A Survey of Results for Deletion Channels and Related Synchronization Channels

Michael Mitzenmacher*

Harvard University
School of Engineering and Applied Sciences
michaelm@eecs.harvard.edu

1 Introduction

The binary symmetric channel, where each bit is independently received in error with probability p, and the binary erasure channel, where each bit is erased with probability p, enjoy a long and rich history. Shannon developed the fundamental results on the capacity of such channels in the 1940's [37], and in recent years, through the development and analysis of low-density parity-check (LDPC) codes and related families of codes, we understand how to achieve near-capacity performance for such channels extremely efficiently [4, 25, 34].

Now consider the following channel: n bits are sent, but each bit is independently deleted with fixed probability p. This is the *binary independently and identically distributed (i.i.d.) deletion channel*, which we may refer to more succinctly as the *binary deletion channel* or just the *deletion channel* where the meaning is clear. A deletion channel should not be confused with an erasure channel. With an erasure channel, when n bits are sent, n symbols are received; a third symbol, often denoted by '?', is obtained at the receiver to denote an erasure. In contrast, with a deletion channel, there is no sign of which bits have been deleted. For example, if 10101010 was sent, the receiver would obtain 10011 if the third, sixth, and eighth bits were deleted, and would obtain 10?01?1? if the bits were erased.

What is the capacity of this channel? Surprisingly, we do not know. Currently, we have no closed-form expression for the capacity, nor do we have an efficient algorithmic means to numerically compute this capacity. Not surprisingly, this lack of understanding of channel capacity goes hand in hand with a lack of good codes for the deletion channel.

More generally, channels with synchronization errors, including both insertions and deletions as well as more general timing errors, are simply not adequately understood by current theory. Given the near-complete knowledge we have channels with erasures and errors, in terms of both the channel capacity and

* Supported in part by NSF grant CCF-0634923.

codes that can nearly achieve capacity, our lack of understanding about channels with synchronization errors is truly remarkable.

On the other hand, substantial progress has been made in just the last few years. A recent result that we will highlight is that the capacity for the binary deletion channel is at least $(1-p)/9$ for *every* value of p [14, 32]. In other words, the capacity of the deletion channel is always within a (relatively small) constant factor of the corresponding erasure channel, even as the deletion probability p goes to 1! As the erasure channel gives a clear upper bound on the capacity, this result represents a significant step forward; while in retrospect the fact that these capacities are always within a constant factor seems obvious, the fact that this factor is so small is somewhat surprising.

The purpose of this survey is to describe recent progress along with a clear description of open problems in this area, with the hope of spurring further research. The presentation is necessarily somewhat biased, focusing on my own recent research in the area. Results to be presented will include capacity lower bound arguments [12–14, 32], capacity upper bound arguments [8], codes and capacity bounds for alternative channel models [23, 30, 31], and related problems such as trace reconstruction [15]. Background information can be found in, for example, [2, 5, 7, 18, 22, 21, 36, 38]. The survey by Sloane on correcting single deletions [38], in particular, gives useful insight and demonstrates the complexity of even the simplest problem related to deletions. There are also several older relevant works on the deletion channel or related problems, including [9–11, 39, 40, 42, 43].

Before beginning, it is worth asking why this class of problems is important. From a strictly practical perspective, such channels are arguably harder to justify than channels with errors or erasures. While codes for synchronization have been suggested for disk drives, watermarking, or general channels where timing errors may occur, immediate applications are much less clear than for advances in erasure-correcting and error-correcting codes. However, this may be changing. In the past, symbol synchronization has been handled separately from coding, using timing recovery techniques that were expensive but reasonable given overall system performance. Indeed, even the model we suggest assumes some high-level synchronization, as both sender and receiver know that n bits are being sent in a transmission. Still, the model appears most natural and appropriate, and a there is clear goal in handling transmission sizes n efficiently and with a high coding rate. With recent improvements in coding theory, it may become increasingly common that synchronization errors will prove a bottleneck for practical channels. Because we are currently so far away from having good coding schemes for even the most basic synchronization channels, in practice coding is rarely if ever considered as a viable solution to synchronization.

If efficient codes for synchronization problems can be found, it is likely that applications will follow. If such codes are even a fraction as useful as codes for erasures or errors have been, they will have a significant impact. The work we describe on capacity lower bounds demonstrates that there is more potential here than has perhaps been realized.

Of course, coding theory often has applications outside of engineering, and channels with deletions and insertions prove no exception, appearing naturally in biology. Symbols from DNA and RNA are deleted and inserted (and transposed, and otherwise changed) as errors in genetic processes. Understanding deletion channels and related problems may eventually give us important insight into genetic processes.

But regardless of possible applications, scientific interest alone provides compelling reasons to tackle these channels. While the deletion channel appears almost as natural and simple as the binary erasure and error channels, it has eluded similar understanding for decades, and appears to hide a great deal of complexity. The fact that we know so little about something so apparently basic is quite simply disturbing. Besides the standard questions of capacity and coding schemes for this and related channels, there appear to be many further easily stated and natural related variations worthy of study. Finally, the combinatorial structure of these channels brings together information theory and computer science in ways that should lead to interesting cross-fertilization of techniques and ideas between the two fields.

2 The Ultimate Goal: A Maximum Likelihood Argument

Rather than start with what has been done recently, we ambitiously begin with what could, if it was well enough understood, be the final word on the subject of capacity for the deletion channel: maximum likelihood decoding. Maximum likelihood decoding simply means finding the most likely codeword given the received sequence and the codebook. If one could suitably analyze maximum likelihood decoding, *and* simultaneously determine near-optimal codebooks, then it might be possible to obtain very tight bounds on the capacity of the deletion channel. At the very least, progress in this direction would likely surpass previous results. Moreover, there are many open questions on the combinatorics of random sequences and subsequences related to this approach that seem interesting in their own right, even if they do not lead directly to near-optimal coding schemes.

Maximum likelihood decoding has a natural formulation in the setting of deletion channels. When the channel sends n bits, we aim for a codebook with 2^{Cn} strings that allows successful decoding with high probability; this would gives us a lower bound of C for the capacity. When n bits are sent and m bits are received over an i.i.d. deletion channel, every set of $n-m$ bits is equally likely to have been deleted. It follows that when considering a received sequence, the probability that it arose from a given codeword is proportional to the number of different ways it is contained as a subsequence in that codeword. Formally, let $X = x_1 x_2 \ldots x_n$ represent the codeword and $Y = y_1 x_2 \ldots y_m$ represent the received word. We use X to signify "X was sent" where the meaning is clear, so that we write $\mathbf{Pr}(X \mid Y)$ for $\mathbf{Pr}(X \text{ sent} \mid Y \text{ received})$. Also, let $\#S(X,Y)$ be the number of times the string Y appears as a subsequence of X. (Recall that a subsequence of X need not consist of a contiguous set of characters of X; that is called a *substring*.) Then

$$\mathbf{Pr}(X \mid Y) = \mathbf{Pr}(Y \mid X)\frac{\mathbf{Pr}(X)}{\mathbf{Pr}(Y)} = \#S(X,Y)d^{n-m}(1-d)^m\frac{\mathbf{Pr}(X)}{\mathbf{Pr}(Y)}.$$

Hence as long as codewords are a priori equally likely to be sent, then given the recevied sequence, we have $\mathbf{Pr}(X \mid Y)$ is proportional to $\#S(X,Y)$.

We therefore have a simple maximum likelihood decoding algorithm for the deletion channel: take the received sequence, count how many times it appears as a subsequence of each codeword, and output the codeword with the largest count. This is not meant to be a practical algorithm, as with a codebook of size exponential in n this counting process is not efficient, but as we are (at least for now) interested just in capacity results, efficiency is not important. There are two questions to answer. Given a codebook, can we analyze this decoding process to bound the probability of failure? How large can we make C so that maximum likelihood decoding will give the correct answer with high probability?

These questions appear quite complicated, and much of the survey will suggest weaker approaches that attempt to approximate this maximum likelihood approach. To highlight some of the challenges, let us ignore the issue of the choice of codebook and simply consider random codebooks where each string in the codebook is initially chosen uniformly from all n bit strings. While we will see in later sections that choosing codebooks in this way is generally *not* a good idea in this setting, even this version of the problem yields interesting open questions. For sucessful decoding, if X was the codeword sent, Y was the string received, and W is any other codeword, we want that

$$\mathbf{Pr}(\#S(X,Y) > \#S(W,Y) \mid X) = o(2^{-Cn}). \tag{1}$$

We could then take a union bound over all codewords W (excluding X) to show the overall failure probability was vanishing, or $o(1)$.

We emphasize the subtle importance on the conditioning on X being sent; the received string Y is not just a subsequence of X, *but a subsequence of X* *obtained via deletions*. Hence the probability of Y being received given that X is sent is itself proportional to $\#S(X,Y)$.

The natural approach to proving equation 1 would be as follows:

1. Find (or bound) the distribution of $\#S(X,Y)$ conditioned on X being sent, where Y is the received string.
2. Find (or bound) the distribution of $\#S(W,Y)$ where Y and W are random strings of the appropriate length.
3. Use the distributions to show that $\#S(X,Y) > \#S(W,Y)$ with suitably high probability.

