Code Design for Short Blocks: A Survey

Gianluigi Liva, Lorenzo Gaudio and Tudor Ninacs

Abstract—The design of block codes for short information blocks (e.g., a thousand or less information bits) is an open research problem which is gaining relevance thanks to emerging applications in wireless communication networks. In this work, we review some of the most recent code constructions targeting the short block regime, and we compare them with both finite-length performance bounds and classical error correction coding schemes. We will see how it is possible to effectively approach the theoretical bounds, with different performance vs. decoding complexity trade-offs.

I. INTRODUCTION

During the past sixty years, a formidable effort has been channeled in the research of capacity-approaching error correcting codes [1]. Initially, the attention was directed to short and medium-length linear block codes [2] (with some notable exceptions, see e.g. [3], [4]), mainly for complexity reasons. As the idea of code concatenation [5] got established in the coding theorists community [6], the design of long channel codes became a viable solution to approach channel capacity. The effort resulted in a number of practical code constructions allowing reliable transmission at fractions of decibels from the Shannon limit [7]–[16] with low-complexity (sub-optimum) decoding.

The interest in short and medium-block length codes (i.e., codes with dimension $k$ in the range of $50$ to $1000$ bits) has been rising again recently, mainly due to emergent applications requiring the transmission of short data units. Examples of such applications are machine-type communications, smart metering networks, remote command links and messaging services (see e.g. [17]–[20]).

When the design of short iteratively-decodable codes is attempted, it turns out that some classical code construction tools which have been developed for turbo-like codes tend to fail in providing codes with acceptable performance. This is the case, for instance, of density evolution [21] and extrinsic information transfer (EXIT) charts [22], which are well-established techniques to design powerful long low-density parity-check (LDPC) and turbo codes. The issue is due to the asymptotic (in the block length) nature of density evolution and EXIT analysis which fail to properly model the iterative decoder in the short block length regime. However, competitive LDPC and turbo code designs for moderate-length and small blocks have been proposed, mostly based on heuristic construction techniques [23]–[44]. While iterative codes retain a large appeal due to their low decoding complexity, more sophisticated decoding algorithms [45]–[49] are feasible for short blocks leading to solutions that are performance-wise competitive (if not superior) with respect to iterative decoding of short turbo and LDPC codes.

II. A CASE STUDY

In this section, we provide an exemplary comparison of short codes. We focus on the case study of codes with block length and code dimension $n = 128$ and $k = 64$ bits, respectively, which are the parameters of the shortest code recently standardized by Consultative Committee for Space Data Systems (CCSDS) [50] for satellite telecommand links [51]. The performance of the schemes is measured in terms of codeword error rate (CER) versus signal-to-noise ratio (SNR) over the binary-input additive white Gaussian noise (bi-AWGN) channel, with SNR given by the $E_b/N_0$ ratio (here, $E_b$ is the energy per information bit and $N_0$ the single-sided noise power spectral density). Besides, we discuss other metrics such as the capability to detect errors and (although not exhaustively) the complexity of decoding. For this block size, we defined a list of viable candidate solutions comprising

i. Short binary LDPC and turbo codes, and their non-binary counterparts;
ii. The $(128, 64)$ extended Bose-Chaudhuri-Hocquenghem (BCH) code (with minimum distance $22$), under ordered statistics decoding (OSD);
iii. Two tail-biting convolutional codes with memory $m = 8$ and $m = 11$;
iv. A polar code under successive cancellation (SC) decoding.\(^1\)

The performance of the codes is compared in Figure 1 with three finite-length performance benchmarks, i.e., the 1959 Shannon’s sphere packing bound (SPB) [52] (——), Gallager’s random coding bound (RCB) [53] for the bi-AWGN channel (−−−), and the normal approximation of [54] (−−−).\(^2\) As reference, the performance of the $(128, 64)$ binary protograph-based [27], [58] LDPC code from the CCSDS telecommand standard [51] is provided too (−−−−). The CCSDS LDPC code performs somehow poorly in terms of coding gain. The code is outperformed at moderate error rates (CER $\approx 10^{-4}$) even by a standard regular $(3, 6)$ LDPC code (−−−). The CCSDS LDPC is also outperformed by an accumulate-repeat-3-accumulate (AR3A) LDPC code [59] (−−−−−) an by an accumulate-repeat-jagged-accumulate (ARJA) LDPC code [60] (−−−−).\(^3\) At low

\(^1\)In future versions of the manuscript, the performance of a polar code concatenated with an outer code, under list decoding [49], will be included.

