# Concatenated Codes for Deletion Channels

# Concatenated Codes for Deletion Channels

Johnny Chen
Michael Mitzenmacher
Chaki Ng
and
Nedeljko Varnica

TR-07-03


Computer Science Group
Harvard University
Cambridge, Massachusetts

# Concatenated Codes for Deletion Channels

Johnny Chen, Michael Mitzenmacher, Chaki Ng and Nedeljko Varnica
Division of Engineering and Applied Sciences
Harvard University
johnnyc@eecs.harvard.edu, michaelm@eecs.harvard.edu, chaki@eecs.harvard.edu, varnica@hrl.harvard.edu

*Abstract*— **We design *concatenated* codes suitable for the deletion channel. The inner code is a combination of a single deletion correcting Varshamov-Tenengolts block code and a marker code. The outer code is a low-density parity-check (LDPC) code. The inner decoder detects the synchronizing points in the received symbol sequence and feeds the outer LPDC decoder with soft information. Even using regular LDPC codes as outer codes our results are promising. Our simulation results demonstrate that the bit error rates of $10^{-6}$ can be obtained at rate 0.21 when the probability of deletion is 8%.**

## I. Introduction

In the memoryless deletion channel model [1], each transmitted symbol is independently deleted with probability $P_d$; otherwise it is transmitted correctly. This model is similar to the erasure channel, but with the additional difficulty that the positions of the lost symbols are not known. Lower bounds on the capacity of the memoryless deletion channel can be found in [1].

Early work involving the deletion channel used block codes capable of correcting a single deletion per block [2] [3] [4], under the assumption that the boundaries of blocks could be found. These codes were of primarily theoretical interest, since they do not address synchronization and multiple deletions per block.

Recently, a block code that can correct multiple deletions, insertions and substitutions was constructed and implemented [5]. (See also the related [6] and [7].) This code combines sparsified LDPC encoded bits [8] and watermark bits in the encoder, and uses a forward-backward algorithm to maintain synchronization in the decoder [5]. The maximum code rates obtainable by this method are given in [9]. These schemes represent a step toward good practical code constructions for deletion channels (and similar models). They also motivate the search for a better, and possibly less complex, coding method for medium-to-high deletion probabilities ($P_d = 3\% - 10\%$).

In this paper we propose concatenated block codes that achieve promising results for medium to high probabilities of deletion, while keeping the complexity of the encoding/decoding algorithm low. The inner code in our scheme is a Varshamov-Tenengolts (VT) code [4] with inserted marker bits used for detecting synchronization points in the decoder. The inner decoder returns the soft information to the outer LDPC decoder, which recovers the remaining lost bits. As an example of our approach, we develop a code with rate 0.21 and bit error rate less than $10^{-6}$ when the probability of deletion is 8%.

This paper is organized as follows. In Section II we review the codes used in our encoding/decoding scheme. In Section III we present our code construction methods for each encoding/decoding element. We give the simulation results obtained by the proposed scheme in Section IV. Finally, in Section V we conclude the paper and discuss possible improvements.

## II. The Concatenated Code Structure

Our coding scheme is the concatenation of three codes, as shown in Figures 1 and 2. The outer code in our scheme is an LDPC code [8], and the inner code is a combination of a VT code [4] and a *marker code* [10]. The *marker code* here refers to the encoder that inserts bits into the VT encoded bit-sequence, and the decoder that detects the synchronizing points in the received bitstream.

### A. Outer Code - A Low Density Parity Check Code

LDPC codes are very well known error-correcting codes discovered by Gallager [8]. These codes achieve near Shannon-capacity performance on a variety of communication channels, given large enough code block lengths[1] and the reasonable computational resources [12] [13] [14] [15].

