Iterated Decoding of Modified Product Codes in Optical Networks

Jørn Justesen
Department of Photonics
Technical University of Denmark
DK-2800 Kgs.Lyngby, Denmark
Email: joju@fotonik.dtu.dk

Abstract— **Appendix I of the standard ITU-T G.975 contains several codes that have been proposed for improved performance of optical transmission. While the original application was submarine cables, the codes are now also used in terrestrial systems where wavelength-division multiplexing (WDM) is introduced. Currently codes suitable for data rates of 100 Gbits/sec in each channel are being studied. We discuss performance limits for codes with the specified block lengths of about 130,000 and rates about 15/16. The most promising codes appear to be modified product codes, and iterated decoding is used in several cases. We discuss the construction, decoding, and performance of such codes.**

I. Introduction

In the ITU-G975 standard, Appendix I [1], several codes are suggested for use in optical networks. Here the frame length is $N = 255 * 512$ and the rate is 239/255 (since the codes replace interleaved 8-error correcting Reed-Solomon codes). The target performance is an output bit error probability of 10^{-15} for a channel error rate of about 10^{-3}. The improved error-correcting power was first used on submarine cables, but is now required when wavelength-division multiplexing in introduced in existing systems. The errors are mostly caused by non-linear interference between channels, and soft-decision information is not assumed to be available (or useful). The decoder has to operate at the channel bit rate of 10-40 Gbit/sec, but future codes for 100 Gbit/sec are currently being studied. The complexity of the proposed schemes clearly demonstrates that this application represents a significant challenge, both in terms of the code construction and the decoder implementation.

II. Performance Limits for Random Codes

As a starting point we specialize well-known results for the performance of long codes on the binary symmetric channel to the given parameters. Asymptotic results indicate that for high rates, the error exponent is given by the sphere-packing bound, and this exponent can be reached with a randomly chosen code. Translated to a fixed long code and varying channel error probability this means that for poor to moderate channels, the decoding error probability can be estimated by the performance of a code meeting the Hamming bound. For the rate in question we find the upper limit on the error probability, p, as

$$H(p) = 0.063, p = 0.0073.$$

If the block length is limited to 130,560 bits, the average number of errors per block could be at most $t = 953$. But to get an error probability of 10^{-15} we need to accommodate about $t + 8 * \sqrt{t}$ errors. The margin depends on the number of errors that are left in frames that are not correctly decoded. Assuming that this is on the order 100, the probability of such an event must be at most 10^{-12}. Thus the average number of errors in a block cannot exceed 735 and $p < 0.0056$.

If a code with the given parameters is decoded maximum-likelihood (ML), the performance curve can be calculated as the probability that the number of errors on the channel exceeds 950. For channels with $p < 0.005$, the error probability will eventually be dominated by error patterns of lower weight, and we need to know the weight distribution of the code. However, for an average code this is not necessary in the range of interest.

A code with low minimum distance has the same performance on weak channels, but the performance curve will have a smaller slope on better channels. If the decoding is not ML, the performance is worse, mostly because of decoding failures, i.e. cases where the decoder does not reach a codeword.

We do not know of code constructions or decoding methods that come close to the optimal performance. The various sections of the standard propose methods than range from a single algebraic code to iterative decoding of LDPC codes. Several of the codes fail to reach the required performance. The most successful proposals are based on a modification of product codes or concatenated codes. Such codes are analyzed in the remaining part of the paper.

III. Modified Product Codes and Concatenated Codes

A codeword in a product code can be described as an array of symbols where each row and column is a codeword of a component code. In a concatenated code the columns are Reed-Solomon(RS) codes, but the rows are encoded by a binary code with a suitable mapping of the RS symbols as binary vectors. For simplicity we refer to all these codes as product codes. Since the size of the codewords would allow only an array of, say 255 by 512 bits, a code on a single row would be too short to give the required rate. Thus the constructions use component codes that cover several rows or columns. We refer to such codes as *block product codes*. In this section we give a few properties of such codes, and in particular we note the changes compared to usual product codes. For simplicity we assume that all row codes are identical and binary, and that the same is true for the column codes, although the column codes may differ from the row codes. Extension of these remarks to the case where one of the codes is an RS code is straight forward.

