It remains to analyze the complexity of the coding scheme. By Theorem 3.1, the time needed for the encoding of each B_i is $O(d^3)$ and for the decoding is $O(d^2)$. Thus the overall encoding time on $q = ((1-\beta)n-l)/d$ segments is $O(d^2((1-\beta)n-l))$ and the overall decoding time is $O(d((1-\beta)n-l))$. Similarly, we can show that the time needed for the encoding of segment I is $O(l^2n)$ and for the decoding is $O(ln)$. Thus the overall encoding time at this recursive level (excluding the time needed for the recursion on segment R) is $O(d^2((1-\beta)n-l)+l^2n)$ and the overall decoding time is $O(d((1-\beta)n-l)+ln)$.

Now let us denote $T_E(n)$ (resp., $T_D(n)$) to be the overall time needed for the encoding (resp., decoding) of a segment of length n. Then we have the following recurrent equations:

$$T_E(n) = T_E(\beta n) + O(d^2((1-\beta)n-l)+l^2n)$$
$$T_D(n) = T_D(\beta n) + O(d((1-\beta)n-l)+ln).$$

Solving the recurrences, we have that $T_E(n) = O(d^2n)$ and $T_D(n) = O(dn)$. Indeed, for instance, for the first equation, we have

$$\begin{aligned} T_E(n) &= O(d^2\beta n) + O(d^2((1-\beta)n-l)+l^2n) \\ &= O(d^2n + l^2n - ld^2) = O(d^2n) \end{aligned}$$

since by (8)

$$l < \log\frac{2}{\beta} < d.$$

This completes the proof. □

IV. Concluding Remarks

This correspondence introduces binary codes correcting t localized erasures. The redundancy of codes is $t(1+\epsilon)$ and is thus very close to the minimal possible. The complexity of the simplest construction is linear in n and inversely proportional to ϵ^2. The number ϵ can take almost any value in $(0, 1)$, contrary to some other recursive constructions in which it has to be small, which results in high encoding/decoding complexity. We note that it is possible to construct almost optimal low-complexity binary codes that correct localized errors and erasures at the same time. This could be the subject of a future work. Another interesting problem would be to construct low-complexity codes correcting t localized erasures with redundancy $t + o(n)$ as for codes correcting defects [7].

References

[1] R. Ahlswede, L. A. Bassalygo, and M. S. Pinsker, "Asymptotically optimal binary codes of polynomial complexity correcting localized errors," *Probl. Pered. Inform.*, vol. 31, no. 2, pp. 76–83, in Russian, and *Probl. Inform. Transm.*, pp. 162–168, 1995, English translation.

[2] N. Alon and M. Luby, "A linear time erasure resilient code with nearly optimal recovery," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1732–1736, Nov. 1996.

[3] C. A. Asmuth and G. R. Blakley, "Pooling, splitting and restituting information to overcome total failure of some channels of communication," in *Proc. 1992 Symp. Security and Privacy* (IEEE, New York, 1982), pp. 156–169.

[4] A. Barg, "Complexity issues in coding theory," in *Handbook of Coding Theory*, V. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998, pp. 649–754.

[5] L. A. Bassalygo, S. I. Gelfand, and M. S. Pinsker, "Codes for channels with localized errors," in *Proc. 4th Swedish–Soviet Workshop Information Theory*, 1989, pp. 95–99.

[6] I. I. Dumer, "Asymptotically optimal codes correcting memory defects of fixed multiplicity," *Probl. Pered. Inform.*, vol. 25, no. 4, pp. 3–10, in Russian, and *Probl. Inform. Transm.*, pp. 259–264, 1989, English translation.

[7] ____, "Asymptotically optimal linear codes correcting defects of linearly increasing multiplicity," *Probl. Pered. Inform.* vol. 26, no. 2, pp. 3–17, in Russian, and *Probl. Inform. Transm.*, pp. 93–104, 1990, English translation.

[8] G. A. Kabatianskii and E. A. Krouk, "Coding decreases delay of messages in networks," in *Proc. IEEE Int. Symp. Inform. Theory* (San Antonio, TX, 1993), p. 255.

[9] A. V. Kuznetsov and B. S. Tsybakov, "Coding for memory with defective cells," *Probl. Pered. Inform.*, vol. 10, no. 2, pp. 52–60, in Russian, and *Probl. Inform. Transm.*, pp. 132–138, 1974, English translation.

[10] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. 29th ACM Annual Symp. Theory of Computing (STOC'97)*, ACM, 1997, pp. 150–159.

[11] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* Amsterdam, The Netherlands: North-Holland, 1977.

[12] A. J. McAuley, "Reliable broadband communication using a burst erasure correcting code," in *Proc. SIGCOMM'90.* Philadelphia, PA: ACM, 1990, pp. 287–306.

[13] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. Assoc. Comput. Mach.*, vol. 36, no. 2, 1989, pp. 335–348.

[14] D. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1723–1731, Nov. 1996.

[15] R. Urbanke and A. Wyner, "Packetizing for the erasure broadcast channel with an Internet application," 1997, preprint.