LDPC codes are typically represented by bipartite graphs [16]. The decoding algorithm that is normally performed on this graph is based on message-passing. The messages being passed across the edges of the graph can be symbols or probabilities corresponding to the symbols. The former decoding method is called *hard decoding*, and the latter is known as *soft decoding*. For more details about the hard and the soft message-passing decoding on LDPC graphs we refer the reader to [8] [12]. Here we concentrate on soft LDPC message-passing decoding, since it is suitable and advantageous given the concatenated coding scheme that we propose.

### B. Inner Code

The inner code is comprised of a marker code, which maintains synchronization during decoding, and a VT code, which passes probabilities to the outer code for soft decoding. In particular, the VT code corrects single errors, so it passes very good probabilities when there is

[1] Even at shorter block lengths (less then several thousand) the performance of multi-edge type LDPC codes [11] is comparable to the turbo codes performance.

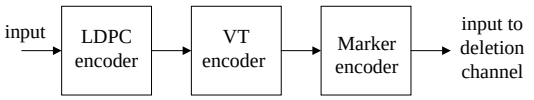


Fig. 1. Encoder structure. Encoder consists of an outer LDPC encoder and an inner encoder that ties together a VT encoder and a Marker encoder.

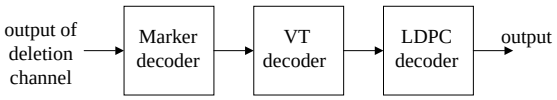


Fig. 2. Decoder structure. Decoder consists of an inner decoder that detects the synchronizing points and feeds an outer LDPC decoder with soft information.

at most one deletion in an inner codeword. We design the inner code so that good probabilities are passed to the LDPC code even when more than one deletion occurs in an inner codeword.

### B.1 Marker Codes

As the codewords are passed through the deletion channel the location and the size of each codeword become unclear. To solve this synchronization problem we adopt the concept of marker code [10]. The basic idea is to insert a marker between each codeword prior to entering the deletion channel. A marker is a set of bits with specific length (marker length), inserted between a predetermined number of code bits. A uniform marker code has the same bit patterns for each marker, while a non-uniform marker code has marker bits that are different throughout [10]. We currently use uniform markers of all 0's in this work.

We chose headers consisting entirely of 0's in conjunction with VT codewords with high Hamming weight. This allows both the long and short headers to be easily detected, and minimizes possible confusion when codeword bits have been deleted. In addition, we chose a set of "self-synchronizing" VT codewords. These codewords have 1 as a terminal bit, providing a natural way of finding boundaries between codewords and header sequences.

### B.2 VT Codes

For fixed $n$ and a parameter $a$ with $0 \leq a \leq n$, the Varshamov-Tenengolts code $\mathrm{VT}_a(n)$ is the set of all length $n$ binary strings $\underline{x} = x_1 \dots x_n$ satisfying

$$\sum_{i=1}^{n} i x_i \equiv a (mod\ (n+1)).$$

This set of codewords can perfectly correct any single deletion; that is, any two strings in this set cannot be confused by a single deletion [3]. It is known that setting $a = 0$ gives the largest codebook for fixed $n$. In fact, $\mathrm{VT}_0(n)$ is conjectured to be optimal (and known to be asymptotically optimal); that is, it provides the largest code on $n$ bits that perfectly corrects for single deletions. Sloane [17] provides a good survey of these codes.

We use VT codes because they are easy to generate for small $n$. Any subset of a VT code can also obviously perfectly correct single deletions. When using VT codes in this work, we often require only a subset of the codewords of the full VT code. Codeword choice is crucial to aid the marker code in synchronization; using the conjectured optimal VT codes provides a large choice of codewords. We use $\mathrm{VT}_0(10)$ in this paper.

## III. Construction of the Concatenated Codes for Deletion Channels

In the following subsections we present our coding scheme, shown in Figures 1 and 2, in more detail.

### *A. The General Scheme*

Information bits are first encoded by an outer LDPC encoder. We denote the LDPC block length with $N$. Then the blocks of $N$ encoded bits are broken into short blocks of length $k$. The VT encoder encodes short blocks into blocks of length $n$. Marker bits are inserted in the *Marker encoder*, see Figure 1.

