Performance-Complexity Tradeoffs for Turbo and Turbo-like Codes

Sam Dolinar, Dariush Divsalar, Aaron Kiely, Fabrizio Pollara
Jet Propulsion Laboratory, California Institute of Technology
e-mail: {sam,dariush,aaron,fabrizio}@shannon.jpl.nasa.gov

Abstract — **We take a first empirical step toward understanding the performance-complexity tradeoffs that have newly arisen since the advent of turbo codes. For a set of turbo and turbo-like codes, we study this tradeoff, measuring the degree to which the code/decoder fails to attain the sphere packing bound.**

I. Introduction

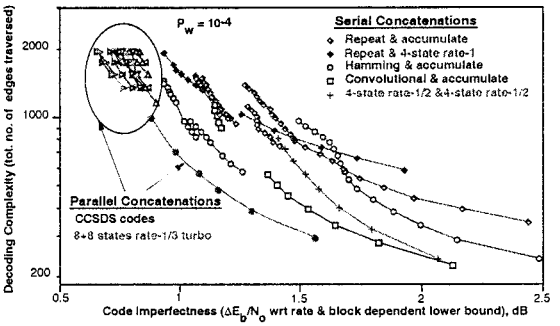
The introduction of turbo codes radically changed the coding community's understanding of the fundamental tradeoffs between performance and complexity that govern codes operating near capacity: powerful near-optimum codes can be decoded with algorithms only slightly sub-optimum but orders of magnitude less complex than optimum decoders. However, even within the world of turbo codes, it has been observed that codes with higher structural complexity can outperform simpler codes (e.g., turbo codes with 16- or 8-state components vs. those with 4- or 2-state components). Furthermore, new turbo-like codes whose performance approaches capacity even more closely than that of the original turbo codes seem to prevail only at the cost of many more decoding iterations. As yet, there is no good theoretical basis for analyzing this performance-complexity tradeoff.

II. Definitions and Tradeoffs

Our performance measure is defined so as to normalize out its dependence on code size and rate. We measure the "imperfectness" of an (n, k) code (and its associated decoder) with respect to the performance predicted by the sphere packing bound on the additive white Gaussian noise (AWGN) channel. This imperfectness is measured (in dB) as the excess bit-SNR required to achieve a specified word error probability P_w over that predicted by the bound.

There are many possible facets to the complexity of a code and its associated decoder: number of computations, amount of memory, complexity of primitive operations, structural complexity, iterative complexity. We ignore fine details of particular implementations, and consider iterative decoding using the sum-product algorithm (also called APP algorithm). In the APP algorithm, each primitive product operation multiplies two probabilities, and each primitive sum operation adds two probabilities. The log-APP variant can be dealt with similarly. Denoting the complexity of a generalized product or sum operation by κ_p or κ_s, respectively, we define the total decoding complexity per decoded bit in one iteration for the jth component decoder as $\kappa_j = O_{p,j}\kappa_p + O_{s,j}\kappa_s$, where $O_{p,j}$, $O_{s,j}$ are the number of product and sum operations performed by the jth component decoder per decoded bit in one iteration.

During one turbo decoding iteration, each component decoder performs three generalized product operations for each edge of the code trellis: for the forward and backward metrics, and for the extrinsic information to be used in the next iteration. The number of generalized *product operations* per decoded bit per iteration is (neglecting minor details) $O_{p,j} = 3E_j$, where E_j is the number of trellis edges per decoded bit for the jth decoder. The number of generalized *sum operations* can be expressed as $O_{s,j} = 2(E_j - V_j) + (E_j - 2)$, for parallel turbo codes or for the inner code of serial codes. However, for an outer rate-$1/n_j$ code of a serially concatenated code, the sum operations must be performed for each of the n_j code symbols that label a given trellis edge. In this case the formula becomes $O_{s,j} = 2(E_j - V_j) + n_j(E_j - 2)$, and thus $\kappa_j/E_j = 3\kappa_p + (2 + n_j - 2V_j/E_j - 2n_j/E_j)\kappa_s$. For rate $1/n_j$ component codes, $E_j = 2V_j$, which gives $\kappa_j/E_j = 3\kappa_p + (n_j + 1 - 2n_j/E_j)\kappa_s \le 3\kappa_p + (n_j + 1)\kappa_s$. We avoid the dependency on the individual complexities κ_p, κ_s by bounding κ_j in terms of $(\kappa_p + \kappa_s)$: $\kappa_j \le 3[\kappa_p + (n_j + 1)\kappa_s]E_j \le (\kappa_p + \kappa_s)\max[3, n_j + 1]E_j$. This allows us to separate the decoding complexity into an implementation-dependent factor $(\kappa_p + \kappa_s)$ and a factor determined by the number of edges in the trellis. Keeping track of the number of bits k_j input to each component decoder per information bit input to the overall decoder and multiplying by the total number of iterations I, we arrive at the total complexity $\kappa = I\sum_j k_j\kappa_j \le I(\kappa_p + \kappa_s)\left[\sum_j k_j \max[3, n_j + 1]E_j\right]$, with three factors: the number of iterations, an implementation-dependent factor, and a structural complexity factor.



We have plotted the above bound on total complexity as a normalized quantity, $\kappa/(\kappa_p + \kappa_s)$, showing complexity-performance curves for a number of turbo and turbo-like codes for $P_w = 10^{-4}$. Each curve corresponds to one code, and therefore a constant structural complexity, and it is traced by varying the number of iterations I. The curves clustered in the upper left corner of the figure are for the CCSDS family of parallel turbo codes, using 16-state components, and ranging in rate from 1/2 to 1/6 and in block size from 1784 to 8920 bits. The tradeoff curve for an 8+8-state parallel turbo code, with rate 1/3 and block size 1024, is shown over a wider tested SNR range. Additional curves are shown for several families of serially concatenated codes. One family consists of simple "repeat and accumulate" (RA) codes, using an outer repetition code of rate 1/3 or 1/4. Two families of rate-1/2 RA codes are obtained by replacing the outer repetition code with either the (8,4) Hamming code or the octal (5,7) 4-state convolutional code. A fourth family is formed by a rate-1/3 repetition outer code with a 4-state (1/7) rate-1 inner code. Finally, a rate-1/4 code is obtained by concatenating the octal (5,7) 4-state rate-1/2 convolutional code with a recursive 4-state octal (1,5/7) rate-1/2 inner code. Separate curves within each code family pertain to different block sizes. The parallel concatenated codes exhibit generally better performance at lower complexity than the serially concatenated codes. However, this picture reverses at lower error rates, e.g., $P_w = 10^{-6}$. We are working toward a deeper understanding of the lower envelope of the complexity-performance curves.

The work described was funded by the TMOD Technology Program and performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.