Let the codeword be an r by s array of vectors of length b, and require each row to be a codeword of a component code of length $n_1 = rb$, and each column to be a codeword in a component codeword of length $n_2 = sb$. Thus the overall code has length $N = rsb = n_1n_2/b$. The parity check matrix can be described as a block matrix consisting of b by b sub-matrices, and the 'check on check' rectangle of the codeword will be the same when computed rows first or columns first only if the sub-matrices commute. If this is not the case, the dimension of the code is reduced, or we may prefer to calculate the 'checks on checks' only in one dimension. The minimum distance of the code no longer follows directly from the two minimum distances.

We may assume that the row codes correct the larger fraction of errors, and thus they are decoded first. Since the rate of the component codes is high, only error patterns within half the minimum distance are corrected, and most heavier error patterns cause decoding failure. The best approach is to leave uncorrected codewords unchanged. Since the initial distribution of errors in the rows is binomial, the average number of remaining errors can easily be calculated. When the columns codes are decoded, the total number of remaining errors is reduced, but in general the resulting bit error probability is not sufficiently low. For this reason several iterations of decoding rows and columns is specified in the standard. After several iterations either all errors are corrected, or there are more than t_1, t_2 errors left in each of the remaining rows and column. The number of remaining errors in this situation is not directly related to the minimum distance. The cases where the decoding leads to a codeword different from the one that has been transmitted usually do not contribute significantly to the output error probability.

In a usual product code the smallest error pattern that is not decoded by iterated decoding has weight $(t_1 + 1)(t_2 + 1)$ or a little more than one quarter of the minimum distance. However, in a block product code, a single block may contain enough errors to cause decoding failure.

Since an analysis of iterated decoding of product codes has only recently be presented, we summarize the result in the next sections.

IV. Cores in Random Graphs

We describe the product of two codes of lengths n_1 and n_2 as a complete bipartite graph. The right vertices represent row codes and the left vertices column codes. The code symbols are labels on the branches, and symbols on the branches that connect to a given vertex have to satisfy the parity checks of the corresponding code.

The row codes correct all error patterns of weight at most t_1, the column codes correct t_2 errors. A total of W errors are assumed to occur at randomly chosen positions. Since the decoding is independent of the codeword and the error values, it is sufficient to consider the error graph, a bipartite graph with $n_1 + n_2$ vertices and W randomly chosen branches.

In the present analysis we make the simplifying assumption that the decoder of the component code corrects t_i or fewer errors, and in other cases the symbols are left unchanged. If t_i is not too small, the probability of decoding errors when more than t_i errors occur, is approximately $1/t_i!$. This result is well known for RS codes, but it also true for high rate BCH codes. For binary codes correcting a small number of errors, we always prefer codes of even distance to reduce the probability of decoding error.

If $t_1 = t_2$ and the decoding of the component codes is repeated until a stable result is obtained, decoding failure is described by the following concept:

Definition: A k-core in a graph is a subgraph with the property that all vertices have degree at least k.

Lemma 1: Iterative decoding of the product code fails if and only if the error graph contains a $t + 1$ core.

The well-known procedure for finding a core in a graph consists in successively removing any vertex of degree less than k and all branches connected to it. In terms of the product code, this procedure clearly amounts to decoding component codes and correcting all error patterns of weight at most t.

The existence of cores in random graphs has been a subject of considerable interest in graph theory. In particularly the following result due to Pittel et al. is important [2]: Let G be a random graph with n vertices and m edges. For any $k > 2$, a k connected core exists with high probability when $m > c_kn/2$, but not for smaller m. The core includes a large fraction of the vertices. Here

$$c_k = min_\lambda[\lambda/\pi_k(\lambda)], \lambda > 0$$
$$\pi_k(\lambda) = P[Poisson(\lambda) \geq k - 1]$$

Thus $c_3 = 3.35$, $c_4 = 5.14$, $c_5 = 6.80$, $c_6 = 8.37$, $c_9 = 12.78$. Asymptotically $c_k \approx k + \sqrt{klogk}$.

The result applies without change to random bipartite graphs. The argument can be based on a simplified proof of the basic result [3].

The result in [2] is asymptotic, i.e. k is fixed while the size of the graph increases. Thus the result for product codes is exact for codes of increasing length and fixed error-correcting capacity.