The inner decoder regains synchronization and obtains the data for the outer decoder. The information that is transferred from the inner decoder to the outer decoder is soft. We found this approach to be very useful, since it is possible to obtain reliable probabilities from the inner decoder that can be passed to the outer message-passing decoder. Construction of the encoding/decoding elements is presented in the following subsections.

### *B. Inner Code Construction*

The VT encoder receives LDPC encoded bits (see Figure 1), and assigns a length $n$ codeword to each block of $k$ bits. The choice of codewords and specific values of $k$ and $n$ are discussed below. The VT codewords are sent to the marker encoder, which inserts synchronizing sequences between codewords. During decoding, the marker decoder receives a block from the channel and attempts to extract the original codewords, some of which are shortened by deletion. If a codeword is extracted correctly and has lost at most one bit, then the original $k$ LDPC encoded bits are recovered. In the case where deletion or synchronization returns a shorter sequence, the inner decoder looks up the sequence from a table and returns $k$ probabilities for the original LDPC bits.

### B.1 Choosing VT Codewords

A VT codeword can pass very reliable probabilities to the outer code if it has lost one or zero bits; in other

cases, there is more to be done. To achieve good performance under high deletion rates, we would therefore like for there to be at most one deletion per inner codeword as much as possible, while still keeping the code rate high. For rates of deletion as high as 10% this implies that a reasonable choice for codeword length $n$ is at most $10-15$. Here, we choose $n=10$, and $k=5$, yielding the VT inner code rate $r_{vt}=k/n=0.5$.

The choice for the value of $k$ was constrained by the inner code. Since $\mathrm{VT}_0(10)$ has 94 codewords, the maximal value of $k$ is 6. The choice of codewords from the 94 possible codewords is governed by trying to make the synchronization of the marker code as effective as possible. Since we choose the synchronization bits that consist of consecutive 0's, we choose VT codewords with high Hamming weight, no long runs of 0's and a terminal 1 to minimize the chance that marker bits are confused with codeword bits.

### B.2 Generating Probabilities

The choice $\mathcal{C} \subset \mathrm{VT}_a(n)$ of codewords allows for $|\mathcal{C}|! = (2^k)!$ possible VT encoding functions $f : \{0,1\}^k \to \mathcal{C}$. We describe our choice of $f$ in Subsection B3. Denote a VT encoder input message by $\underline{b} = b_1 \dots b_k$ and let $\underline{x} = f(\underline{b})$ be the associated codeword. Let $\underline{w}$ be the observed string after $\underline{x}$ has passed through the channel and marker decoding. The VT decoder $g$ assigns a $k$-vector $\underline{p} = (p_1 \dots, p_k)$ of probabilities to $\underline{w}$, where

$$p_i = Pr[b_i = 1 | \underline{w}]$$

is the conditional probability that the $i$th message bit is 1 given that $\underline{w}$ is observed. We compute $p_i$ by summing over possible $\underline{b}$

$$p_i = \sum_{\underline{b}: b_i = 1} Pr[f(\underline{b}) | \underline{w}]. \qquad (1)$$

To make decoding faster we preprocess $g(\underline{w})$ for all $\underline{w}$ and store these values in a table. Decoding then reduces to a table lookup.

### B.3 Choosing the Encoding Function

The choice of $f$ influences the reliability of the generated probabilities. Intuitively, the more likely that two codewords $\underline{x}_1$, $\underline{x}_2$ are confused after being passed through the channel, the smaller the Hamming distance $d_H(f^{-1}(x_1), f^{-1}(x_2))$ should be. We impose a measure $\delta$ on encoding functions $f$ as follows.