Asymptotically Good Codes Correcting Insertions, Deletions, and Transpositions

Leonard J. Schulman and David Zuckerman

***Abstract*—We present simple, polynomial-time encodable and decodable codes which are asymptotically good for channels allowing insertions, deletions, and transpositions. As a corollary, they achieve exponential error probability in a stochastic model of insertion–deletion.**

***Index Terms*— Asymptotically good, asynchronous communication, deletion, edit distance, error-correcting codes, insertion, transposition.**

I. Introduction

In an asynchronous noisy channel, characters of the received message are not definitively identified with antecedents in the transmitted message. We describe a code which allows for correction of data modified in the following ways:

A Insertion and deletion of characters. (Note that this implies also alteration of characters.)

Manuscript received April 13, 1997; revised October 16, 1998. This work was supported in part by NSF NYI under Grant CCR-9457799, a David and Lucile Packard Fellowship for Science and Engineering, and an Alfred P. Sloan Research Fellowship.

L. J. Schulman is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280 USA (e-mail: schulman@cc.gatech.edu).

D. Zuckerman is with the Computer Science Division, University of California at Berkeley, Berkeley, CA 94720 USA, on leave from the University of Texas at Austin (e-mail: diz@cs.utexas.edu).

Communicated by A. Vardy, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(99)07308-3.

B Transpositions of blocks of data: a message of the form ABC is transformed into ACB. (Note that this implies also general transpositions, i.e., $ABCDE$ transforming to $ADCBE$.)

Our code encodes n bits of information in codewords of length n/r, for some positive constant r called the *rate* of the code. The code corrects up to $e_A n$ errors of type A and $e_B n/\log n$ errors of type B, for certain positive constants e_A, e_B. This result is, up to the values of the constants, best possible; thus we refer to it as an "asymptotically good" code for this error model. This is the first constructive code of this type. The code can be encoded and decoded in polynomial time up to its designed distance. Reversals of segments of the codeword can also be accommodated using the methods described.

This is a generalization of the constructive, asymptotically good codes for the Hamming distance given by Justesen [9]. Those codes could correct only alterations of characters, whereas here we allow more general errors.

Channels with insertions and deletions occur in various situations, for example:

- Insertion and deletion errors occur in reading magnetic and optical media (in addition to the more familiar character-alteration errors). This was the motivation for considering insertions and deletions in [17] and [4].
- If the error-correcting code employed in a digital communication system is designed for a synchronous model (i.e., one without insertions or deletions) then occasional synchronization pulses must be transmitted over the channel. It is likely that the best rate for such a channel is instead achieved by directly designing a code that allows for timing uncertainties in the statistics of the channel. This was the motivation for considering insertions and deletions in [7].
- In a medium with only occasional transmissions (e.g., radio) it may not be apparent whether a noise burst has obscured transmissions.
- Genetic material undergoes just such transformations between generations. It is possible that some of the complex mechanisms of, say, protein production serve to protect the functionality (the phenotype) from such changes in the genotype. Here the possibility of transpositions is also significant.
- In Internet protocols, long messages are commonly split into small packets, each of which is routed separately, and some of which may be lost, on the way to the Internet recipient. The end client, however, may be linked to the Internet recipient via an unreliable channel such as a telephone line. Currently, coding for these two stages is handled separately (and the first stage is usually not coded at all, but interest in such coding appears to be growing, especially for real-time applications [1], [2]); the channel as a whole, however, is of the type considered in this correspondence, and it may be possible to improve transmission rates by coding for the entire process. The ability to handle transpositions is essential in this example, since the order of transmission of the packets is lost due to their separate routing.

For a discussion of these and other application areas, along with related algorithms, see the survey by Kruskal [10].

Codes for insertion and deletion errors were first considered in 1965 by Levenshtein [12]. He obtained bounds on the number of codewords possible for correcting any constant number (not fraction) of errors in a block, and suggested the use of buffers between codewords in an extended transmission. These and later bounds [13] on the volume of metric balls imply the existence of exponentially large families of codewords, and therefore that asymptotically good codes for insertion and deletion errors exist. A series of papers has followed, developing codes for such channels, especially on account of their occurrence in magnetic and optical media. These codes, as with other codes for such media, employ "run-length limited" codes, in which the length of any maximal run of 0's is bounded below by some d and above by some k. As is most relevant for the media considered, these codes correct insertions and deletions only of 0's.

Ours is the first construction of asymptotically good codes for deletion/insertion channels. Our purpose is to present these codes and accompanying encoding/decoding algorithms as simply as possible, without optimization of rate or computational overhead for particular applications or ranges of parameters.

It is not difficult to use Justesen codes to construct (d, k)-constrained codes with constant rate correcting a small enough constant fraction of insertions and deletions of 0's. Namely, associate the Justesen codeword $(\alpha_1, \alpha_2, \cdots, \alpha_l)$ over an alphabet of size $k-d+1$ with the (d, k)-constrained codeword $10^{\alpha_1+d}10^{\alpha_2+d}\cdots 10^{\alpha_l+d}$. Let $k = O(1)$ (otherwise the constrained code cannot have constant rate). If the original codeword has a small enough constant fraction of errors, then a small enough constant fraction of the α_i are modified, so the Justesen code can be used to recover the original word.

