Turbo Decoder Performance Improvement for OpenLTE and A Case Study for DLSCH

Ziming He, John Wilson
Path Intelligence Limited, Portsmouth, England, UK
E-mail: {ziming.he, john.wilson}@pathintel.com

Abstract—The report compares the performances of turbo decoder function in OpenLTE (i.e., turbo_decode function in \(/liblte/src/liblte_phy.cc\) and that in open source software IT++ (http://itpp.sourceforge.net/4.3.1/). The performances are investigated in terms of both error-correction ability (e.g., bit-error-rate and detection probability) and computational complexity. It is found that the IT++ turbo decoder significantly outperforms the OpenLTE turbo decoder in terms of error-correction. For example, it improves the downlink shared channel (DLSCH) detection probability by at least 6 dB in signal-to-noise ratio.

I. INTRODUCTION OF TURBO CODEC AND DLSCH

The turbo codec is used for major LTE transport channels DLSCH and UL SCH [1, section 5.1.3]. The 3GPP LTE turbo encoder is defined in [1, section 5.1.3.2.1]. The two recursive systematic convolutional (RSC) soft-input-soft-output (SISO) decoders are used for the two constituent encoders in [1, Figure 5.1.3-2], respectively. The input of the decoder is soft bits, i.e., log-likelihood ratio (LLR) of received bit\(^1\). Apart from received soft bits, another input into the RSC SISO decoder is the extrinsic information which represents the LLR of information bits (\(x_k\) or \(x'_k\)). The output of the RSC SISO decoder is also the extrinsic information represents the LLR of information bits. Generally, the BCJR algorithm [2] is used to obtain the extrinsic information output. \(L_{e12}\) and \(L_{e21}\) are the extrinsic information output from decoder 1 and 2, respectively, and \(L'_{e12}\) and \(L'_{e21}\) are interleaved and de-interleaved version of \(L_{e12}\) and \(L_{e21}\), respectively. For each iteration, \(L_{e12}, L_{e21}, L'_{e12}\) and \(L'_{e21}\) are calculated and the combined LLR \(L(\hat{x}_k) + L_{e12} + L_{e21}\) are used to perform hard bits decision. The IT++ provides the implementation of 4 types of turbo decoders (in TurboCodec Class) under the framework of BCJR, they are MAP, LOGMAP, LOGMAX and QLLR decoders, MAP is the original method, and the others are reduced complexity method based on LLR computation in logarithm domain. Scaling of the extrinsic information affects the error correction performance of LOGMAX decoder, for the other decoders, the scaling factor can be simply set as 1. In theory, signal-to-noise (SNR) estimation at receiver benefits the turbo decoder and IT++ TurboCodec Class can support this. However, the current OpenLTE does not implement this. Therefore, in Sec. II and III, we only consider turbo decoding without the knowledge of noise level and SNR.

A typical application of turbo code is in DLSCH, the transmission chain is shown in Fig. 2. For a code block, DLSCH receive chain need to know the size of \(d\) bits \(E\) and \(E\) bits 3D, where \(D = K + 4\) and \(k\) is the Turbo code internal interleaver size for the code block defined in [1, section 5.1.2].

1Soft bit is a real value, for example in openLTE, if soft bit is larger than 0, the hard decision bit is 0, otherwise 1.
II. PERFORMANCE OF OPENLTE AND IT++ TURBO DECODERS

This section evaluates the performance of OpenLTE turbo decoder (\texttt{turbo\_encode}) function in \texttt{/liblte/src/liblte/} and IT++ turbo decoders with different SNRs and number of iterations ($N_{\text{iter}}$). Fig. 3-6 evaluate the error correction performance in terms of bit error rate (BER), Fig. 7 evaluates the computation complexity performance in terms of computational time in msec (run in a Linux machine with Intel i7 processor). The simulation results indicates that IT++ turbo decoders significant outperforms openLTE turbo decoder in terms of error correction, with the pay of computational time. In the following, we investigate how to choose a suitable IT++ turbo decoder for LTE.

Fig. 3 indicates that the LOGMAX decoder with scale factor 0.7 performs a slightly better than that with scale factor 1. Therefore, we only use LOGMAX decoder with scale factor 0.7 in the following simulations. Fig. 4 shows LOGMAX, LOGMAP, QLLR decoders perform the almost same in terms of error correction. Fig. 5 shows LOGMAX decoder performs almost the same with MAP decoders. The turbo decoder are evaluated with $K = 232$, the results are similar with $K = 168$ and $K = 200$, these parameters are the frequently seen from real FDD-LTE base-stations. Fig. 7 shows the computational time as a function of iterations. The computational time is a linear function with iterations for the IT++ turbo decoders. Note that the number of iterations cannot be configured in OpenLTE turbo decoder, so it is a straight line. Since LOGMAX decoder (scale = 0.7) has the best trade-off between error correction performance and computational complexity, it is chosen to be integrated into OpenLTE. Fig. 6 shows the impact of iterations and indicates we can set the maximum number of iterations to be around 6.

III. PERFORMANCE OF DLSCH DETECTION

The section compares the performance of openLTE and IT++ turbo decoder in terms of DLSCH detection probability in a single code block case (i.e., $B \leq 6144$, $G = E$). The functions used in the simulations are \texttt{dlsch\_channel\_decode()} in \texttt{/liblte/src/liblte\_phy.cc} and a new introduced function \texttt{dlsch\_channel\_decode\_itpp()}. The received bits passing the CRC decoder is treated as a successful detection of DLSCH. The IT++ LOGMAX decoder (scale = 0.7) is chosen and the maximum number of iterations is set to 6. The decoder automatically stops when converge, i.e., decoded bits does not
change any more with further increase of iterations. Fig. 8 shows the detection probability with different $K$ and $G$. $G$ is the size of $f$ bits and defined in both [1, section 5.1.4.1.2] and [1, section 5.3.2.5]. The considered $K$ and $G$ in Fig. 8 are observed from FDD-LTE base-stations in real live. Fig. 8 indicates the IT++ turbo decoder outperforms OpenLTE turbo decoder by at least 6 dB in terms of detection probability. Moreover, the ratio $G/K$ affects the performance, the larger the ratio, the better the performance, as more redundant bits are transmitted after rate matching. Fig. 9 indicates the computational time of IT++ decoder reduces in relatively high SNR range, this is because the convergence is faster in higher SNR and then less number of iterations is required.

REFERENCES