Currently, there do not appear to be known bounds on either $\#S(X,Y)$ or $\#S(W,Y)$ that appear to lead significantly in the right direction. There is some related work on pattern matching (see, e.g., [24][Chapter 7]), where the distibution of the number of times a fixed-length string is a subsequence of another string has been considered. But here the subsequence grows linearly with the size of the string. The problems of determining these particular distributions appear open, and would seem to be interesting problems in the combinatorics of random strings in their own right. One can imagine other similar related combinatorial questions. For example, if we thinking about changing the conditioning above, we might ask about

$$\mathbf{Pr}(\#S(X,Y) > \#S(W,Y) \mid |Y| = m, Y \text{ is a uniformly chosen subsequence of } X).$$

Or slightly further afield, we might ask the distribution of $|\{y : \#S(X,y) > 0\}|$; that is, what is the distribution of the number of distinct subsequences of a random string. Interestingly, the answers to these questions also do not seem to be known.

To gain insight into these distributions it seems natural to use the fact that given strings A and B, we may set up the a dynamic-programming formulation to count $\#S(A,B)$ efficiently. Let $\#S(A_k,B_j)$ be the number of times the prefix of length j of B is a subsequence of the prefix of length k of A. We have

$$\#S(A_k,B_j) = \#S(A_{k-1},B_j) + I[a_k = b_j]\#S(A_{k-1},B_{j-1}), \tag{2}$$

where $I[a_k = b_j] = 1$ if $a_k = b_j$ and 0 otherwise. The challenge in this simple-looking recurrence is that the $I[a_k = b_j]$ are fiercely dependent. If instead they were independent random bits, then rewriting equation 2 (including the indices, and using $S_{i,j}$ for $\#S(A_k,B_j)$) would give

$$S_{i,j} = S_{i-1,j} + R_{i,j}S_{i-1,j-1},$$

where the $R_{i,j}$ are unbiased, independent random bits. In matrix form, letting S_i be the column vector associated with the ith row of $S_{i,j}$, we would have

$$S_i = M_i S_{i-1}$$

where each random matrix M_i is 1 along the diagonal and has random 0-1 entries just below the diagonal. Generally such random recurrences lead to lognormal or power law distributions [19, 29], and we expect similar behavior here, although the dependencies appear to make the resulting analysis much more challenging.

Open Questions:

- Determine the distribution of $\#S(X,Y)$ when X and Y are independent, uniform bit strings of length n and m, respectively.
- Determine the distribution of $\#S(X,Y)$ when X is a uniform bit string of length n and Y is obtained from X by passing through a deletion channel.
- Determine the distribution of the number of distinct subsequences of a bit string X chosen uniformly at random.
- Generalize the above questions, where appropriate, to more general distributions for X.
- Determine techniques for analyzing recurrences with the form of equation (2).
- Find appropriate codebooks for maximum likelihood decoding for the deletion channel, and analyze the performance of these codebooks.

3 Shannon-Style Arguments

Given our discussion of the maximum likelihood approach above, one might wonder why the basic approach of Shannon [37] fails for the deletion channel. After all, it works perfectly well for the binary symmetric channel and the binary erasure channel.

For context, let us recall one form of the standard Shannon argument for the binary symmetric channel. Codewords are chosen uniformly at random from all n-bit strings, and decoding is done by finding the codeword whose Hamming distance from the received string is closest to the expected number of errors. Shannon's argument shows that this approach provides a code, and hence a lower bound for the capacity, that is essentially optimal. The key to proving this lower bound is to show that with high probability, the Hamming distance between the received string and the codeword sent is near its expectation, and all other codewords besides the one actually sent are sufficiently far from the expected distance from the received codeword.

At the heart of this argument are three important steps:

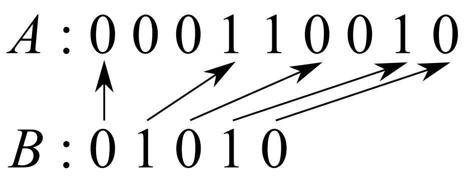

Fig. 1. Example of the greedy algorithm for checking subsequences.

1. a method for choosing a codebook,
2. a notion of what a typical input/output pair from the channel looks like,
3. and a method for decoding a received codeword.

3.1 How The Basic Argument Fails

Let us consider what happens when we try generalize the standard Shannon argument to the deletion channel. Codewords are again be chosen independently and uniformly at random. A typical codeword has roughly pn deletions. The natural decoding approach is to determine if the received sequence is a subsequence of exactly one codeword; if this is the case, then decoding is successful, and otherwise it fails. This choice of codewords and decoding scheme leads to a very weak lower bound on the capacity, which we henceforth denote by C_p. In particular, the bound is non-zero only when $p < 0.5$.

To see this, we first note that there is a simple greedy approach to determining if a string B is a subsequence of a string A. Reading from left to right, simply take the first character of B, and match it with the leftmost appearance of this character in A; then take the second letter of B, and match it with the subsequent leftmost appearance of this character in A; and so on down the line (see Figure 1).

Now consider what happens when the deletion probability p is larger than 0.5. The random codeword X is sent over the channel, yielding a string Y with on average $n(1-p)$ bits. For large enough n, Y will have fewer than $n/2$ bits with high probability. Consider *any* other codeword W, which was chosen uniformly at random independently from X and Y, and consider how the greedy algorithm above behaves when checking if Y is a subsequence of W. For each bit of Y to be matched, on average we will need two bits of W to match it; specifically, the number of bits of W needed to match a bit of Y is geometrically distributed with mean two. It follows that with high probability only about $2n(1-p) < n$ bits of W are needed by the greedy algorithm to confirm that Y is a subsequence of W. That is, each other codeword has Y as a subsequence with high probability when $p > 0.5$, and this basic Shannon argument gives no useful lower bound on the channel capacity.

Given our previous discussion of maximum likelihood, it would seem natural to consider improved decoding methods. However, work in the mid 1990's first focused on the step of finding a better codebook. While there was earlier work on the deletion channel and related channels, we take this work as our starting point for discussion of recent developments for deletion channels.

3.2 Codebooks from First Order Markov Chains

The work of Diggavi and Grossglauser [6, 7] improves on the naïve Shannon argument, using the insight that choosing codewords uniformly at random does not seem to be the right approach for the deletion channel. Indeed, this insight pervades much of the subsequent work after their paper. Instead, they consider codewords generated randomly by a first order Markov process, which yields codewords consisting of blocks (or, equivalently, runs) of alternating 0's and 1's, with the lengths geometrically distributed. More concretely, a codeword starts with a bit chosen uniformly at random. Subsequently, we construct the codeword by having each new bit be the same as the previous bit with probability γ (independently for each bit). We will generally have $\gamma > 1/2$, giving longer contiguous blocks of 0's and 1's, although this is not a requirement of the analysis. Intuitively, using such codewords make a great deal of sense: if bits are going to be deleted, sending multiple copies of a bit, as in a repetition code, should be helpful. In the context of decoding using the greedy algorithm to determine if the received sequence is a subsequence of a codeword, longer blocks intuitively can be better because it makes it harder for the received string to be a subsequence of another codeword, as now long blocks of 0's and 1's in the received string would have to align with the corresponding blocks of another codeword. We summarize the more formal argument of [6] here.

The idea is to again analyze the behavior of the greedy algorithm on a pair W and Y, where Y is the received string and W is a codeword independent of the transmitted string X. The analysis is based on a few key lemmas. The first lemma says that the received string Y can be thought of as having arisen itself from a first order Markov chain, with a different parameter.

Lemma 1. *The received string Y behaves as a random string given from a first order Markov chain with parameter*

$$q = \frac{1-\gamma}{1+p(1-2\gamma)}.$$

Proof. We think of the received string Y as being generated as follows: as bits of X are being generated, undeleted bits pass through to the received sequence. By symmetry the first bit of Y will be either 0 or 1 each with probability $1/2$. The process generating the received sequence is clearly Markovian. To determine the probability that a bit of Y is the same as the previous bit, note that the number of bits X that are deleted after the jth undeleted bit is k with probability $p^k(1-p)$. The corresponding probability that the $(j+1)$st undeleted bit (assuming it exists) is the same as the jth if k bits are deleted after the jth bit is easily found to be $(1+(2\gamma-1)^{k+1})/2$ using standard methods. We therefore have that Y behaves like a first order Markov chain with parameter q where

$$q = \sum_{k=0}^{\infty} p^k(1-p)\frac{1+(2\gamma-1)^{k+1}}{2} = \frac{1}{2}+\frac{1}{2}\frac{(1-p)(2\gamma-1)}{1-p(2\gamma-1)} = 1-\frac{1-\gamma}{1+p(1-2\gamma)}.$$

The second lemma is merely a generalization of the argument for the behavior of the greedy algorithm in this setting. We ignore the first bit of W and Y (which match with probability 1/2; ignoring it does not affect the asymptotics), and consider how many bits the greedy algorithm takes in matching subsequent bits. Here, there are two cases.