Bours [4], following on Roth and Siegel [17], improved the constants above by constructing fixed-length (d, k)-constrained codes using the more appropriate Lee metric. In the Lee metric, the distance between digits i and j modulo q is

$$\min\left((j-i) \bmod q, (i-j) \bmod q\right)$$

(where it is understood that the modular representatives are in the range $0, \cdots, q-1$), as compared to 1 if $i \neq j$ in the Hamming metric.

Some other works giving constructions, or bounds, for codes for insertions and deletions are those of Ullman [21], Calabi and Hartnett [5], Okuda, Tanaka, and Kasai [15], Tanaka and Kasai [19], Tenengolts [20] (single error), Levenshtein [13], and Kløve [11] (single error which may be either a transposition of adjacent characters or an insertion or deletion of a 0).

Along different lines, there are also essentially optimal codes for lossy packet-based channels such as the Internet [1], [2]. Those codes handle only deletion of complete packets (a packet is a character from a very large alphabet, e.g., of order 2^{50}, and decoding cannot rely on order of packet arrival).

Gallager [7] and Dobrushin [6] discussed stochastic models of insertion and deletion errors. Gallager showed how the random convolutional coding method of Wozencraft and Reiffen [23], [16] could be adapted to this situation. (Note that this method is not a code but a probability distribution over codes; successful transmission requires that the transmitter and receiver share a random seed identifying the code to be used.) Dobrushin proved existence of a channel capacity for a fairly broad class of memoryless stochastic channels with insertions and deletions. A slightly different model, allowing a restricted kind of channel memory, will be described in Section IV; our codes provide for block coding with exponential error probabilities, for channels in this class (subject to limits on the error probabilities).

We emphasize that our codes allow arbitrary insertions, deletions, and transpositions, subject only to numerical limits; the errors do not have to be of restricted types, or distributed randomly.

II. The Code

Our code, like a Justesen code [9], is a two-level code. The outer level can be given by polynomial evaluation, i.e., a Reed–Solomon code [14] (and decoded using the Welch–Berlekamp algorithm [22], see also [8] and [3]); or by any asymptotically good, efficiently

encodable and decodable code. The inner level is given by a code which we find by brute force (e.g., by a "greedy" construction). Since we will only use this code on words of length $\log n$, its construction, as well as encoding and decoding, will require only polynomial time. We first describe these ingredients, and then describe the encoding and decoding procedures of our code.

We describe two variants of our code, a "buffered" form in Section II-D and an "unbuffered" form in Section II-E.

A. Integrality Constraints and Notation

Throughout the exposition we ignore round-off errors, assuming when needed that a number is an integer. It is not hard to see that this does not affect our analysis. We also assume, when needed, that n is sufficiently large, so that, e.g., $\lg n$ is bigger than some fixed constant; and that n is a power of a prime. Thus there is a finite field on n elements, which can be constructed in time polynomial in n, and whose arithmetic operations may be performed in time polynomial in $\log n$.

We use $[m]$ to denote the set of integers $\{1, 2, \cdots, m\}$, and $\lg$ to denote the logarithm to the base 2. By an interval of a string y we mean a contiguous subsequence of the string, $y_i y_{i+1} \cdots y_j$.

B. The Outer Code

The outer code $T\colon \{0,1\}^n \to (\{0,1\}^{2\lg n})^{c_0 n/\lg n}$ outputs a sequence of blocks in $\{0,1\}^{2\lg n}$ (actually, the blocks will be of length slightly smaller than $2\lg n$; this is not important, and one could always pad). We sometimes think of T in the equivalent form $T\colon \{0,1\}^n \times [c_0 n/\lg n] \to \{0,1\}^{2\lg n}$. When $T(x)$ is transmitted, the order of the blocks is scrambled, and errors may occur. By an error, we mean either a received block that is not a block of $T(x)$, or a block of $T(x)$ that was not received. The decoder of the outer code thus receives a set of blocks in $\{0,1\}^{2\lg n}$, in no particular order. The decoder has the property that, if there are at most $n/\lg n$ errors, then x can be efficiently determined. We mention two means of accomplishing this.

1) Reed–Solomon Codes: The first method (which we will adhere to in the rest of the correspondence) uses Reed–Solomon codes and Welch–Berlekamp decoding. First partition $x \in \{0,1\}^n$ into blocks $g_1, \cdots, g_d$, each of length $\lg n$, where $d = \frac{n}{\lg n}$. Regard these as the coefficients of a degree $d-1$ polynomial g over GF(n). Set

$$T(x, i) = i \circ g(i) \text{ (here } \circ \text{ denotes concatenation),} \quad \text{for} \quad 1 \le i \le c_0 n/\lg n.$$

We call these indexed Reed–Solomon codes.