Fix $f$. For an observed word $\underline{w}$ and the decoding function $g$ let $\underline{p} = g(\underline{w})$. Define $\hat{\underline{b}} = (\hat{b_1}, \dots, \hat{b_k})$ to be the sequence of bits obtained by hard decoding on $\underline{p}$. For $1 \le i \le k$, let $Z_i^f(\underline{w})$ be the indicator for $\hat{b_i} = b_i$; then $E[Z_i^f(\underline{w})] = \max\{p_i, 1-p_i\}$. Here $Z_i^f(\underline{w})$ is a function of $f$ because $f$ determines the set of messages $\underline{b}$ that can result in $\underline{w}$ after passing through the channel, see Equation (1).

The distribution of $\underline{w}$ depends on the probability of deletion $P_d$. For fixed $P_d$, define

$$\delta_{p_d}(f) = E\left[\sum_{i=1}^{k} Z_i^f\right].$$

Note that $\delta_{p_d}(f)$ represents the number of message bits we expect to decode correctly using hard decoding. For a fixed $P_d$, we consider $f$ *optimal* if it attains the value $\max_f \delta_{p_d}(f)$. Since $\delta_{p_d}(f)$ is computed from the probabilities for all possible received words $\underline{w}$, we can quickly compute $\delta_{p_d}(f)$ using the table of probabilities generated for decoding.

Finding an optimal $f$ appears to be difficult; even for $k=5$ there are 32! possible $f$. Instead we use a "hill-climbing" algorithm for finding locally optimal $f$. Choose a random encoding function $h$. For a pair $\underline{b}, \underline{b}'$ of messages, let the encoding function $h_{\underline{b},\underline{b}'}$ be equal to $h$ except

$$h_{\underline{b},\underline{b}'}(\underline{b}) = h(\underline{b}'),\ h_{\underline{b},\underline{b}'}(\underline{b}') = h(\underline{b}).$$

Call this operation a *swap*. We improve the $h$ by finding the swap $\underline{b}, \underline{b}'$ that maximizes $\delta_{p_d}(h_{\underline{b},\underline{b}'}) - \delta_{p_d}(h)$. Iterating this process until no improving swap exists finds a locally optimal $f$. For the range of $P_d$ considered in this paper we repeated this algorithm for many random choices of $h$ and selected the $f$ with the largest value $\delta_{p_d}(f)$.

## *C. Marker implementation*

### C.1 Marker Encoder

The marker encoder performs two steps on the bitstream. First, a marker is inserted after each codeword. In our implementation for $n$-bit codewords, we used uniform markers consisting of $M$ zeros. Second, a size-$L$ block marker is inserted after every block of $B$ codewords. The goal is to minimize the effect of undecodable codewords on the rest of the bitstream. By having such block markers, we can isolate any block with synchronization problems. Since each block marker is inserted right after a $M$-bit marker at the end of a block, we effectively insert a marker of $M+L$ zeros after each block. See Figure 3 for an example. (For long transmissions, we could include longer markers at a higher level of superblocks, and so on; we do not consider this further here.)

### C.2 Marker Decoder

The goal of the marker decoder is to find the original markers inside the post-deletion bitstream, and identify the post-deletion size and bit pattern of each individual codeword.

The marker decoder first uncovers block markers and divides the codewords into distinct blocks. We choose a large enough block marker size $L$ such that the probability of misidentifying a large marker is negligible.

Next, the decoder scans and decodes within each block separately. To decode individual codewords, the decoder

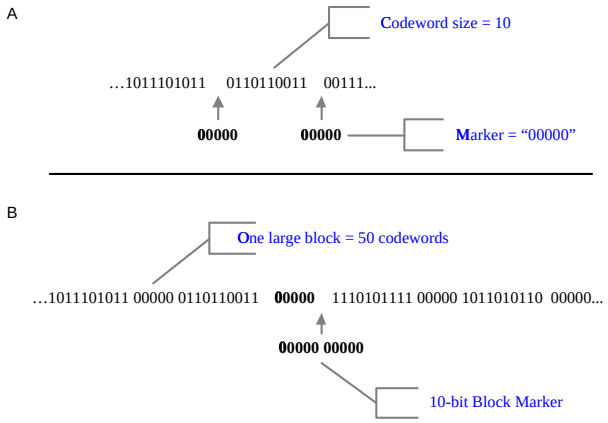


Fig. 3. An example of marker encoder with $n = 10$, $M = 5$, $B = 50$ and $L = 10$.