Lemma 2. *Consider a codeword W and a received string Y. Then (except for the first bit), the number of bits required to match each bit using the greedy algorithm has the following distribution: if a bit of Y is the same as the previous bit, a match takes 1 bit with probability γ, and i bits for $i \geq 2$ with probability $(1-\gamma)^2\gamma^{i-2}$; otherwise, a match takes i bits for $i \geq 1$ with probability $(1-\gamma)\gamma^{i-i}$.*

Proof. Suppose we are matching the kth bit of Y after having matched the $(k-1)$st bit. When these bits are the same, the next bit in W examined by the greedy algorithm is also the same with probabiity γ; otherwise, we must wait for the next "switch" in the Markov chain for W, giving the next match after i bits with probability $(1-\gamma)^2\gamma^{i-2}$. If the kth and $(k-1)$st bits of Y differ, and the next "switch" in the Markov chain for W occurs after i bits with probability $(1-\gamma)\gamma^{i-i}$.

The resulting analysis is reasonably straightforward, although the expressions that arise are somewhat unwieldy. Given p and γ, one can obtain high probability bounds on the length of a received sequence Y (around its expectation $(1-p)n$) and derive an expression on the distribution of the number of times successive bits differ using Lemma 1. Then, using Lemma 2, we can bound the probability that Y is a subsequence of a codeword W, using standard the Chernoff bound methodology. By applying the union bound, we find that the inverse of this probability essentially tells us the number of codewords we can have while still decoding successfully with high probability, providing a bound on the rate. We thereby obtain an expression to find the best parameter γ given the deletion probability p. This expression can be evaluated numerically to find a non-trivial lower bound on the capacity for every $p < 1$.

Specifically, we have the following bound from [7][Corollary 4.2]

Theorem 1. *The capacity C_p for the deletion channel with deletion probability p satisfies*

$$C_p \geq \sup_{0<\gamma<1,\delta>0} \left[-(1-p)\log_2((1-q)A + qB) - \frac{\delta}{\ln 2} \right],$$

where $A = \frac{(1-\gamma)e^{-\delta}}{1-\gamma e^{-\delta}}$, $B = \frac{(1-\gamma)e^{-2\delta}}{1-\gamma e^{-\delta}} + \gamma e^{-\delta}$, and $q = 1 - \frac{1-\gamma}{1+p(1-2\gamma)}$.

We wrap up with some final considerations. The analysis of the greedy algorithm used here can be extended to other block length distributions; this extension (along with other small improvements) was studied in [13], where other distributions were used to obtain better bounds for high deletion rates. Specifically, the authors show that a “Morse-coding” approach, where blocks take on one of two lengths, either short or long, yields improved capacity bounds for $p \geq 0.35$. This raises again the question of what is the right block distribution, both specifically for arguments based on the greedy algorithm and for other approaches for obtaining capacity bounds.

We now detail another possible improvement to the greedy approach that, although not apparently effective, raises some further interesting questions. Suppose we try to improve our decoding algorithm by considering not just whether a received string Y is a subsequence of another codeword W, but how many bits of W are needed for Y to be covered according to the greedy algorithm. If Y is a subsequence of the first αn bits of the actual codeword X for some $\alpha < 1$ with high probability, then we could immediately improve our analysis, requiring that a failure occur only when Y is a subsequence of the first αn bits of some other codeword W.

We prove that this approach in fact fails in the case where the codewords are chosen uniformly at random. Unfortunatley, our argument does not appear to generalize naturally to the case where codewords are generated by first order Markov chains, although experiments strongly suggest the result should. This remains an open question.

We establish some notation. Consider the greedy algorithm working on strings X and Y, where Y was obtained from X via deletions. If $x_i = y_j$ was not deleted, let $G(i)$ be the difference between i and the number of bits needed to cover up to y_j by the greedy algorithm. If x_i was deleted, let $G(i)$ be the difference between i and the number of bits needed to cover up to y_k, where $k < i$ is the largest index for which y_k was not deleted. For convenience, we introduce an initial gap $G(0) = 0$; implicitly, we assume a 0th character that was not deleted. For example, for the sequences $X = 0110101$ and $Y = 111$, if the second, third, and seventh bits were not deleted, the sequence of gaps $G(i)$ would be $0,1,0,0,1,2,3,2$. The final gap, 2, gives the number of bits from the end of X where the greedy subsequence algorithm matches the last bit of Y.

Our goal will be to show that the gap G behaves like a simple Markov chain, according to the following lemma:

Lemma 3. *When X is chosen uniformly at random from all n-bit sequences, and Y is obtained by deleting bits independently with probability p, then $G(i)$ behaves as follows:*

$$G(i+1) = \begin{cases} G(i)+1 & \textit{with probability } p \\ G(i)-c & \textit{with probability } (1-p)/2^{c+1}, 0 \leq c \leq G(i)-1 \\ 0 & \textit{with probability } (1-p)/2^{G(i)}. \end{cases}$$

We prove this lemma below. The Markov chain that models the gap behavior is very similar to the one-dimensional Markov chain on the number line with a boundary at 0 that moves right with probability p and left with probability $1-p$. While the gap can move down by more than one on any step, when the gap is large the expected increase in the gap is approximately $2p-1$, and the variance in the change of the gap at each step is bounded. This similarity suggests the following theorem, which can be proven using standard techniques.

Theorem 2. *The gap $G(n)$ is $(2p-1)n + o(n)$ with high probability when $p > 1/2$; is $O(\log n)$ with high probability when $p < 1/2$; and is $O(\sqrt{n}\log n)$ with high probability when $p = 1/2$.*

Proof (Lemma 3). We first claim that, conditioned on the state $G(i) = j$ (with $j \leq i$), bits $x_{i-j+1}, \ldots, x_i$ remain uniform over all possible bit sequences. This can be proven by induction on i. It is true when $i = 0$. Suppose that it is true for $i \leq k$. If x_{k+1} is deleted, then $G(k+1) = G(k)+1$, regardless of the value of x_{k+1}, and the induction holds. If x_{k+1} is not deleted, then since inductively bits $x_{k-G(k)+1}, \ldots, x_k$ are uniform, regardless of the value of x_{k+1}, the number of bits needed to cover x_{k+1} is a geometrically distributed random variable, truncated at $G(k)+1$. The probability that $G(k+1) = j$ conditioned on the value of $G(k)$ is thus independent of the value of x_{k+1}, completing the induction.

The lemma now follows. If $G(i) = j$, with probability p bit x_{i+1} is deleted, and in this case $G(i+1) = j+1$. Otherwise, regardless of the value of x_{i+1}, the number of bits of X starting from x_{i-j+1} to cover x_{i+1} is a geometrically distributed random variable, truncated at j+1. This gives the distribution for $G(i+1)$ stated in the lemma.

This argument highlights an intriguing threshold behavior. When $p > 1/2$, a received string Y will require on approximately $2(1-p)n$ bits of X when the greedy algorithm is run on X and Y. Of course, the same is true the greedy algorithm is run with Y and another codeword W, or on X and another random string Z of length $(1-p)n$; there is, in this regard, no distinction. This contrasts with the case where $p > 1/2$, where the greedy algorithm will detect that Y is a subsequence of X using nearly all n bits of X, but will with high probability fail on pairs of strings Y and W, or on pairs X and Z, where again W is another codeword and Z is a random string. Experimentally, there appears to be a similar threshold behavior when codewords are generated by a first order Markov chain; above some deletion threshold, the greedy algorithm behaves essentially the same on random subsequences and purely random strings, but it seems harder to prove this more general result.

While the following related open questions on the behavior of the greedy algorithm are perhaps now of less immediate interest to the analysis of the deletion channel given the improved decoding methods to be considered in the next section, they remain interesting combinatorial questions.

Open Questions:

- Find optimal block distributions for generating codewords for the analysis of the capacity of the deletion channel based on the greeedy algorithm.
- Find optimal codebooks for the deletion channel based on the greeedy algorithm.
- Generalize Lemma 3 to determine the behavior of the gap when the string X is generated by a first-order Markov chain.

4 Better Decoding via Jigsaw Puzzles

We have seen that the choice of codebook makes a significant difference in the power of decoding using the greedy algorithm. Even so, there are clear limitations of this approach that suggest that it could never reach the capacity promised by the maximum likelihood approach. Moreover, it is not clear how to extend the greedy approach when there are both insertions and deletions. In such cases, the received string need not be a subsequence of the codeword (nor vice versa).

We now consider a different decoding approach that allows for much stronger capacity lower bounds for the deletion channel, as presented in [14]. Further,

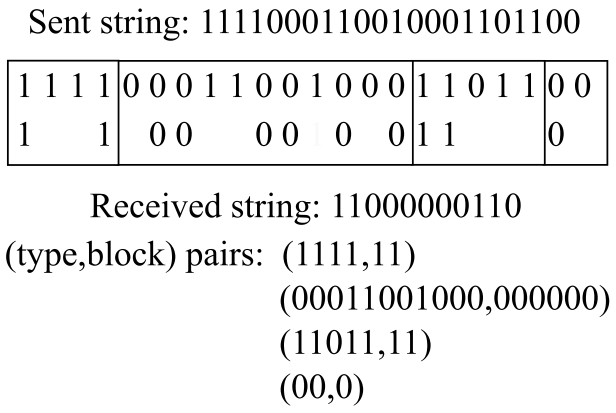

Fig. 2. Examples of types, and how they relate to received blocks and the sequence of deletions.

the approach also naturally handles a certain class of insertions, specifically *duplications*: a bit that passes through the channel might be deleted, or might be duplicated a number of times. While more general insertion channels could conceivably also be tackled utilizing this approach, the analysis becomes much more tractable when considering only duplications.