V. Alternating Between Decoding Rows and Columns

We can get a model that is closer to the actual decoding of product codes. Initially the number of errors in each row follows a Poisson distribution since n is large compared to t. The fraction of the W errors which are decoded when all row codes are decoded can then be found from this distribution as

$$\sum_{x \leq t} xe^{-m}m^x/x!$$

We now introduce the simplifying assumption that these decoded positions are randomly distributed in the columns. Clearly this is not exactly the case, since t errors in a particular row must be located in different columns. However, as long as the total number of errors is large, the approximation is extremely close. Thus the first decoding of the column codes operates on a Poisson distribution with a reduced mean value.

Lemma 2: If the errors decoded by the row codes are randomly distributed in the columns and visa versa, the distribution after each decoding step is a truncated Poisson distribution.

We can now follow the progress of the decoding:

- Initially the number of errors is $W = Mn$, and the number of errors in each row follows a Poisson distribution with mean M.
- Let $\pi(m)$ indicate the probability that a random variable with Poisson distribution of mean m has a value greater than or equal to t. The expected number of errors after the first decoding is
$$n \sum_{j \geq t} je^{-M}M^j/j! = nM\pi(M)$$
using an equality for the truncated Poisson distribution that is commonly known in traffic theory. If we make the approximation that these errors are randomly distributed in the columns, then the number of errors per column follows a Poisson distribution with mean
$$m(1) = M\pi(M)$$
- Starting from these initial values we prove by induction that the Poisson parameters after the following stages of decoding are
$$m(j) = M\pi(m(j-1))$$
- If the initial value, M, is less than $min\{m/\pi(m)\}$, $m(j)$ converges to zero, while for M less than this threshold, m converges to the largest value such that $m' = M\pi(m')$.

The results of this analysis coincide with the properties of random graphs found in [2]. Simulations of decoding with $t = 8$ confirm that the truncated Poisson distribution of errors at each step of the decoding closely approximates the actual values.

The iteration can be illustrated graphically as a sequence of points on the line $m = x$ and the graph of $M\pi(x)$. The graphic also indicates how the expected number of iterations increases as the number of errors approaches the threshold.

For small values of t, experiments indicate that the best performance (highest rate for a given fraction of corrected errors), is obtained with different values of t, t_1 and t_2 on the right and left respectively. For small values of $t_1 + t_2$, the difference in rate is small, but it increases with the value of the sum. Thus the choice of 8 and 5 errors in the DVD code is good also for iterated decoding.

The original proof of cores in random graphs is not easily modified to work with different values of t in subsets of the vertices. However, in our analysis such a change is easily made. If the definition of the function π is modified to alternate between t_1 and t_2, the parameters are still updated by

$$m(j+1) = M\pi(m(j))$$

The errors are corrected if the initial number of errors is below a certain threshold, but for larger values the decoding process reaches a stationary point with a pair of parameters, $(m', m")$.

The iteration can be illustrated graphically (in the form well-known from EXIT graphs) as a staircase line between the graph of $\pi(x)$ for t_1 and a reflected version of the graph of $\pi(x)$ for t_2. The graphic indicates that the decoding threshold is reached when these two curves touch.

Modified product codes are similarly described by bipartite multi-graphs, and the errors by randomly selected sub-graphs. Thus much of the analysis is still valid, but the approximation is not so close, since the number of component codes is smaller. Nevertheless it is still true that for $t_1, t_2 \geq 2$, if the average number of errors in each row is initially not much larger than t_1, and the average number of errors in a column after the first decoding is not larger than t_2, all errors are decoded with high probability.

VI. Performance of Block Product Codes

For block product codes we can still find a good approximation to the average number of errors at each step of the decoding by assuming truncated binomial distributions and applying the approximation that errors corrected in one dimension are randomly distributed in the other dimension. However, this analysis does not give a bound on the probability of decoding failure in cases where the decoding succeeds on the average.

The asymptotic analysis referenced above does include a bound on the probability that a core exists, but it is not sufficiently tight for our purpose. Thus an estimate of the performance of a specific code must in part be based on simulations. The performance for low bit error rates can be calculated based on our understanding of the properties of the iteration.