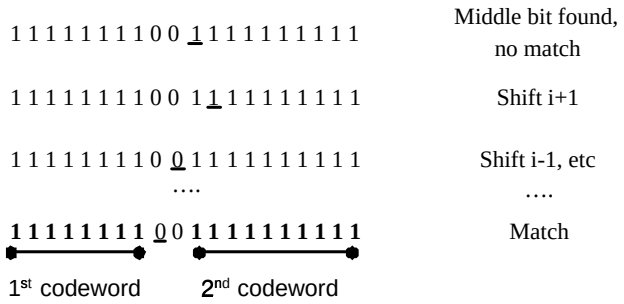


Fig. 4. Splitting algorithm execution example.

first looks for bit patterns of $\ell$ consecutive zeros that match all ($\ell = M$) or most ($\ell_{min} \leq \ell < M$) of the marker.

Let $p_c$ denote the probability of observing a pattern of $\ell_{min}$ consecutive zeros that originate from a VT codeword and let $p_m$ be the probability that the marker is shorter than $\ell_{min}$ bits. To choose the threshold $\ell_{min}$ we want $p_c \cdot p_m$, the probability of misidentifying part of a codeword as a marker, to be as small as possible. We compute $p_c$ via an exhaustive search algorithm. The other probability is simply

$$p_m = \sum_{i=0}^{\ell_{min}-1} \binom{M}{i} P_d^{M-i} \cdot (1 - P_d)^i.$$

For example, if we want $p_c \cdot p_m \leq 10^{-3}$ and $P_d = 10\%$, $n = 10$, $k = 5$ and $M = 5$ then $\ell_{min} \leq 3$. Note that we do not consider here the case when the bits in the pattern are from both markers and VT codewords. We address this scenario and present a solution below.

There are special cases that the decoder needs to handle. The first case is when there are more 0's than expected from a marker. This happens when either the received word preceding the marker ends with one or more 0's, and/or the received word following the marker begins with one or more 0's. For example, when the decoder finds a run of $M+2$ zeros, it is often not possible to tell whether each codeword owns one 0 (and the remaining marker pattern is $M$ 0's), or two 0's (and the remaining marker pattern is $M-2$ 0's), etc. To solve this problem, we make the engineering decision that the decoder will always declare the first $M$ bits as the marker, and pass the rest of 0's to be the leading bits of the following codeword. (We tried various techniques and tests to exhaust the cases due to this special case and have found no better alternative.)

It is possible that the second codeword receives extra zero bit(s). The decoder fixes this by checking all such received codewords that follow a full marker ($M$ zeros) and have lengths $c \geq n$ bits. The leading consecutive zero bits, up to a total of $c-(n-1)$ bits, are deleted. Note that for the case of a $c = n$-bit codeword, the decoder will always delete the leading bit if it is a 0. This is acceptable because with $n-1$ bits the VT decoder will always be able to recover the codeword.

The second special case occurs when a marker cannot be detected because too many bits have been deleted. For the $M$-bit marker, our algorithm would not have detected any pattern of $\ell_{min}-1$ (or less) zeros as a marker. In this (unlikely) case, the codeword that the marker decoder initially retrieves is longer than $n$ bits because it consists of two original codewords and the remaining bits of the marker pattern in between them. We apply an iterative splitting algorithm to extract the original codewords from the long codeword. Denote the position of the middle bit (or one of the bits nearest to the middle) of the long codeword by $i$. We look for the pattern of $\ell_{min}-1$ zeros in the region bounded by the $i-3$th and the $i+3$th location. Once we find this pattern, we assign the bits preceding and following the pattern to two codewords. If we fail to find it, we change the pattern to $\ell_{min}-2$ zeros and start over, etc. In the end, if we fail to split the long codeword into 2 codewords, we simply assign probabilities of 1/2 to all the bits in both codewords and proceed. An algorithm execution example is shown in Figure 4.