To begin, consider a maximal block at the receiver. The bits of that block came from, or correspond to, one or more blocks from the sender. For example, the block 000 at the receiver could come from a single block 00000 from the sender. Alternativley, it could have come from the sequence of blocks 001100 from the sender, if the two 1 bits and one of the 0 bits were deleted. (For concreteness, we say that the first block from the sender corresponding to the block at the receiver is the block from which the first bit in the received block came. So here, we assume that the first received zero in the block was indeed one of the first two zeroes, and that in the subsequent block of ones after these six bits, at least one bit is not deleted.) We will call the sequence of blocks in the codeword from which a block at the receiver corresponds to the *type* of the received block. Now, given the original codeword and the deletions that occurred, we can think of the codeword and the received sequence of being described by an ordered collection of ordered pairs (t_i, k_i), where k_i is the length of the ith block in the received sequence and t_i is its corresponding type. (Naturally, one also needs to consider whether the blocks are of 0 bits or 1 bits, but once one knows whether the first received block consists of zeroes or ones, the parity of all further blocks is known, so we can ignore this issue in the analysis.) Examples of (type,block) pairs, with both the type and block expressed as bit strings, is given in Figure 2.

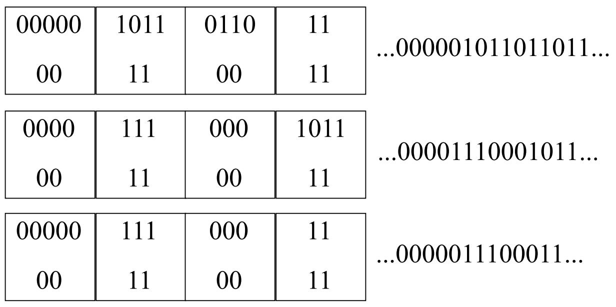

Fig. 3. Examples of different ways that the jigsaw puzzle tiles might cover a substring of the received string. Each tile configuration gives a different possible codeword, and the codewords are then scanned for a match.

Given this framework, we can consider the requirements for decoding. Recall that in the standard Shannon argument for the symmetric binary error channel, we can require that the number of errors is close to its expectation, as this will be true with high probability. In this setting, loosely speaking, we can instead require that the number of pairs (t,k) in the description of the codeword and received string be close to its expectation, as this will be true with high probability.

Let us temporarily assume that the number of pairs (t,k) in the description of the codeword and received string is exactly equal to its expectation (rounded appropriately), so that we exactly know the number of (t,k) pairs for every (t,k). Then to decode, it suffices to match the types to the block lengths of the received string in such a way that we obtain a codeword. If only one codeword results from this type of matching, then we have a successful decoding.

In [14] the analogy is made between decoding this way and a jigsaw puzzle. Pictorially, we can represent this as in Figure 3. A pair (t,k) can be represented by a tile that corresponds to a block in the received sequence (at the bottom) and the corresponding type for the pair (at the top). To decode, the tiles must be arranged in such a way that the string of concatenated blocks at the bottom corresponds to the received sequence, and the string of concatenated corresponding types forms a codeword. It is worth emphasizing that, with the jigsaw-puzzle approach, it is no longer the case that the received string be a subsequence of only a single codeword, as in the greedy approach. Instead, it has to be a subsequence of a single codeword in, essentially, "the right way," although there may be several arrangements of the tiles that lead to that codeword.

Although the above argument was expressed as though we knew the number of appearances of each possible pair (t,k) in the description, all we really know is that these numbers are concentrated around their expectation. This is handled by simply utilizing the jigsaw-puzzle approach over multiple sets of pieces, where each set of pieces corresponds to an enumeration of the number of pairs (t,k) in a description of the codeword and received string consistent with this concentration. In the asympotics, the effect of all of these cases does not change the overall capacity.

We briefly sketch a high-level argument showing how this leads to an expression for a capacity lower bound. Let P be the distribution of block lengths at the sender, so that P_j is the probability a block at the sender has length j, and similarly let Q be the corresponding distribution of block lengths at the receiver. (The argument requires P and Q to satisfy some basic requirements; distributions with exponentially decreasing tails suffices.) Hence the number of blocks at the sender is (up to lower order terms) $\frac{n}{\sum_j jP_j}$, and the number of blocks at the receiver is (up to lower order terms) $\frac{n(1-p)}{\sum_j jQ_j}$. Let K represent a random block (or, more precisely, its length) in the received string, and T the type corresponding to this block. The number of codewords considered by the jigsaw-puzzle decoding process can be upper bounded as follows: over all of the $\frac{Q_k n(1-p)}{\sum_j jQ_j}$ blocks of length k, there are (again, up to lower order terms)

$$2^{\frac{Q_k n(1-p)}{\sum_j jQ_j} H(T \mid K=k)}$$

ways of assigning the appropriate types to these blocks. Taking the product over all values of k gives an upper bound of

$$2^{\frac{n(1-p)}{\sum_j jQ_j} \sum Q_k H(T \mid K=k)} = 2^{\frac{n(1-p)}{\sum_j jQ_j} H(T \mid K)}$$

possible codewords; this is an upper bound, as many of these possible codewords will be repeated, arising from multiple different combinations. For example, consider a substring 001100110011 of a codeword sent over a deletion channel that leads to blocks 0011 at the receiver. There can be many different ways of assigning types to the blocks 00 and 11 at the receiver that can lead to the matching substring 001100110011 from the sender, which causes us to overcount the possible codewords that can arise. Avoiding or reducing this overcounting of possible codewords could improve the resulting capacity bound.

Standard techniques give that (with high probability) each possible codeword has probability

$$2^{-\frac{n}{\sum_j jP_j} H(P)}$$

of being generated. Hence, by a union bound, if we start with 2^{Cn} codewords, the probability of another codeword other than the original codeword sent arising from jigsaw-puzzle decoding is

$$\left(2^{C+\frac{(1-p)}{\sum_j jQ_j}H(T\mid K)-\frac{1}{\sum_j jP_j}H(P)}\right)^n,$$

which vanishes as long as the exponent is negative.

All of this can be formalized, leading to the following lower bound for the capacity:

$$C_{del} \geq \frac{1}{\sum_j jP_j}H(P) - \frac{(1-p)}{\sum_j jQ_j}H(T\mid K),$$

for any suitable distribution P.

The authors test both geometric distributions P and the "Morse-code" based distributions of [13] in this framework. Here, geometric distributions appear to do slightly better across the board. As an example, the calculted lower bound for the capacity of the deletion channel when $p = 0.9$ is 0.012378. This result over two orders of magnitude better than that given the argument using the greedy algorithm of [6], and is rather striking given that the obvious upper bound on the capacity based on the erasure channel is 0.1.

Like Shannons's original arguments, this approach shows the existence of a code with a given capacity, but does not give an explicit codebook nor an algorithmically *efficient* decoding algorithm. As we discuss further below, designing efficient algorithms for deletion channels remains quite open for further research.

Open Questions:

- Improve the analysis of jigsaw puzzle decoding by reducing or avoiding the overcounting of possible codewords generated by taking all consistent combinations of types.
- Find better distributions, or a method for finding near-optimal distributions, for codebooks for the jigsaw-puzzle analysis.
- Extend the jigsaw-puzzle approach to find lower bounds for capacity under more general models of insertions.
- Find a variant of jigsaw-puzzle decoding that is efficiently implementable.

5 A Useful Reduction for Deletion Channels

The arguments thus far provide lower bounds that, in the end, rely on a non-trivial calculation for each specific deletion probability p. An interesting alternative path was suggested in [32] by building a correspondence between deletion channels and a novel insertion-deletion channel dubbed a Poisson-repeat channel. This correspondence is given in the form of a reduction, whereby *any* decoding algorithm for *any* Poisson-repeat channel can be turned into a decoding algorithm for *any* deletion channel. Using this reduction, we can obtain lower bounds on the capacity of all deletion channels simply by finding a lower bound on the capacity of any Poisson-repeat channel. Specific Poisson-repeat channels can then be analyzed using the previously established machinery of Section 4.

While the bounds obtained in this fashion are not particularly good for smaller values of p, they appear quite good for large values of p. Indeed, one benefit of this approach is that it allows us to consider the behavior of deletion channels as the deletion probability p goes to 1.

A Poisson-repeat channel with parameter λ on binary inputs of length n can be defined as follows: as each bit passes through the channel, it is replaced by a discrete Poisson number of copies of that bit, where the number of copies has mean λ and is independent for each bit. Notice that such a channel may incur deletions when the number of copies is zero, and that the channel fits into the class of deletion-duplication channels of Section 4 for which numerical capacity lower bounds can be found.