\(^2\)Excellent surveys on performance bounds in the finite block length regime are given in [54]–[56]. A useful library of routines for the calculation of the benchmarks is available at https://sites.google.com/site/durisi/software [57].

\(^3\)All LDPC codes introduced in this comparison have been designed through a girth optimization technique based on the progressive edge growth (PEG) algorithm [61]. A maximum of 200 belief propagation iterations have been used in the simulations (though, the average iteration count is much lower, especially at high SNRs thanks to early decoding stopping rules).
error rates (e.g. CER $\approx 10^{-6}$) the CCSDS LDPC code is likely to attain lower error rates than the above-introduced LDPC code competitors thanks to its remarkable distance properties [27]. The four binary LDPC codes introduced so far perform relatively poorly with respect to the benchmarks (roughly 1 dB away from the RCB at CER $\approx 10^{-4}$). Despite its uninspiring performance, we shall see in the Section II.B that the CCSDS LDPC code design is particularly suited for application to satellite telecommand links.

The performance of a turbo code introduced in [62] based on 16-states component recursive convolutional codes is also provided (- – –). The turbo code shows superior performance with respect to binary LDPC codes, down to low error rates. The code attains a CER $\approx 10^{-4}$ at almost 0.4 dB from the RCB. The code performance diverges remarkably from the RCB at lower error rates, due to the relatively low code minimum distance.\footnote{In [31], design improvements for the specific case of (128, 64) turbo codes have been presented, which are able to overcome the error floor issue down to CER $\approx 10^{-7}$. The proposed design leverages on the use of tail-biting component codes together with a thorough interleaver search procedure.}

Finally, the CER of three tail-biting convolutional codes has been included [65]. The first code (– –) is based on a memory-8 encoder with generator polynomials (in octal form) given by [515 677]. The second code (– –) is based on a memory-11 encoder with generator polynomials [5537 6131]. The wrap-around Viterbi algorithm (WA V A) algorithm has been used for decoding [66]. The memory-11 convolutional code reaches the performance of the BCH and LDPC codes under OSD. The memory-8 code loses 1 dB at CER $\approx 10^{-6}$, but still outperforms binary LDPC and turbo codes over the whole simulation range. The third code (– –) is based on a memory-14 encoder [67] with generator polynomials (in octal form) given by [75063 56711]. The code outperforms all other codes in Figure 1 (at the expense of a high decoding complexity due to the large number of states in the code trellis).

A. The Elephant in the Room: Complexity

In the comparison presented at the beginning of this Section, an important aspect has been (purposely) overlooked: the cost of decoding. The codes that perform close to the SPB rely on relatively complex decoding algorithms. An exhaustive decoding complexity comparison would require a lengthy and rigorous analysis. Moreover, aspects that are not directly measurable in terms of algorithmic complexity (such as, for example, the probability vs. log-likelihood ratio domain form of the decoding algorithms) but still have large impact in hardware implementation can be difficultly compared. We provide next only a few qualitative remarks on complexity aspects for the decoding algorithms employed in the simulations.

Remark 1 (Binary vs. non-binary iterative decoding). Binary iterative decoding for LDPC and turbo codes can be efficiently performed in the logarithmic domain, with obvious benefits for finite precision (hardware) implementations. The belief propagation algorithm for the non-binary LDPC and turbo codes presented in this manuscript is performed in the probability domain to allow for FFT-based decoding at the check nodes [68]. Thanks to the FFT, complexity of iterative decoding is proportional to $q \log_2 q$ (being $q$ the field order), whereas the conventional iterative decoding complexity would scale with $q^2$. From an algorithmic complexity viewpoint, it has been estimated that the FFT-based decoding of the (128,64) non-binary LDPC code is $\approx 64$ times larger than the one of (iterative decoding of) the CCSDS LDPC code [64]. Complexity reductions for non-binary LDPC codes can be obtained by applying sub-optimum check node update rules, with various trade-offs between coding gain and decoding complexity [69].