We believe our simple marker detection scheme could be improved, which should yield noticeable improvements in performance. This remains a point of future work.

### D. Outer Code Construction

The fact that the inner decoder obtains reliable probabilities and the near-capacity performance of LDPC codes [18] motivate the choice of soft outer LDPC decoder. In this manuscript we implement binary LDPC codes. We note that LDPC codes over GF($2^k$) would enhance the performance of our scheme, since these codes implicitly exploit the dependencies of the probabilities $p_i$, $1 \leq i \leq k$. However, the decoders of these codes are more complex and are practical only for very small values of $k$.

The choice of probabilities at the output of the VT

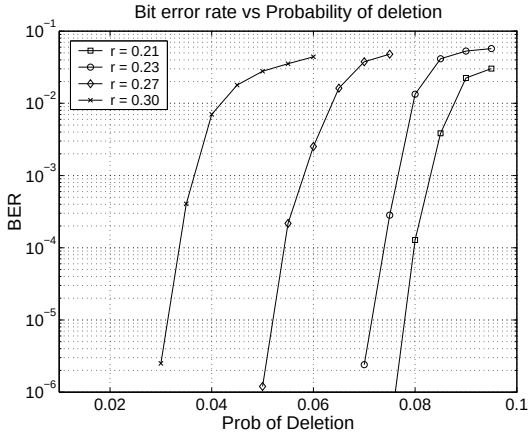


Fig. 5. Bit error rate vs Probability of deletion for marker length $M=5$.

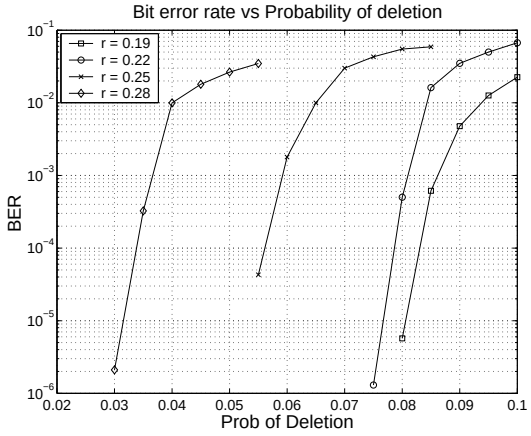


Fig. 6. Bit error rate vs Probability of deletion for marker length $M=6$.

decoder affects the LDPC decoder performance. A potential problem is that the marker decoder may (rarely) incorrectly report a single deletion or no deletions to the VT decoder. Hence the VT decoder should never output a soft probability of 0 or 1 to the LDPC decoder, since it might force the LDPC decoder to fail on an erroneous bit. The proper probabilities are very close to 0 or 1, and could be estimated by simulation to determine the exact probability of this type of failure in the marker code. As an engineering decision based on simulation experiments, we have found that using 0.01 and 0.99 is very effective; while this undoubtedly overestimates the probability of marker failure, it avoids the problem of making a bit too difficult to change for the LDPC decoder. This remains a point for future study.

## IV. Simulation Results

In this section we demonstrate the performance of our codes with two examples. In both examples the marker bits are inserted after every VT codeword in the encoded bitstream. The length of the VT encoded codeword is $n=10$ bits. The rate of the VT code is $r_{vt}=k/n=1/2$. The length of the short markers is $M=5$ and 6 bits, for the first and second example, respectively. The long (size $L=2M$) markers are inserted after every $B=50$ codewords in both examples. This implies that the rates of the inner code are $r_{in}=0.329$ in the first example and $r_{in}=0.308$ in the second. The rate of the outer LDPC code is varied to obtain appropriate overall rates. For example, in the first example if the outer LDPC code rate is $r_{out}=0.7$ than the overall rate is $r=r_{out}\cdot r_{in}=0.7\times 0.329=0.230$. The maximum possible overall rate is 0.329 and 0.308 in the first and the second example, respectively.