Decoding relies on the following lemma, which is the essence of the Welch–Berlekamp decoder [3], [8], [22]:

Lemma 1: Let F be a field with efficiently implementable arithmetic operations, and assume t, d, and κ are nonnegative integers such that $2t + d < \kappa$. Given κ points $(x_i, y_i) \in F^2$, there is an algorithm which finds a degree d polynomial g such that $g(x_i) = y_i$ for all but t values of i, if such a g exists (in which case it is unique). The running time of the algorithm is polynomial in κ, dominated by the time to invert a $\kappa \times \kappa$ matrix over F.

It follows that with $c_0 = 3$, for example, the code T can tolerate a constant fraction of incorrectly received pairs $i \circ g(i)$.

2) Linear-Time Codes: We can improve the running time of our decoder, at the expense of worse rate constants, by using Spielman's linear-time codes [18].

In fact, we can base our work on any asymptotically good and computationally efficient code $C\colon \{0,1\}^n \to \{0,1\}^{c_0 n}$. To do this, we divide the output of C into $c_0 n/\lg n$ blocks of length $\lg n$, i.e., we convert C to a function $C\colon \{0,1\}^n \times [c_0 n/\lg n] \to \{0,1\}^{\lg n}$. Then we can set $T(x, i) = i \circ C(x, i)$. It is straightforward to verify that this satisfies the required decoding condition for a sufficiently large constant c_0.

C. Greedily Constructed Codes

We will use the following metric in this correspondence: d_A, the insertion–deletion distance, is the minimum number of insertions or deletions (i.e., type A errors) required to transform one string into another. (For strings of the same length this is equivalent to the minimum total number of deletions, or the minimum total number of insertions, to convert the two strings into a common string [12].)

We will use two slightly different greedily constructed codes S_1 (used in the buffered codes) and S_2 (used in the unbuffered codes). S_1 is a function from $\{0,1\}^{t/2} \to \{0,1\}^{2t}$ (where t will be $\Theta(\lg n)$). We will guarantee a somewhat lower rate for S_2: for some constant c it is a function from $\{0,1\}^{ct} \to \{0,1\}^{t}$.

Code S_1 satisfies "condition 1:" the d_A distance between any two codewords is $\Omega(t)$. Furthermore, every interval (of even length) has at least half 1's.

Code S_2 satisfies the stronger "condition 2:" for any two codewords $u \neq v$, the d_A distance between any two intervals in u and v, each of length at least $t/5$, is more than $t/30$.

There is a greedy algorithm to construct a code of type 1: pick a codeword w_1, then pick a codeword w_2 that is far from w_1, then a w_3 that is far from both w_1 and w_2, etc. This algorithm runs in time $2^{O(t)}$. A code of type 2 can be constructed similarly, ensuring that each sufficiently long interval is far from all intervals in previously selected codewords.

Lemma 2: The greedy algorithm can construct codes of types 1 and 2.

Proof:

Type 1: The number of words in $\{0,1\}^t$ that are within d_A-distance $2d$ of a particular word w in $\{0,1\}^t$ is at most $\binom{t}{d}^2 2^d$. To see this, note that to go from w to say $v \in \{0,1\}^t$ using distance $2d$ entails d deletions and d insertions. There are $\binom{t}{d}$ possible characters in w to delete. Then there are $\binom{t}{t-d}$ ways to place the remaining characters in proper positions in v, and 2^d possibilities for the inserted characters in v.

We now choose d to be a small enough constant times t, which makes $\binom{t}{d}^2 2^d \le 2^{t/2}$.

We ensure that every interval has half 1's by inserting 1's into every other position. That is, if G is the code we have constructed greedily, and if we define

$$N(x_1 x_2 \cdots x_n) = 1x_1 1x_2 \cdots 1x_n$$

then the small code we use is $S(x) = N(G(x))$. Note that

$$d_A(N(x), N(y)) \ge d_A(x, y).$$

Type 2: For any interval z of length at least $t/5$, we upper-bound the number of words w in $\{0,1\}^t$ that possess an interval z' closer than d_A-distance $2d$ to z. By multiplying our upper bound by t^2, we can fix the starting positions of the intervals z and z'. Let $D(x)$ denote x with the first character deleted. Since

$$d_A(z, w) \ge d_A(D(z), D(w))$$

by deleting initial characters we may assume that z has length exactly $t/5$.

Let t' be the length of z', which is within $2d$ of $t/5$. We will fix t' and multiply our upper bound by another factor of t. If the d_A distance from z to z' consists of u insertions and d deletions, then $u + v = 2d$ and $u - v = t/5 - t'$. To bound the number of

possibilities for w, note that there are at most $\binom{t/5}{u}$ ways to delete u characters from z, then there are at most $\binom{t'}{v}$ ways to add a particular set of v characters to form the interval z'. We still have to multiply by the number of possible sets of v characters; instead we multiply by the number of ways to fill in characters to get to the full word w, namely, $2^{v+t-t'} = 2^{u+t-t/5}$. Therefore, the desired upper bound is the maximum of

$$t^3 \binom{t/5}{u} \binom{t'}{v} 2^{u+4t/5}$$

over all feasible choices of t' (which determines u and v).