What is the relationship between a deletion channel with parameter p and a Poisson-repeat channel with parameter λ? Suppose that we have a deletion channel. Before sending a message, independently replace each bit sent with a *random* number of copies, according to an Poisson random variable with mean $\lambda/(1-p)$. Because of the properties of the discrete Poisson distribution, the number of copies of a bit that arrive at the receiver has exactly a Poisson distribution with mean λ, and the number of copies is independent for each bit. Therefore our deletion channel will now behave exactly as a Poisson-repeat channel with mean λ on the original input. A code for any Poisson-repeat channel thereby gives us a code for any deletion channel. (More precisely, we have a procedure that includes a random replacement step; using standard arguments, this shows the existence of a fixed code with the desired rate.) Because each bit of the input must be blown up by a expected factor of $\lambda/(1-p)$, we find that if the capacity of the Poisson-repeat channel with parameter λ is L_λ, then

$$C_p \geq L_\lambda(1-p)/\lambda.$$

A pictorial representation of this reduction is given in Figure 4.

Using calculations from the arguments of Section 4, we find that $L_1 > 0.1171$, and in fact we can do slightly better with $L_{1.79}/1.79 > 0.1185$. The result can be more pleasantly if somewhat less accurately expressed by $C_p \geq (1-p)/9$.

In retrospect, it seems clear that there should be a lower bound for C_p that scales with $1-p$, since one could deterministically replace each bit to be sent by

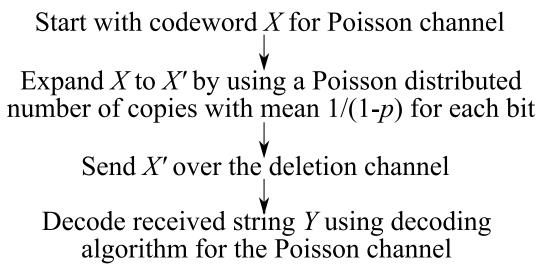
Start with codeword X for Poisson channel
Expand X to X' by using a Poisson distributed number of copies with mean 1/(1-p) for each bit
Send X' over the deletion channel
Decode received string Y using decoding algorithm for the Poisson channel

Fig. 4. How to turn a codebook and decoding algorithm for a Poisson-repeat channel into a code for the deletion channel.

$1/(1-p)$ bits, and "on average" each bit would pass through once. In effect, the reduction involving the Poisson-repeat channel codifies this intuition; however, the scaling does not appear to have been proven previously to [32].

While this reduction has been used to obtain a capacity lower bound, it is worth noting that it could also be used to derive efficient encoding and decoding algorithms for the deletion channel: an algorithm for encoding and decoding on any Poisson-repeat channel would immediately give a corresponding algorithm for the deletion channel for all values of p. While it is likely that such a code would only offer reasonable rates for large values of p, the prospect remains interesting. Indeed, perhaps, asymptotically as p goes to 1, essentially the best we can do is to turn the deletion channel into a Poisson-repeat channel by duplicating each bit a suitably large number of times.

Open Questions:

- Improve the lower bounds for the capacity of Poisson-repeat channels.
- Find an efficient encoding and decoding algorithm for a Poisson-repeat channel with parameter 1 (or some other fixed parameter).
- Prove or disprove the following conjecture: There exists a λ such that

$$\lim_{p \to 1} C_p/(1-p) = L_\lambda/\lambda.$$

- Find a similar reduction that provides stronger results in the regime where p approaches 0.

6 A Related Information Theoretic Formulation

The lower bound results presented thus far have been derived via an algorithmic approach, based on designing a decoding scheme and using combinatorial analysis to determine its performance. Can we re-frame these results in a more purely information-theoretic context? Such a framework could allow the powerful tools of information theory to be more easily applied to these problems.

Recently, Drinea and Kirsch [12] have provided an alternative, information-theoretic analysis to obtain lower bounds on the capacity of the deletion channel that may prove a promising path further work. The starting point for their analysis is the work by Dobrushin, who generalized Shannon's theorem to show that for a broad class of channels with synchronization errors, including deletion channels, the information and transmission capacities are equal [9]. As before, they assume that codewords are generated according to a distribution on the block lengths. Their insight begins with the recognition that the channel can be viewed as a joint renewal process; each time a block at the receiver ends, a corresponding block at the sender must also have ended, and therefore it is as though the whole process restarts. While this idea is essentially implicit in to the type-based arguments of Section 4, it is utilized here differently, in order to directly derive bounds on the information capacity. By directly working with the information capacity, they avoid some of the technical issues that arise in analyzing a specific decoding algorithm.

The main result of this approach is an alternative and seemingly simpler proof of the lower bounds described in Section 4. The proof is certainly likely to be much more intuitive and natural to those who prefer information-theoretic formulations to algorithmic ones. But a further advantage is that the proof sheds light on how the lower bound can be improved, providing a clear direction for future progress. Specifically, in the terms used previously in the discussion of types, where P is the distribution of block lengths at the sender and Q is the corresponding distribution of block lengths at the receiver, they prove that the mutual information in fact equals

$$\frac{1}{\sum_j jP_j} H(P) - \frac{(1-p)}{\sum_j jQ_j} H(T \mid K) + E_{P,Q},$$

The first two terms correspond exactly to the lower bound of [14]. The last term $E_{P,Q}$ is an expression that corresponds to the uncertainty in where the blocks from the sender from which a block at the receiver was derived. In algorithmic terms, $E_{P,Q}$ corresponds to the loss from jigsaw-puzzle decoding that stems from using pieces that correspond to a single block at the receiver. Instead of considering the type of a single block at a receiver in the jigsaw-puzzle decoding, suppose each tile represented a pair of consecutive blocks at the receiver, along with the possible bit sequence that pair of blocks could have arisen from at the sender. By considering pairs (or larger groups) of blocks under jigsaw-puzzle decoding, one could naturally reduce the overcounting of possible codewords, and obtain a better bound. This proves quite complex in terms of the calculations required and was not pursued in [14].

While this additional expression $E_{P,Q}$ does not at this point have a formal lower bound, in [12] the authors study it by simulation, and show that non-trivial improvements in the capacity lower bounds remain possible. Because they find an exact expression for the information capacity (albeit in terms that are not yet directly calculable), in a strong sense this work shows the limits of what is possible using independent and identically distributed (i.i.d.) block lengths to generate codewords. It is not clear, however, that one could not obtain even better lower bounds by allowing more general methods to generate a codebook.

Open Questions:

- Using the information-theoretic formulation of [12], bound $E_{P,Q}$ to derive improved lower bounds for the deletion channel.
- Find ways to more formally connect the information-theoretic analysis and the algorithmic analysis for the lower bound.
- Consider random codebooks for the deletion channel that are *not* constructed by using i.i.d. block lengths. Can one show that more general methods can lead to higher transmission capacity?

7 Upper Bounds on the Capacity of Deletion Channels

Given the advances that have occurred in finding lower bounds for the capacity of deletion channels, it seems natural to consider upper bounds. Our lower bounds might, after all, be approaching their natural limit. Unfortunately, very little is known about upper bounds for the deletion channel; past upper bounds have focused on other error models. Until recently, the only formal upper bound for the deletion channel that we are aware of is the trivial upper bound from the erasure channel of $1-p$.

The question of upper bounds was recently tackled in [8]. The authors first provide an upper bound approach based on a transformation that allows one to instead bound a memoryless channel's capacity per unit cost. (See similarly the results for sticky channels below in Section 8.1.) Binary input strings are transformed into a sequence of run lengths, so the string 0010111 would be represented as $2,1,1,3$. The output can also be viewed as a sequence of run lengths. In order to turn the deletion channel into a memoryless channel, the output is enhanced with side information: if an entire run of the input is deleted, a run length of 0 is given at the output. For example, if the input 0010111 led to an output of 001111, the corresponding output run lengths with side information would be $2,1,0,3$, instead of the actual block lengths $2,4$.

In this setting, it makes sense to define the *cost* of a symbol for the new channel to simply be the corresponding number; i.e. the symbol 2 has a cost of two, as it corresponds to two bits of the input. Then the capacity per unit cost of the new channel with side information gives an upper bound on the capacity of the deletion channel. Standard techniques [1] then give that the capacity can be bounded by

$$C \leq \sup_x \frac{I(x)}{x},$$

where $I(x) = D(P(x), Q)$ is the information divergence between the output distribution on input x and Q is any distribution over the positive integers. It remains a technical challenge to choose a near-optimal distribution Q over the infinite domain and numerically determine bounds; this is discussed further in [8].

The resulting upper bounds remain reasonably far from the known lower bounds, as shown in Table 7, taken from [8]. Given the power of the additional side information, it would suggest that the upper bound could be further improved. In particular, for large deletion probabilities, the bounds become worse than the trivial $1-p$ bounds; the problem is that the additional value of the side information of having a 0 output symbol becomes more significant for large p, as entire blocks are deleted more often.

Another question considered in [8] is the value of the asymptotic bound as p goes to 1. Recall that a capacity lower bound of $c_1(1-p)$ with $c_1 = 0.1185$ was obtained, as described in Section 5. An asymptotic upper bound of the form $c_2(1-p)$ as p goes to 1 provides corresponding limits on how far such results can be taken.

The upper bound utilizes some of the insights of the lower bound argument. Specifically, we can again introduce side information in this setting, by having a marker every $a/(1-p)$ bits for some constant a. Then, asymptotically, one expects to receive on average a bits between markers, and the number of bits between markers has, approximately, a Poisson distribution. The markers again make this a discrete memoryless channel, with input symbols consisting of sequences of $2^{a/(1-p)}$ bits and output symbols consisting of a sequence of bits, the number of which has an approximately Poisson distribution.