We have opted to use regular LDPC codes in this paper for simplicity. Since the performance of $(3,R)$-regular LDPC codes is often close to optimal performance of regular codes [19], we have implemented our concatenated codes with these codes as outer codes. This restricts the possible LDPC code rates and, accordingly, the overall rates[2]. Here $(3,R)$ denotes the regular codes with the variable node degrees equal to 3 and the check node degrees equal to $R$ [19]. $R$ is an integer chosen so that the rate $(R-3)/R$ is as close as possible to the desired outer code rate $r_{out}$.

Simulation results for these two code examples with the LDPC code block length $N=5000$ are shown in Figures 5 and 6. For medium-to-high probabilities of deletion (above 3%) the results are promising. Our results cannot be directly compared to the results from [5] due to the mismatch in the channel model (we consider the deletion channel model, whereas the results in [5] were obtained for the case when both deletions and insertions are possible). We do note that our code at rate $r=0.21$ achieves block error rate $10^{-3}$ for $P_d=8\%$ and the code from [5] achieves similar block error rate at $r=0.21$ for the probability of deletion $P_d=4\%$ and the probability of insertion $P_i=4\%$. Our decoding algorithm has much lower complexity than the algorithm described in [5], which uses a forward-backward algorithm that becomes memory and time consuming as LDPC block lengths grow large. In comparison our decoder is simpler - synchronization involves finding marker sequences, probabilities are returned with table lookups, and the LDPC decoder takes $2-20$ iterations to recover all the bits. A drawback of our scheme is that it cannot be used for construction of the high rate codes (for very low probabilities of deletions) due to relatively low rate of the inner code. Results for different block lengths of the LDPC codes are shown in Figure 7. From this figure we see that at the code rate $r=0.21$ the probability of a bit error event is less than $10^{-6}$ for the deletion channel with $P_d=8\%$. We also observe from Figure 7 that the performance of the code improves as the code block length $N$ grows larger. This is an expected result since the block length does not affect the marker code performance due

[2]This constraint can be relaxed by use of irregular codes

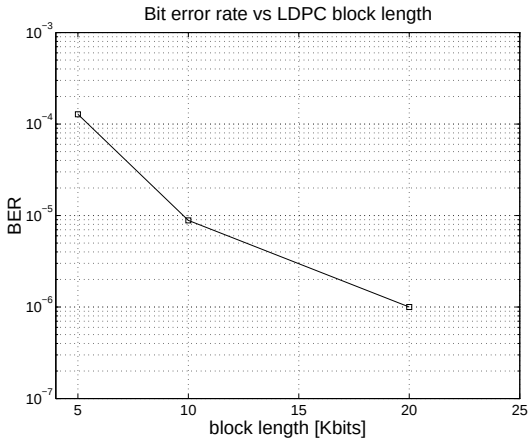


Fig. 7. Bit Error Rate vs LDPC Block length when $P_d = 8\%$ and $r = 0.21$

to the presence of large markers and LPDC codes generally improve as the block length grows. However, this results in higher coding complexity.

While these results are promising, we note that our codes are still far from the lower bound on the capacity of the deletion channel computed in [1]. For example, at the probability of deletion $P_d = 8\%$ the lower bound from [1] stands at $r = 0.598$, whereas our codes achieve low bit error rates at $r = 0.21$ for the same probability of deletion.

## V. Conclusion and Discussion

We have presented concatenated codes for deletion channels. The constituent codes in the concatenated coding scheme are an LDPC code that acts as an outer code and a VT code and a Marker code tied together as an inner code. The decoding method is based on the soft LDPC decoding strategy. The soft information is obtained from the VT decoder. Marker codes are used to recover synchronization. With a regular LDPC code used as an outer code the bit error rate is less than $10^{-6}$ on the channel with the probability of deletion 8% at the code rate 0.21.