To get the best constants, we would note that increasing t' by 2 increases v by 1 and decreases u by 1, which would show that $u/(t/5)$ should be roughly twice v/t'. Since we have not optimized constants in this correspondence, we can get a quicker upper bound by noting that $u \le 2d$, $t' \le t/5 + 2d$, and $\binom{t/5}{u}\binom{t'}{v}$ is maximized for $5u/t = v/t'$ and hence is at most $\binom{t/5+2d}{d}^2$. For $d = t/100$ this gives the upper bound

$$t^3 2^{(41/50+(11/25)*H(1/22))t} \le 2^{.94t}$$

for large enough t, where H denotes the binary entropy function.

D. Buffered Code

1) Encoding: In order to encode messages in $\{0,1\}^n$ against errors of types A and B, we begin with the (greedily constructed) code of type 1, S_1: $\{0,1\}^{2\lg n} \to \{0,1\}^{8\lg n}$. The minimum insertion–deletion distance between codewords is at least $\delta \lg n$ for some $\delta > 0$. Such a code can be constructed in polynomial time in n. Encoding of codewords, and decoding up to the designed distance, can also be performed in polynomial time.

Our code R_1: $\{0,1\}^n \to \{0,1\}^{cn}$ is defined as follows:

$$R_1(x) = S_1(T(x,1))0^{\lg n} S_1(T(x,2))0^{\lg n} \cdots 0^{\lg n} S_1(T(x,k))$$

where $0^{\lg n}$ denotes 0 repeated $\lg n$ times, and k is $3n/\lg n$. Note also that $c = |R_1(x)|/|x| = 27$ if an outer indexed Reed–Solomon code is used.

2) Decoding:

Theorem 1: Let $e_A = \delta/96$ and $e_B = 1/8$. Let y be the received string. Suppose the number of errors of type A is at most $e_A n$, and the number of errors of type B is at most $e_B n/\lg n$. Then x is determined by y, and can be computed in time polynomial in n.

(Note: For Hamming distance the number of errors is always entirely defined from x and y, but here that is not the case. What we use is that there exists some sequence of at most $e_A n$ errors of type A and $e_B n/\lg n$ errors of type B converting x to y.)

Proof: We begin the decoding process by attempting to determine the original buffers of 0's. To this end, we search intervals of length $\lg n$ from left to right until we first find one that contains at most $\delta/24$ fraction of 1's. We assume this is the buffer: we mark the left and right endpoints, and continue searching intervals of length $\lg n$, starting with the left endpoint of the first new interval at the right endpoint of the presumed buffer.

We then look at all the words in between the presumed buffers, and collect those within insertion–deletion distance $\delta \lg n$ of a codeword of S_1. We then obtain these close codewords $z_1, \cdots, z_\ell$, and invert S_1 on each z_i. Thus we have a collection $\mathcal{C}$ of blocks, and we use the decoder for the outer code to determine the original word.

Correctness: First, an intuitive description. Call $S_1(T(x,i))$ block i, and the buffer after it buffer i. Note that if not too many A errors, and no B errors, occur in both block i and buffer i, then buffer i will be located approximately correctly. Thus if not too many errors occur in blocks and buffers $i-1$ and i, then the buffers surrounding block i are determined approximately correctly, and block i will appear close to a codeword, and will be decoded properly.

To make this rigorous, we give precise meanings to the above terms. By not too many A errors in a block or buffer, we mean at most $\delta \lg n/24$ errors. By located approximately correctly, we mean within $\delta \lg n/6$ places. By determined approximately correctly, we mean to within less than $\delta \lg n/2$ errors.

Note that more than $\delta \lg n/24$ errors can corrupt one block or buffer, which can mean that two blocks are improperly decoded. Hence, $e_A n$ errors can lead to fewer than $2e_A n/(\delta \lg n/24) = n/(2\lg n)$ errors in the collection of blocks $\mathcal{C}$. We also must take into account that totally new blocks and buffers can be created with enough insertions. But such insertions are less efficient at corrupting the code: $\lg n$ insertions are required to create a new buffer, which is more than the $\delta \lg n/24$ needed above.

Note that a transposition error affects up to two blocks or buffers. Each block or buffer corrupted can mean that two blocks are decoded improperly. Thus each transposition error can cause at most four blocks to be improperly decoded. Hence the transposition errors contribute to at most $4e_B n/\lg n = n/(2\lg n)$ errors in $\mathcal{C}$.

Hence fewer than $n/\lg n$ errors are made in $\mathcal{C}$. Of these, say that τ introduce a pair $T(x,i) = i \circ g(i)$ where i appears only once in the decoded list; while the remaining at most $n/\lg n - \tau - 1$ values of i coincide with some other (possibly correct) pair. Such duplicate, inconsistent pairs are discarded before applying Lemma 1. In the notation of that Lemma, $\kappa \ge 3n/\lg n - 2(n/\lg n - \tau - 1)$, $t = \tau$, and $d = n/\lg n$. Now $\kappa - (2t + d) \ge 1$. So by the decoding property of the outer code, i.e., Lemma 1, x can be determined.