When the channel is good enough for the decoding to succeed in most cases, the step curve discussed above passes through a relatively narrow gap between the two limiting graphs. Thus most cases of decoding failure occur when the decoding stops in this bottleneck, and the number of remaining errors is roughly constant (and a large fraction of the initial number of errors). This part of the performance curve can easily be simulated, and it can be verified that the number of remaining errors does not change much with the channel parameter. Thus the error probability falls off steeply, and if continued, the slope of the performance curve in the usual log-log plot would equal the expected number of remaining errors. The observable part of the curve does not reach this slope, but we can get an upper bound on the error probability by extending it with the observed slope.

If the decoding passes the bottle neck region, it is unlikely to stop until very few errors are left, since the boundary graphs are far apart. Clearly the error probability on very clean channel is given by the minimum weight error pattern that cause decoding failures, i.e. the smallest possible cores. However, it is easy to count the number of such cores and thus to give a close bound on the probability of these events. As noted previously the smallest cores in usual product codes have $t_1 + t_2$ nodes, but in a block product code, a decoding failure may be caused by t_1 errors in a single block (assuming $t_1 > t_2$). The output bit error probability due to such an event is upper bounded by

$$P_e < rs\binom{b}{t_1}p^{t_1}t_1/N$$

Again this is a straight line with slope t_1 in the usual format. Since there are no other significant contributions to the error probability, the result is the sum of these two terms, and the performance curve changes quite sharply from the initial steep decline to the 'error floor' of the second term. It may not be practical to simulate the performance at low bit error rates for the codes under consideration, but the behavior described here can be verified in smaller codes.

Example: The block product code of Section 4. One of the codes that has been used is the RS-BCH block product code in Section 4 of the Appendix. Here we give a slightly simplified description, and use the above analysis to derive the expected performance. In this code the components are 16 RS codes over $F(2^{10})$ shortened to 781 symbols and 64 BCH codes of length 2047. All component codes correct 8 errors. Two component codes intersect in about 120 bits. The (inner) BCH codes are corrected first. Since they correct a fraction of $8/2047 = 0.004$ errors, we expect the code to start being effective at this point, which is confirmed by simulations. At $p = 0.003$ the output error probability is 10^{-5}, and decoding failures typically leave about 200 errors. For decreasing values of p, the slope of the performance curve is at least 50. However 9 errors in a single block causes decoding failure leading to an 'error floor' of

$$P_e = \binom{120}{9}9p^9/120$$

This gives a line that reaches 10^{-15} at $p = 10^{-3}$. The two lines intersect for an output error rate of 10^{-12}, and thus the 'error floor' is important for the desired performance, but difficult to observe in simulations.

VII. Decoding at High Data Rates

One reason for the choice of modified product codes is that the decoders can be implemented in gate-array technology at very high data rates.

Since the component codes are BCH (or RS) codes, the initial step in decoding of the row codes is the syndrome calculation. In many real decoders this is the step that requires most resources. Working in parallel on the rows gives a significant reduction of the clock rate, but it is not sufficient to reach the target rate. In an RS code, the clock rate in the symdrome calculation is further reduced by the number of bits in each symbol. In a binary BCH code, the division by a polynomial can be performed on blocks of input symbols using an xor function of more variables. Solving for the error-locator is a complex calculation, but it requires relatively few steps when the component code is long and has high rate. It is an advantage if the search for error locations can be replaced by a more direct factorization of the locator polynomial.

The first decoding of the columns is similar, and the two steps are pipelined. When the decoding is iterated, one possibility is again pipelining of identical decoders. However at least in principle, a faster approach is to modify the syndromes to account for the corrections in the previous steps, thus avoiding the relatively expensive syndrome calculation. With this method, the problem is not so much the amount of computation as the problem of routing the data to the locations where they are needed.

The existing implementations indicate that the target data rate of 100 Gbit/sec (producing 100 Merrors/sec!) is a significant technical challenge, but not unrealistic.

References

[1] International Telecommunication Union, *ITU-T Recommendation G.975.1 Forward error correction for high bit-rate DWDM submarine systems*, 2004.

[2] B. Pittel, J. Spencer, and N. Wormald: "Sudden emergence of a giant k-core in a random graph", *J.Comb.Theory, Series B*, vol.67, pp. 11-151, 1996.

[3] J. Justesen and T. Hoeholdt: "Analysis of iterated hard decision decoding of product codes with Reed-Solomon component codes", *Proceedings ITW 2007.*