This channel appears difficult to analyze, especially given the space of input symbols, which again correspond to $2^{a/(1-p)}$ possible sequences. A technical argument reduces the problem to a more manageable state space. One can consider outputs of up to only k bits for a finite and not too large k at the cost of only a small overestimate of the constant c in the upper bound. Under the assumption that the output is only k bits, it can be shown that one need only consider input sequences consisting of at most k alternating blocks of zeroes and ones. That

p	LB	UB
0.05	0.7283	0.816
0.10	0.5620	0.704
0.15	0.4392	0.6188
0.20	0.3467	0.5507
0.25	0.2759	0.4943
0.30	0.2224	0.4466
0.35	0.1810	0.4063
0.40	0.1484	0.3711
0.45	0.1229	0.33987
0.50	0.1019	0.31082
0.55	0.08432	0.28382
0.60	0.06956	0.25815
0.65	0.05686	0.2331
0.70	0.04532	0.2083
0.75	0.03598	0.183
0.80	0.02727	0.157
0.85	0.01938	0.1298
0.90	0.01238	0.0999*
0.95	0.00574	0.064*

Table 1. Comparison of lower bounds from [14] (LB) with upper bounds from [8] (UB). Entries denoted * are worse than the $1-d$ bound.

is, any input symbol can be effectively represented by its starting bit and a j-dimensional vector $(q_1, q_2, \ldots, q_j)$, with $1 \leq j \leq k$, and $q_i > 0$ being the fraction of the $2^{a/(1-p)}$ bits in the ith alternating block. Since we are considering asymptotic upper bounds, we can allow the q_i to take on any positive real values. With this framework, we now again have a setting where we can upper bound the capacity by the information divergence by a suitable numerical optimization. The current best result gives an asymptotic upper bound of $0.7918(1-p)$ as p goes to 1, so there is still a significant gap between the asymptotic upper and lower bounds.

Open Questions:

- Design new capacity upper bound approaches specifically designed for insertion/deletion channels. Specifically, consider approaches for general p and for p approaching 1.
- Consider the asymptotic behavior for the deletion channel as p goes to 0; how does it compare to $1-p$, the capacity of the erasure channel?

8 Variations on the Deletion Channel

Given that the i.i.d. binary deletion channel has proven so hard to analyze, it seems worthwhile to ask if there are reasonable ways to modify the problem that would allow meaningful progress. We describe here some variations of the deletion channel, why they may be interesting, and some recent results for them.

8.1 Sticky channels

As we have seen, the block structure of the codewords and the received sequences appear quite useful in analyzing channels with synchronization errors. If we no longer allow deletions, but only allow duplications of symbols sent over the channel, then the block structure at the sender and receiver will be the same. We use the term *sticky channels* to refer to the class of channels that independently duplicates each transmitted symbol a random number of times at the receiver, according to some fixed distribution on the positive integers. As an example of a sticky channel, when typing at an electronic keyboard, if the key is held too long, multiple copies of the pressed symbol can appear even though only one copy of the symbol was intended. Sticky channels were studied in [31]. In part, the motivation for sticky channels is the difficulty of handling general insertion/deletion channels; in some sense, this should be the easiest class of insertion/desertion channels to study. Another reason to consider sticky channels is that they can serve as a testing ground for ideas for capacity bounds or practical codes. Techniques that perform well generally should handle the easy case of sticky channels wells.

As with the upper bound argument of [8], a key step in [31] is to think of the block lengths themselves as being the symbols. For example, suppose that the input to the sticky channel was:

$$0011000101101111001011.$$

We could instead think of the message as a sequence of block lengths:

$$2\ 2\ 3\ 1\ 1\ 2\ 1\ 4\ 2\ 1\ 1\ 2.$$

With a sticky channel, the receiver will not only obtain a bit string that can be interpreted as a sequence of block lengths, but the receiver will also have the same number of symbols as the sender.

In terms of capacity, we can relate the capacity of the original sticky channel to the capacity per unit cost of the derived channel. Thinking of our symbols as integers, we again assign the symbol (integer) i a cost of i in this derived channel, since it corresponds to i bits in the sticky channel. It is then intuitively clear that the capacity of the sticky channel is equal to the capacity per unit cost of the derived channel. A more formal argument proving this correspondence is given in [31].

This correspondence allows us to find a lower bound on the capacity of the sticky channel by finding a lower bound on the capacity per unit cost of the derived channel. If the transition probability matrix matrix P_{ij} of the derived channel is finite, then because the symbol costs are all positive, there are numerical methods for computing the capacity using a variation of the Blahut-Arimoto algorithm [16]. This approach does not provide an actual coding scheme, but yields a distribution of block lengths from which the capacity per unit cost can be derived.

Since this numerical approach requires the transition probability matrix P_{ij} to be finite, we can only handle a finite number of input and output symbols. This means the block lengths for the sticky channel input must be limited, so that we have a run-length limited code. Similarly, the block lengths at the output must be limited. For some channels, such as one where each bit is duplicated with probability p, this limit arises naturally once we constrain the input block length. For some channels, such as one where each bit is duplicated a geometrically distributed number of times, this is not the case. However, we can similarly limit the output block length, collapsing all outputs above some maximum length to the maximum length in order to effectively truncate the matrix P. The numerical calculations are straightforward and fast, so these limits can be set quite high, and we would not expect them to affect the capacity calculation significantly. There is no general formal proof of this that we know of, however. Hence, while the approach does give a formal lower bound, and intuitively this bound can be made very tight by taking a suitably large limits on the input and output block lengths, formally obtaining a corresponding upper bound currently can require significant additional work.

For the specific case where each bit is independently duplicated exactly once with probability p, [31] gives very tight bounds on the capacity. Generally, the bounds match to the third decimal place. The formal upper bound utilizes the same general technique for deletion channel upper bounds in [8]: bound $\sup_x \frac{I(x)}{x}$ numerically.

It is worth noing that technically the techniques described for sticky channels hold more generally for channels defined by a finite transition probability matrix P_{ij}, where P_{ij} is the probability that a maximal block of i contiguous equal symbols at the sender yields a corresponding block of length $j \geq 1$ copies of the symbol at the receiver. That is, we could consider bits being deleted as well as inserted, as long as there is a guarantee that no *block* from the sender is ever completely deleted.

Open Questions:

- Find a simpler approach for giving formal upper bounds on the capacity of sticky channels. Possibly, the approach should be connected to the method of obtaining a lower bound by taking a finite truncation of the matrix P.
- Develop efficient encoding and decoding schemes that give near-capacity performance for sticky channels. Specifically, one can start with the channel where each bit is independently duplicated with probability p.
- Derive a closed-form expression for the capacity of a sticky channel. Again, one can start with the channel where each bit is independently duplicated with probability p.

8.2 Segmented deletion and insertion channels

Part of the difficulty inherent in the deletion channel is that, eventually, there will be large blocks of deletions. With constant error probability, codewords with length n will have a patch of $\Omega(\log n)$ conseutive deletions somewhere with high probability. Finding a large missing block appears to be a significant part of the challenge in desigining codes for the deletion channel.

This suggests that we consider adding an additional assumption restricting how deletions occur to prevent long blocks of consecutive deletions. Liu and Mitzenmacher [23] consider a variation of a deletion channel satisfying a *segmentation assumption*: the input is grouped in consecutive segments of b consecutive bits, and there is at most one error in each segment. For example, if segments consist of eight bits, and at most one deletion occurs per segment, on the input

$$0001011100101111,$$

it would be possible that the fourth and eleventh bits were deleted, so that the received sequence would be

$$00001110001111,$$

but *not* that last two bits were deleted, leaving

$$00010111001011.$$

To be clear, from the receiver's point of view the segments are implicit, and no segment markers appear in the received sequence (as shown in Figure 5). If one had markers, then we could just use a 1-deletion correcting code, where each segment of length b would be a codeword. The codewords could be decoded one at a time using the marker boundaries. 1-deletion correcting codes

10000001 ⋮ 10010110 ⋮ 01011010

1000000100101100011010

Fig. 5. An example of a segmented deletion channel. In particular, there are no markers denoting segment boundaries at the receiver.

are covered in Sloane's survey [38]; the most well known such codes are the Varshamov-Tenengolts (VT) codes [40]. While 1-deletion correcting codes are not the answer in this setting, they provide direction for a solution.

Besides being motivated by the difficulty of the i.i.d. deletion channel, the segmentation assumption captures the idea that many synchronization errors are due to small drifts in clock synchronization. There might then naturally be a minimal gap between deletions, as the slow drift may require some minimal amount of time before the timing error translates into another bit error. The segmentation assumption captures this model, and is in fact more general.

Deterministic, linear-time, zero-error codes with relatively high rates are possible under the segmentation assumption [23] The codes utilize codes of 1-deletion codes of length b with certain additional properties; codewords for the segmentation channel are obtained by concatenating codewords of this code. The additional properties are specially selected so that one can decode by reading the received sequence left to right without losing synchronization.