3) Improving the Computational Efficiency: We can improve the efficiency of the algorithm, at some cost to the rate of the code, by recursing. That is, we can use our code in place of the greedy code. The decoding of the inner (now "middle") codes will then require only time $n \lg^{O(1)} n$. By recursing j times, we can reduce the time to $n(\log_{\#j} n)^{O(1)}$, where $\log_{\#j}$ denotes the logarithm iterated j times. Using the outer code based on Spielman's error-correcting code then gives an overall running time of $n(\log_{\#j} n)^{O(1)}$. In practice, it is unlikely to be desirable to recurse more than once (i.e., to use $j > 1$).

E. Unbuffered Code

1) Encoding: In order to encode messages in $\{0,1\}^n$ against errors of types A and B, we begin with a code

$$S_2\colon \{0,1\}^{2\lg n} \to \{0,1\}^{c_1 \lg n}$$

of type 2. As noted, such a code can be constructed in time polynomial in n. Encoding of codewords, as well as decoding up to the designed distance, can also be performed in polynomial time. We will take the outer code T to be an indexed Reed–Solomon code; by changing the constants in the definition of the greedy code we can also base it on Spielman's linear-time code.

Our code R_2: $\{0,1\}^n \to \{0,1\}^{cn}$ is as follows:

$$R_2(x) = (S_2(T(x,1)), \cdots, S_2(T(x,k)))$$

where k is $4n/\lg n$. Note also that $c = |R_2(x)|/|x| = 4c_1$.

2) Decoding:

Theorem 2: Let $R_2(x)$ be the transmitted codeword, and let y be the received string. For $e_A = 0.01$ and $e_B = 0.2$, let there be a sequence of at most $e_A n$ insertions or deletions, and $e_B n/\lg n$ transpositions, transforming $R_2(x)$ to y. Then x is determined by y, and can be computed in time polynomial in n.

Proof: Set $b = 1/100$ and $a = (1/5) + (1/100) = 0.21$.

We begin the decoding process by determining, for every interval y' of length at most $(1+b)c_1 \lg n$, whether there is a codeword z of S_2 such that $d_A(z, y') < bc_1 \lg n$. We call such a codeword *close* to y'.

Note that two intervals which overlap in more than $ac_1 \lg n$ characters cannot be close to different codewords. For, this would imply that the overlap was at distance at most $bc_1 \lg n$ to some interval of length at least $(a-b)c_1 \lg n = (c_1/5)\lg n$ in each codeword, and hence that those intervals were at distance at most $(2bc_1)\lg n$ from each other, which violates the distance condition $(c_1/50)\lg n$ of the code S_2.

Every codeword that is close to some interval of y can therefore be regarded as the sole owner of a segment of y beginning at most $(ac_1/2)\lg n$ after the start of the interval that is close to the codeword, and ending at least $(ac_1/2)\lg n$ before the end of that interval. Hence the segment is of length at least $(1-a-b)c_1 \lg n$; note further that the received string y is of length at most $(c+e_A)n$. It follows that the number of codewords that are close to some interval y' is at most

$$\frac{(c+e_A)n}{(1-a-b)c_1 \lg n} = \frac{(4c_1+e_A)k}{4(1-a-b)c_1}.$$

Suppose that some sequence of at most $e_A n$ insertions or deletions, and $e_B n/\lg n$ transpositions, transforms $R_2(x)$ to y. Then at most fraction $e_B/2$ codewords of S_2 are ever bisected by a transposition boundary. Also, at most a fraction

$$e_A n/((c_1 \lg n/100)(4n/\lg n)) = 25e_A/c_1$$

codewords can be affected by more than $c_1 \lg n/100$ insertions or deletions. Hence at least $(1-(1/2)e_B - 25e_A/c_1)k$ codewords of S_2 are correctly decoded.

Each decoded word yields a pair $(i, g(i))$. If some i occurs more than once (which can happen only if all but one are erroneous decodings) then all such pairs, say h of them, are discarded; at least $h/2$ of these are erroneous decodings. The remaining decoded pairs are submitted to a decoding algorithm for the Reed–Solomon code.

The Reed–Solomon decoding algorithm is thus provided with at most $\kappa = \frac{(4c_1+e_A)k}{4(1-a-b)c_1} - h$ pairs, of which at least

$$\left(1 - \frac{1}{2}e_B - 25e_A/c_1\right)k - h/2$$

derive from the message, or in other words, of which at most

$$t = \frac{(4c_1+e_A)k}{4(1-a-b)c_1} - \left(1 - \frac{1}{2}e_B - 25e_A/c_1\right)k - h/2$$

are erroneous. Recall that the degree of the polynomial in the indexed Reed–Solomon code is $d = n/\lg n = k/4$.

