when SNR is in −4.9 to −4.4 dB, the average number of iterations falls in approximate two times. In the case *ErrFlag* = 1, the BLER performance of CRC-SHA has a loss of 0.1 dB, while the CRC-HARD loses nearly 0.2 dB. Both CRC-SHA and CRC-HARD can dramatically reduce the average iteration times. Whereas at a poor transmission environment, the CRC-HARD may degrade the decoding performance due to its weaker decoding ability. So considering the system complexity and error correcting performance, we take the CRC-SHA algorithm as the alternative solution.

4 Conclusion

This paper discusses a new method to overcome the CRC check drawbacks, primarily the number of iterations, which efficiently enhances the throughput of the turbo decoder. Almost all the current papers about the stopping criterion are based on the first order difference and a threshold value. We have put forward a new idea based on the second order difference. From the simulation results, we can conclude that this method can dramatically improve the decoding speed, reducing the unnecessary iterations. Actually, as the second order difference method includes the first order process, it can be used to stop the iteration alone. We can calculate a threshold about the second order difference. If the first order difference is in a certain interval and the second order difference is beyond the threshold, we can stop the iteration. Compared with the CRC-SHA, this method will reduce the power consumption and further research can be followed.

References

1. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: turbo codes. In: Proceedings IEEE International Conference on Communications, pp. 1064–1070 (1993)
2. Hagenauer, J., Offer, E., Papke, L.: Iterative decoding of binary block and convolutional codes. IEEE Trans. Inform. Theory **42**, 429–445 (1996)
3. Shao, R.Y., Lin, S., Fossorier, M.P.C.: Two simple stopping criterion for turbo decoding. IEEE Trans. Commun. **47**(8), 1117–1120 (1999)
4. Wu, Y., Woerner, B.D., Ebel, W.J.: A simple stopping criterion for turbo decoding. IEEE Commun. Lett. **4**(8), 258–260 (2000)
5. Ngatched, T.M.N., Takawira, F.: Simple stopping criterion for turbo decoding. Electron. Lett. **37**(22), 1350–1351 (2001)
6. Yu, N.Y., Kim, M.G., Chung, S.U.: Efficient stopping criterion for iterative decoding of turbo codes. Electron. Lett. **39**(1), 73–74 (2003)
7. Hou, J., Lee M.H., Park, J.Y.: Adaptive SNR turbo decoding algorithm and stop criterion. In Proceedings IEEE International Conference on PDCAT, pp. 893–895 (2003)
8. Li, F.-M., Wu, A.-Y.: A new stopping criterion for efficient early termination in turbo decoder designs. In Proceedings IEEE International Symposium on ISPACS, pp. 585–588 (2005)
9. Jiang, W., Li, D.: Two efficient stopping criteria for iterative decoding. In: First International Conference on Communications and Networking in China, 2006 , pp. 1–4 (2006)
10. Zhanji, W., Mugen, P., Wenbo, W.: A new parity-check stopping criterion for turbo decoding. IEEE Commun. Lett. **12**(4), 304–306 (2008)
11. Gazi, O.: New early termination method for turbo decoders. In: Signal Processing and Communications Applications Conference (SIU), pp. 1215–1218 (2014)