There are several possible improvements to the proposed scheme. An obvious method to improve the outer code is to use optimal (or suboptimal) irregular LDPC codes. Although the optimization of the irregular LDPC codes for finite block lengths is an open problem, one can obtain suboptimal codes for large block lengths. This can be done by adapting density evolution [18] in a fashion similar to that done in [14] [15] to this channel model.

There are several unresolved questions relating to the VT decoder. Increasing the length of VT codewords improves the rate of the code at the expense of a greater chance of synchronization errors and less reliable probabilities. It would be instructive to determine the limits of our scheme as it relates to $n$. Another open question involves finding an optimal encoding function $f$, or showing that finding the optimal encoding function is, say, NP-hard. A more general question deals with our measure for encoding functions: how can we measure the quality of probabilities for soft LDPC decoding? The measure we chose is a natural one, but there appears to be little theory about the general question.

Finally, we note that we could similarly use concatenated codes for insertion-deletion channels. We might then desire an inner code that was robust against a single insertion and/or a single deletion. We do not believe the theory of such codes is currently well developed.

## References


[1] S. N. Diggavi and M. Grossglauser, "On transmission over deletion channels," in *ISIT 2001*, July 2001.

[2] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals (in Russian)," in *Doklady Akademii Nauk SSSR, 163 (No. 4, 1965), 845-848.* English translation in Soviet Physics Dokl., 10 (No. 8, 1966), 707-710.

[3] V. I. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones (in Russian)," in *Problemy Peredachi Informatsii, 1 (No. 1, 1965), 12-25.* English translation in Problems of Information Transmission, 1 (No. 1,1965), 8-17.

[4] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors (in Russian)," in *Automatika and Telemekhanika, 26 (No. 2, 1965), 288-292.* English translation in Automation and Remote Control, 26 (No. 2, 1965), 286-290.

[5] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions and substitutions," *IEEE Trans. Inform. Theory*, vol. 47, pp. 687–698, Feb. 2001.

[6] L. J. Schulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," in *Proceedings of the eight annual ACM-SIAM symposium on Discrete algorithms 1997, 669-674.*, (New Orleans, Louisiana).

[7] P. A. H. Bours, "Construction of fixed-length insertion/deletion correcting runlength-limited code," *IEEE Trans. Inform. Theory*, vol. 40(6), pp. 1841–1856, November 1994.

[8] R. G. Gallager, *Low-Density Parity-Check Codes.* Cambridge, MA: MIT Press, 1962.

[9] E. A. Ratzer and D. J. C. MacKay, "Codes for channels with insertions, deletions and substitutions," in *the 2nd International Symposium on Turbo Codes and Applications*, (Brest, France), September 2000.

[10] F. F. Sellers Jr., "Bit loss and gain correction code," *IRE Trans. Inform. Theory*, vol. 8(1), pp. 35–38, Jan. 1962.

[11] T. Richardson and R. Urbanke, "Multi-edge type low-density parity-check codes." In the night session of IEEE Int. Symp. on Information Theory, July 2002.

[12] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low density parity check codes using irregular graphs and belief propagation," in *Proc. IEEE Int. Symp. Inform. Theory*, (Cambridge, MA), p. 117, Aug. 1998.

[13] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, February 2001.

[14] S.-Y. Chung, G. D. Forney, Jr., T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 47, pp. 58–60, February 2001.

[15] N. Varnica and A. Kavčić, "Optimized LDPC codes for partial response channels," in *in the Proceedings of the IEEE Int. Symp. on Information Theory*, (Lausanne, Switzerland), July 2002.

[16] G. D. Forney Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47(2), pp. 520–548, February 2001.

[17] N. J. A. Sloane, "On single-deletion-correcting codes," in *D. Ray-Chaudhuri Festschrift*, 2001. available at http://www.research.att.com/njas/doc/dijen.ps.

[18] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, February 2001.

[19] D. J. C. MacKay, "Good error-correcting codes based on very sparce matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, March 1999.