Remark 2 (OSD vs. non-binary iterative decoding). For the code parameters adopted in this comparison, OSD and non-binary belief propagation decoding over a finite field of order 256 have similar decoding complexities, as documented in [70]. However, the decoding complexity of OSD scales less favorably with the block length than that of (non-binary) belief propagation (which is linear in the block length for a fixed iteration count and ensemble degree distributions pair).
Hence, for larger block lengths OSD may be considered impractical. Efficient OSD variants have been introduced during the past decade, which may extend the range of interest for OSD algorithms (see e.g. [46]).

B. Error Detection

Some of the algorithms used to decode the codes in Figure 1 are complete, i.e., the decoder output is always a codeword. Incomplete algorithms, such as belief propagation for LDPC codes, may output an erasure, i.e., the iterative decoder may converge to a decision that is not a (valid) codeword. Hence, while for complete decoders all error events are undetected, incomplete ones provide the additional capability of discarding some decoder outputs when decoding does not succeed. In some applications, it is of paramount importance to deliver very low undetected error rates. This is the case, for instance, of telecommand systems, where wrong command sequences may be harmful. The CCSDS LDPC code has been designed with this objective in mind, trading part of the coding gain for a strong error detection capability [71]. Complete decoders, such as those based on OSD and Viterbi decoding, may be used in such critical applications by adding an error detection mechanism. One possibility would be to include an outer error detection code. Nevertheless, in the short block length regime the introduced overhead might be unacceptable. In this context, a more appealing solution is provided by a post-decoding threshold test as proposed in [72]. Denote by $y = (y_1, y_2, \ldots, y_n)$, with $x + n$, the bi-AWGN channel output for a given transmitted codeword $x$ (n is the noise contribution here). We refer to the conditional distribution of $y$ given $x$ as $p(y|x)$. We further denote the maximum likelihood (ML) decoder decision as

$$x_{\text{ML}} := \arg\max_{x \in \mathcal{C}} p(y|x).$$

In [72] the metric

$$\Lambda_{\text{ML}}(y) := \frac{p(y|x_{\text{ML}})}{\sum_{x \in \mathcal{C}} p(y|x)}$$

was proposed and it was proved that the rule for discarding the decoder decision given by the threshold test

$$\Lambda_{\text{ML}}(y) < \exp(nT)$$

is optimal in the sense on minimizing the undetected error probability for a given (overall) error probability. The metric (1) is in general complex to compute (with some notable exceptions, see e.g. [73], [74]) due to evaluation of the denominator of (1) (which requires a sum over all possible codewords) and to the need of the ML decision $x_{\text{ML}}$. In the case of OSD (and of list decoders in general) an approximation of the metric (1) can be easily obtained by summing the conditional distribution $p(y|x)$ over the codewords present in the list, only. The resulting metric would then be given by

$$\Lambda_{\text{OSD}}(y) := \frac{p(y|x_{\text{OSD}})}{\sum_{x \in \mathcal{C}} p(y|x)}$$

with

$$x_{\text{OSD}} := \arg\max_{x \in \mathcal{C}} p(y|x).$$

(2)
being \( \mathcal{L} \) the list produced by the OSD algorithm. While the performance of the test based on the metric (1) has been extensively studied (see e.g. [72], [75]) the authors are not aware of any attempt at analyzing the performance of the metric (2).

III. CONCLUSIONS

An overview of the recent efforts in the design and analysis of efficient error correcting codes for the short block length regime has been provided. A case study tailored to \((128, 64)\) binary linear block codes has been used to discuss some of the trade-offs between coding gain and decoding complexity for some of the best know code/decoding schemes. The comparison, though incomplete,\(^5\) highlights some promising directions for the design of short and moderate-site block codes.

REFERENCES


\(^5\) Further approaches deserving a particular attention for short and moderate-length codes are, among others, those in [76], [77].