We provide the theorem describing the code properties. First, we require some notation. For a b-bit string u, let $D_1(u)$ be the set of all $(b-1)$-bit strings that can be obtained by deleting one bit from u. We also use $D_1(S) = \cup_{u \in S} D_1(u)$. A zero-error 1-deletion correcting code $\mathscr{C}$ must satisfy for any $u, v \in \mathscr{C}$, with $u \neq v$, $D_1(u) \cap D_1(v) = \emptyset$. For a string x of length $k > 1$, let $\text{prefix}(x)$ be the first $k-1$ bits of x, and similarly define $\text{suffix}(x)$ be the last $k-1$ bits of x. For a set S of strings let $\text{prefix}(S) = \cup_{x \in S} \text{prefix}(x)$ and define $\text{suffix}(S)$ similarly.

Theorem 3. *Consider the segmented deletion channel with segment length b. Let $\mathscr{C}$ be a subset of $\{0,1\}^b$ with the following properties:*

- *for any $u, v \in \mathscr{C}$, with $u \neq v$, $D_1(u) \cap D_1(v) = \emptyset$;*
- *for any $u, v \in \mathscr{C}$, with $u \neq v$, $\mathit{prefix}\big(D_1(u)\big) \bigcap \mathit{suffix}\big(D_1(v)\big) = \emptyset$;*
- *any string of the form $a^*(ba)^*$ or $a^*(ba)^*b$, where $a, b \in \{0,1\}$, is not in C.*

Then, using $\mathscr{C}$ as the code for each segment, there exists a linear time decoding scheme for the segmented deletion channel that looks ahead only $O(b)$ bits to decode each block.

Since the constraints are expressed on pairs of possible strings, by treating each possible bit-string as a vertex and placing an edge between every pair of nodes that cannot both simultaneously be in $\mathscr{C}$, the problem of finding a large code $\mathscr{C}$ reduces to an independent set problem, in a manner similar to other 1-deletion code problems [38]. For small b, solutions to the independent set problem can be found by exhaustive computation; for larger b, heuristic methods can be applied to find large independent sets. As an example, for $b = 8$, or equivalently at most one deletion per byte, a code with rate over 44% was found exhaustively, and for $b = 16$, a code with rate over 59% was found by a heuristic search [23]. Rates appear to improve as b increases.

Liu and Mitzenmacher also consider insertions under the segmentation assumption. Finally, they present a more advanced scheme that experimentally yields a higher rate but is more expensive computationally and not zero-error. However, if the segmentation assumption holds in real systems, the deterministic approach already seems well suited to implementation.

Open Questions:

- Find expressions for the capacity for segemented deletion or insertion channels, either in the zero-error setting or the setting where errors are allowed.
- Find better methods for finding maximum independent sets in graphs that arise in the study of deletion codes and related codes.
- Provide codes that allow for higher rates for segmented deletion channels, by trading off computational effort and code rate.

8.3 Deletions over Larger Alphabets

Thus far we have focused on the case of a binary alphabet. Some of the work generalizes naturally to larger alphabets (see, e.g., [7, 12]). However, for significantly larger alphabets, codes based on blocks of the same repeated symbol does not appear to be the appropriate approach. Indeed, for very large alphabets, one could simply embed a sequence number in the sent symbol itself. In some sense, this is what is already done in the Internet today, where packets contain sequence numbers significantly smaller than the standard packet size, and deletions thereby become more easily handled erasures.

From a theoretical standpoint, however, it is interesting to consider what can be done in the case of deletions over larger alphabets. There may also be practical settings where such coding techniques would apply; in a system where symbols (or packets) consist of for example 64 bits, encoding a sequence number may be too costly.

In [30], a verification-based methodology used previously for channels over q-ary alphabets with errors [26], is suggested in the setting of channels with deletions, transpostions, and random insertions. The framework borrows heavily from that of low-density parity-check codes. For convenience, let us think of q as being a power of 2, so that symbols are bit strings and exclusive-ors (XORs) are the natural operation. The main ideas are the following: first, we assume each message symbol sent appears to be a random symbol from a q-ary alphabet, by XORing some mutually agreed upon (pseudo)-random value to each symbol. Second, we establish an encoding based on constraints of the form that the XOR of certain collections of symbols take on a given, agreed upon (pseudo)-random value.

To see the benefit of this, suppose each constraint consists of an XOR of six symbols (as in a 3-6 LDPC code). Consider an algorithm that simply considers all combinations of six arrived symbols and checks if the XOR of their values is the value given by some constraint. Since each symbol value appears random, for q sufficiently large the probability that any combination of six received symbols besides the "right" six is small and can be ignored. Specifically, if symbols are uniform over the q-ary alphabet, then the probability a collection of six symbols takes on the constraint value is $1/q$, which in the example setting of 64-bit symbols is 2^{-64}. As long as the number of combinations of six received symbols is substantially less than 2^{64}, the probability of mistakenly identifying the symbols associated with a constraint is quite small. Once symbols are identified, their locations can be determined, and once five symbols corresponding to a constraint are determined, the sixth can be determined as well. A suitable LDPC-style analysis determines the rate achievable using such a scheme.

Using more sophisticated analyses based on analyses of similar LDPC codes, one can show that this verification-based coding can achieve rates arbitrarily close to optimal, at the expense of complexity. As shown in [30], when deletions occur with probability p, a rate of $(1-\varepsilon)(1-p)$ is possible with decoding complexity $n^{O(1/\varepsilon^2)}\log q$.

While the approach of [30] appears far from suitable for real implementations, it is possible that the underlying ideas could prove useful. Metzner has suggested improved techniques using similar ideas that appear efficient enough to be potentially practical [27].

Open Questions:

- Design practical implementations of verification-based decoding for deletion channels or other channels.
- Find ways to lower the probability of error for verification-based decoding so that is performs well for smaller alphabets.

9 Trace Reconstruction

Up to this point, we have been focusing on deletions in the setting of codes, where there is a codebook chosen by the sender and the receiver. To conclude this survey, we consider a different but related class of problems, consisting of natural variations of the problem where we do not control the input. Such problems can generally referred to as *trace reconstruction* problems, and include the following canoncial example. A binary string $X = x_1 x_2 \dots x_n$ yields a collection of *traces* $Y^1, Y^2, \dots, Y^m$, where each Y^i is independently obtained from X by passing X through a deletion channel under which each bit is independently deleted with fixed probability δ. Given the traces (and the value of n and δ), we wish to reconstruct the *original string* X exactly with high probability. The question is how many traces are necessary for reconstruction. Here one can consider the problem over worst-case inputs X, or over random inputs X. Combinatorial variations of trace reconstruction were considered by Levenshtein [21, 22], but study of the above variation was recently initiated by Batu, Kannan, Khanna, and McGregor [2] (see also [17]).

The problem can naturally be extended to consider broader classes of errors, but even the case of only deletions has potential real-world implications. For example, in a sensor network, an array of sensors could be used to record a sequence of events, with each event corresponding to a 1 (positive outcome) or 0 (negative outcome). Noise or mechanical imperfections may cause some sensors to fail to detect each event, giving rise to deletions in each individual sensor's trace. Reconstructing the correct sequence of events from a collection of individually inaccurate sensors where each sensor independently misses each event with fixed probability corresponds to the trace reconstruction problem with deletions. Trace reconstruction also seems natural in biological settings; one may have several samples of DNA sequences from descendants of a common ancestor, and the goal is to reconstruct as well as possible the DNA sequence of the ancestor.

Recently, it has been shown that when X is chosen uniformly at random, there exists a universal constant γ such that for for $\delta < \gamma$, reconstruction is possible with polynomially many traces with high probability (over both X and the traces) [15]. The polynomial can be taken to be independent of δ, and the reconstruction itself also takes polynomial time. Previous work for the case of random X has required sub-constant deletion probabilities (e.g., [2]). However, the reconstruction result for constant γ is purely theoretical, in that the polynomial obtained is prohibitive for a practical implementation. Perhaps the most natural open question in this space is to design a practical algorithm for constant deletion probabilities. However, the study of these problems is still in its infancy, and there are many other related questions to consider.

A natural direction is to consider approximate trace reconstruction, where a small number of errors would be tolerated. Tradeoffs between accuracy and the number of traces required have yet to be explored.

As a different direction, there are natural coding-theoretic analogues of the trace reconstruction problem. Consider a channel consisting of k independent deletion subchannels. That is, the sender's binary message is simultaneously sent through k independent deletion subchannels, with each bit being deleted with probability p in each channel, and the receiver obtains the output from all k subchannels. What is the capacity of this channel? This problem appears unstudied, even in the case where $k = 2$. The corresponding problem with erasure channels is trivial; for the corresponding problem on the binary symmetric error channel, the capacity is easily calculated, and density evolution techniques can be used to design practical near-optimal codes [28, 35]. For deletion channels, the problem appears much more difficult. Maximum likelihood approaches would naturally apply, since the likelihood of an initial string X given two independent outputs Y_1 and Y_2 is given by

$$\mathbf{Pr}(X \mid Y_1, Y_2) = \mathbf{Pr}(Y_1, Y_2 \mid X) \frac{\mathbf{Pr}(X)}{\mathbf{Pr}(Y_1, Y_2)} = \mathbf{Pr}(Y_1 \mid X)\,\mathbf{Pr}(Y_2 \mid X) \frac{\mathbf{Pr}(X)}{\mathbf{Pr}(Y_1, Y_2)}$$

and is therefore proportional to the product of $\#S(X, Y_1) \cdot \#S(X, Y_2)$. (The result extends for larger k.) This brings us back to the problems discussed at the beginning of this survey, namely understanding maximum likelihood on the deletion channel, even in the case where the codewords are assumed to be chosen independently and uniformly at random.