In order to guarantee success of the Reed–Solomon decoding procedure we show that the hypothesis of Lemma 1, $2t + d < \kappa$, is satisfied

$$2t + d \le 2\left(\frac{(4c_1+e_A)k}{4(1-a-b)c_1} - \left(1 - \frac{1}{2}e_B - 25e_A/c_1\right)k - h/2\right) + k/4.$$

Noting that $c_1 \ge 2$ gives

$$\begin{aligned} 2t + d &\le \frac{(4c_1+e_A)k}{4(1-a-b)c_1} + \frac{(8+e_A)k}{8(1-a-b)} \\ &\quad - 2\left[\left(1 - \frac{1}{2}e_B - 25e_A/2\right)k + h/2\right] + k/4 \\ &= \frac{(4c_1+e_A)k}{4(1-a-b)c_1} - h + \frac{(8+e_A)k}{8(1-a-b)} \\ &\quad + k/4 - 2k(1 - \tfrac{1}{2}e_B - 25e_A/2) \\ &= \kappa - k\left(\frac{7}{4} - 25e_A - e_B - \frac{(8+e_A)}{8 \cdot 0.78}\right) \\ &< \kappa - k(0.4679 - 25.5e_A - e_B) \end{aligned}$$

which, for the stated values of e_A and e_B, is less than κ.

III. The Code is Asymptotically Good

It has already been shown that the code has positive rate (in both the buffered and unbuffered cases). It remains to argue that it is optimal up to constant factors, namely, that no code of positive rate can tolerate more than $O(n/\lg n)$ transposition errors. We show that $(1+o(1))n/\lg n$ is an upper bound on the number of transpositions that can be corrected (here n denotes the length of the codeword). Suppose that $\varepsilon > 0$ and that a code can correct $d = n/((1-\varepsilon)\lg n)$ transpositions. Consider any string $x \in \{0,1\}^n$. Parse x into d blocks, each of length $(1-\varepsilon)\lg n$. A sequence of d transpositions now suffices to rearrange these blocks in lexicographic order. The only information retained about x is the frequency of occurrence of each block, so no two codewords can share their frequency list. Since there are less than $n^{n^{1-\varepsilon}} = 2^{n^{1-\varepsilon}\lg n}$ possible lists of frequencies, that is an upper bound on the number of codewords.

IV. Stochastic Channel Model

In this section we propose a relatively simple model of a probabilistic asynchronous channel. (We will allow only insertions and deletions here, not block transpositions.) It appears to be a difficult problem to analyze the capacity of such a channel even in the case that only deletions are allowed. Since our codes are "asymptotically good" they achieve exponentially small error probability on such stochastic channels, provided the error rates are below certain constants. We do not know how to code for insertion–deletion channels of arbitrary nonzero capacity.

The Model: We restrict ourselves to discrete channels, with alphabet $0, 1$ (for both input and output). It is not hard to generalize the model to larger alphabets. For a string x, let $\xi(x)$ be the random variable which is the output of the channel on input x. For string y and for $r \ge 0$, let $y_{r\setminus}$ be the string consisting of the first r characters of y; similarly let $y_{\setminus r}$ be the string consisting of the last r characters of y.

Let $p = \{p_i\}_{i=0}^{\infty}$ and $q = \{q_i\}_{i=1}^{\infty}$ be probability distributions with finite first and second moments. Let $(a_j^i)_{0 \le i,\, j \le 1}$ be a stochastic matrix, with $a_j^i = P$ (output $j|$ input i). Let $b = \{b_j\}_{j=0}^{1}$ be a probability distribution representing a certain "background noise."

The noise model is described by the following process. The input x is written on cells $1, \cdots, |x|$ of an input tape. A "read head" is initially located at cell 0 of the tape. An output tape is provided on which a "write head" starts at cell 0. In each step of the process, variables r and w are chosen independently, r from distribution p and w from distribution q. The read head then shifts r cells while the write head shifts w cells. In the intermediate $w-1$ cells of the output tape, independently chosen characters from probability distribution b are written. Then a character j is selected with probability distribution

a^i_j, where i is the character under the read head; j is written to the cell currently under the write head.

Observe that the case $p_1 = 1$ corresponds to a channel with no deletions; while the $q_1 = 1$ case corresponds to a channel with no insertions. When both $p_1 = 1$ and $q_1 = 1$ we have a synchronous channel. If $p_0 = 0$ then there is no "stutter," i.e., every input symbol is represented in at most one output symbol.

Gallager [7] has discussed a different stochastic model of a channel with insertions and deletions. In that model there are four fixed parameters p_e, p_d, p_i, and p_c, summing to 1. Each character of the codeword is independently affected in the following way: with probability p_e the character is flipped; with probability p_d the character is deleted; with probability p_i two random characters are inserted in place of the character; and with probability p_c the character is conveyed correctly. This work was conducted before the first constructions of asymptotically good codes; however, Gallager showed that, if transmitter and receiver have a shared random sequence, the random convolutional coding method of Wozencraft and Reiffen [23], [16] can be employed to yield computationally efficient sequential decoding for rates below a cutoff rate R_{comp}.