Open Questions:

- Consider the limits of trace reconstruction for random strings and worst-case strings. What is the maximum deletion probability that can be tolerated with a polynomial number of traces? What is the tradeoff between the deletion probability and the number of traces need for reconstruction?
- Extend the results of [15] to channels with insertions, transpositions, and/or other errors.
- Find practical algorithms that perform well on trace reconstruction in practice, and prove rigorous statements about their performance.
- Develop an appropriate framework for approximate trace reconstruction, and prove results in that framework.
- Consider lower bounds for the number of traces required for trace reconstruction. (See [15, 41] for some lower bound results.)

– Prove capacity bounds in the coding setting for deletion channels where multiple independent traces can be obtained.

10 Conclusion

The area of synchronization contains many open problems that appear to hold significant challenges. In particular, while there are has been clear progress on the problem of determining the capacity for the deletion channel, tight bounds remain elusive. An analysis of the maximum likelihood approach appears to be the the biggest prize, potentially dominating the previous bounds obtained. Even an analysis for the case of codewords chosen uniformly at random, which corresponds to natural questions on the behavior of random sequences and subsequences, would be a significant step forward.

While in this survey issues related to efficient coding and decoding have arisen only tangentially, the problem of finding practical codes with good performance is also well worth studying. Some recent efforts in this direction for channels with deletions and possibly other errors include [3, 5, 33], but thus far most of the work in this area remains fairly ad hoc. Clear design principles need to be developed. Perhaps the work done thus far to improve bounds on the capacity can naturally lead to reasonably efficient coding methods.

To conclude, although the problem of coding for the deletion channel and other channels with synchronization errors has been around for decades, it remains a largely unstudied area. Just as the edit distance (or Levenshtein distance) between two strings is a much more complicated construct than the Hamming distance, coding over channels with synchronization errors appears potentially much more complicated than coding over channels with only symbol errors. We expect that efforts to confront this complexity will lead to interesting new results, techniques, and connections between information theory and computer science.

References

1. K. A. S. Abdel-Ghaffar. Capacity per unit cost of a discrete memoryless channel. *IEE Electronic Letters*, 29:142-144, 1993.
2. T. Batu, S. Kannan, S. Khanna, and A. McGregor. Reconstructing strings from random traces. *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 910–918, 2004.
3. J. Chen, M. Mitzenmacher, C. Ng, and N. Varnica. Concatenated codes for deletion channels. *Proceedings of the 2003 IEEE International Symposium on Information Theory*, p. 218, 2003.
4. S.Y. Chung, G.D. Forney, Jr., T.J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 dB ofthe Shannon limit. *IEEE Communications Letters*, 5(2):58–60, 2001.

5. M.C. Davey and D.J.C. Mackay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, 2001.
6. S. Diggavi and M. Grossglauser, On Transmission over Deletion Channels. *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pp. 573-582, 2001.
7. S. Diggavi and M. Grossglauser. On information transmission over a finite buffer channel. *IEEE Transactions on Information Theory*, 52(3):1226–1237, 2006.
8. S. Diggavi, M. Mitzenmacher, and H. Pfister. Capacity Upper Bounds for Deletion Channels. In *Proceedings of the International Symposium on Information Theory*, pp. 1716-1720, Nice, France, June 2007.
9. R. L. Dobrushin. Shannon's Theorems for Channels with Synchronization Errors. *Problems of Information Transmission*, 3(4):11-26, 1967. Translated from *Problemy Peredachi Informatsii*, vol. 3, no. 4, pp 18-36, 1967.
10. A. S. Dolgopolov. Capacity Bounds for a Channel with Synchronization Errors. *Problems of Information Transmission*, 26(2):111-120, 1990. Translated from *Problemy Peredachi Informatsii*, vol. 26, no. 2, pp 27-37, April-June, 1990.
11. R. G. Gallager. Sequential decoding for binary channels with noise and synchronization errors. Lincoln Lab. Group Report, October 1961.
12. E. Drinea and A. Kirsch. Directly lower bounding the information capacity for channels with i.i.d. deletions and duplications. *Proceedings of the 2007 IEEE International Symposium on Information Theory (ISIT)*, pages 1731–1735, 2007.
13. E. Drinea and M. Mitzenmacher. On Lower Bounds for the Capacity of Deletion Channels. *IEEE Transactions on Information Theory*, 52:10, pp. 4648-4657, 2006.
14. E. Drinea and M. Mitzenmacher. Improved lower bounds for the capacity of i.i.d. deletion and duplication channels. *IEEE Transactions on Information Theory*, 53:8, pp. 2693-2714, 2007.
15. T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder. Trace reconstruction with constant deletion probability and related results. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 389–398, 2008.
16. M. Jimbo and K. Kunisawa. An Iteration Method for Calculating the Relative Capacity. *Information and Control*, 43:216-233, 1979.
17. S. Kannan and A. McGregor. More on reconstructing strings from random traces: insertions and deletions. In *Proceedings of the 2005 IEEE International Symposium on Information Theory*, pp. 297–301, 2005.
18. A. Kavcic and R. Motwani. Insertion/deletion channels: Reduced-state lower bounds on channel capacities. *Proceedings of the 2004 IEEE International Symposium on Information Theory*, p. 229.
19. H. Kesten. Random difference equations and renewal theory for products of random matrices. *Acta Mathematica*, 131:207-248, 1973.
20. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics - Doklady*, vol. 10, pp. 707–710, February 1966.
21. V.I. Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001.
22. V.I. Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory, Series A*, 93(2):310–332, 2001.
23. Z. Liu and M. Mitzenmacher. Codes for deletion and insertion channels with segmented errors. *Proceedings of the 2007 IEEE International Symposium on Information Theory (ISIT)*, pages 846–850, 2007.

24. M. Lothaire. Applied Combinatorics on Words. Vol. 105 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2005.
25. M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.
26. M.G. Luby and M. Mitzenmacher. Verification-based decoding for packet-based low-density parity-check codes. *IEEE Transactions on Information Theory*, 51(1):120–127, 2005.
27. J. J. Metzner. Packet-symbol decoding for reliable multipath reception with no sequence numbers. In *Proceedings of the IEEE International Conference on Communications*, pp. 809-814, 2007.
28. M. Mitzenmacher. A note on low density parity check codes for erasures and errors. SRC Technical Note 17, 1998.
29. M. Mitzenmacher. A Brief History of Generative Models for Power Law and Lognormal Distributions. *Internet Mathematics*, vol. 1, No. 2, pp. 226-251, 2004.
30. M. Mitzenmacher. Polynomial time low-density parity-check codes with rates very close to the capacity of the q-ary random deletion channel for large q. *IEEE Transactions on Information Theory*, 52(12):5496–5501, 2006.
31. M. Mitzenmacher. Capacity bounds for sticky channels. *IEEE Transactions on Information Theory*, 54(1):72, 2008.
32. M. Mitzenmacher and E. Drinea. A simple lower bound for the capacity of the deletion channel. *IEEE Transactions on Information Theory*, 52:10, pp. 4657-4660, 2006.
33. E. Ratzer. Marker codes for channels with insertions and deletions. *Annals of Telecommunications*, 60:1-2, p. 29-44, January-February 2005.
34. T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, 2001.
35. T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.
36. L.J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, 1999.
37. C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379–423, 1948.
38. N.J.A. Sloane. On single-deletion-correcting codes. *Codes and Designs: Proceedings of a Conference Honoring Professor Dijen K. Ray-Chaudhuri on the Occasion of His 65th Birthday, Ohio State University, May 18-21, 2000*, 2002.
39. J. D. Ullman. On the capabilities of codes to correct synchronization errors. *IEEE Transactions on Information Theory*, 13 (1967), 95-105.
40. R. R. Varshamov and G. M. Tenengolts. Codes which correct single asymmetric errors. *Automation and Remote Control*, 26:2, pp. 286-290, 1965. Translated from *Automatika i Telemekhanika*, 26:2, pp. 288-292, 1965.
41. K. Viswanathan and R. Swaminathan. Improved string reconstruction over insertion-deletion channels. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 399–408, 2008.
42. N. D. Vvedenskaya and R. L. Dobrushin. The Computation on a Computer of the Channel Capacity of a Line with Symbol Drop-out. *Problems of Information Transmission*, 4(3):76-79, 1968. Translated from *Problemy Peredachi Informatsii*, vol. 4, no. 3, pp 92-95, 1968.
43. K. S. Zigangirov. Sequential decoding for a binary channel with drop-outs and insertions. *Problems of Information Transmission*, vol. 5, no. 2, pp. 17–22, 1969. Translated from *Problemy Peredachi Informatsii*, vol. 5, no. 2, pp 23–30, 1969.