Dobrushin [6] studied a fairly general stochastic model; in terms of the above description, his is obtained by requiring that $p_1 = 1$ (thus the read head never skips or repeats) while allowing the output j to be a word (rather than character) chosen from a distribution $\{a^i_j\}_j$ on $j \in \{0, 1\}^*$ which depends on the input character i. Thus his model is both more general than ours (in allowing an arbitrary distribution on words depending on each input character; we allow only a distribution on characters, though several such characters may be selected due to stutter) and more restricted (in being memoryless, which ours is not due to the distribution on skips and stutters). Of course, a common generalization is obtained simply by allowing j in our definition to range over $\{0, 1\}^*$ rather than $\{0, 1\}$ (in which case one can restrict to $q_1 = 1$). Useful error-correcting codes should perhaps be designed with a less general model in mind, but it would be desirable to know the existence of a channel capacity in this generality.

V. Discussion

The code can be modified to also account for reversals, i.e., the modification of ABC into AB^rC, where B^r is the reverse of B. To do this it suffices to make the inner code resilient against such transformations.

In general, larger alphabet sizes make things easier. For example, an alphabet of size 3 allows us to simplify our buffered codes, as one character may be reserved for use as a buffer character, and an (unmodified) greedily constructed code of type 1 can be used for the inner code.

As mentioned previously, we have not optimized constants for rate, error capacity, or computation.

Acknowledgment

The authors wish to thank the anonymous referees for their helpful comments.

References

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," in *35th Annu. Symp. Foundations of Computer Science*, 1994, pp. 604–612.

[2] N. Alon, J. Edmonds, and M. Luby, "Linear time erasure codes with nearly optimal recovery," in *36th Annu. Symp. Foundations of Computer Science*, 1995, pp. 512–519.

[3] E. R. Berlekamp, "Bounded distance +1 soft-decision Reed–Solomon decoding," *IEEE Trans. Inform. Theory*, vol. 42, pp. 704–720, May 1996.

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

[5] L. Calabi and W. E. Hartnett, "A family of codes for the correction of substitution and synchronization errors," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 102–106, Jan. 1969.

[6] R. L. Dobrushin, "Shannon's theorems for channels with synchronization errors," *Probl. Inform. Transm.*, vol. 3, no. 4, pp. 11–26, 1967 (trans. from *Probl. Pered. Inform.*, vol. 3, no. 4, pp. 18–36, 1967).

[7] R. G. Gallager, "Sequential decoding for binary channels with noise and synchronization errors," Lincoln Lab. Group Rep. 2502, Sept. 1966; unclassified document AD266879, Armed Services Technical Information Agency, Arlington Hall Station, Arlington VA.

[8] P. Gemmell and M. Sudan, "Highly resilient correctors for polynomials," *Inform. Processing Lett.*, vol. 43, no. 4, pp. 169–174, 1992.

[9] J. Justesen, "A class of constructive, asymptotically good algebraic codes," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 652–656, Sept. 1972.

[10] J. B. Kruskal, "An overview of sequence comparison: Time warps, string edits, and macromolecules," *SIAM Rev.*, vol. 25, no. 2, pp. 201–237, Apr. 1983.

[11] T. Kløve, "Codes correcting a single insertion/deletion of a zero or a single peak-shift," *IEEE Trans. Inform. Theory*, vol. 41, pp. 279–283, Jan. 1995.

[12] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys.–Dokl.*, vol. 10, no. 8, pp. 707–710, Feb. 1966 (translated from *Dokl. Akad. Nauk SSSR*, vol. 163, no. 4, pp. 845–848, Aug. 1965).

[13] ——, "On perfect codes in deletion and insertion metric," *Discr. Math. Appl.*, vol. 2, no. 3, pp. 241–258, 1992.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[15] T. Okuda, E. Tanaka, and T. Kasai, "A method for the correction of garbled words based on the Levenshtein metric," *IEEE Trans. Comput.*, vol. C-25, no. 2, pp. 172–178, Feb. 1976.

[16] B. Reiffen, "Sequential encoding and decoding for the discrete memoryless channel," Res. Lab. Electron., MIT, Tech. Rep., vol. 374, 1960.

[17] R. M. Roth and P. H. Siegel, "Lee-metric BCH codes and their application to constrained and partial-response channels," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1083–1096, July 1994.

[18] D. Spielman, "Linear-time encodable and decodable error-correcting codes," in *27th Annu. ACM Symp. Theory of Computing*, 1995, pp. 388–397.

[19] E. Tanaka and T. Kasai, "Synchronization and substitution error-correcting codes for the Levenshtein metric," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 156–162, Mar. 1976.

[20] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 766–769, Sept. 1984.

[21] J. D. Ullman, "On the capabilities of codes to correct synchronization errors," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 95–105, Jan. 1967.

[22] L. Welch and E. R. Berlekamp, "Error Correction of Algebraic Block Codes," U.S. Patent 4 633 470, Dec. 1986.

[23] J. M. Wozencraft, "Sequential decoding for reliable communications," Res. Lab. Electron., MIT, Tech. Rep., vol. 325, 1957.