More on Reconstructing Strings from Random Traces: Insertions and Deletions

Sampath Kannan
Dept. Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104, USA
Email: kannan@cis.upenn.edu

Andrew McGregor
Dept. Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104, USA
Email: andrewm@cis.upenn.edu

***Abstract*— We are given a collection of m received strings or *traces* that have been independently generated by randomly inserting and deleting bits from a common string t of length n. Our goal is to reconstruct the string t from these observed traces. This paper considers both the algorithms for doing this reconstruction and seeks to understand the error rates at which reconstruction is possible. Note the difference from the typical coding theory scenario - rather than trying to infer a codeword from a single received word, we are interested in inferring an arbitrary (or near arbitrary) word from multiple, independently generated received words. We present two main results. Firstly we show that for almost all transmitted strings, if the deletion/insertion error probability is $O(1/\log^2 n)$ then with $m = O(\log n)$ traces we can exactly reconstruct the transmitted string with high probability. Furthermore we can still reconstruct in the presence of additional noise that flips each bit with constant probability. Secondly, for arbitrary strings (with no run of length $> n^\epsilon$) we show that with a constant number of received strings we can reconstruct when the deletion/insertion probability is $O(1/n^{1/2+\epsilon})$. This paper continues work initiated in Batu et. al. (2004) which considered only deletion errors.**

Our setting can be viewed as the study of an idealized biological evolutionary process where the DNA string undergoes point mutations, deletions and insertions. Our goal is to understand at what mutation rates, a small number of observed samples can be correctly aligned to reconstruct the parent string.

I. Introduction

Let $t = t_1 t_2 \dots t_n$ be a length n binary string. Suppose we transmit this string over a channel that randomly distorts t such that at each step, either a random bit is received with probability q, the next bit of t is deleted with probability q or the next bit of t is received (possibly being flipped with probability p). An exact specification of this probabilistic channel is given in the next section. We do m such independent transmissions and generate m strings, $r_1, \dots r_m$. How do we reconstruct t from $r_1, \dots r_m$ and how big does m need to be to ensure that we can correctly reconstruct t with high[1] probability?

When communicating a bit string the sender usually has the option of encoding her information in a code designed to be resilient to the errors that will be introduced by the channel. While the majority of the theory of binary codes have been concerned with the binary symmetric channel, there have been codes designed for other channels. For example various error correcting codes for the deletion channel have been studied (e.g., [6], [5], [7], [10]). Such codes allow one to reconstruct the transmitted string (from a single observation) when the transmitted string is actually a *codeword*. So, a decoding algorithm for such an error correcting code can be viewed as an algorithm to solve the problem stated above when t is restricted to belonging to a particular (and exponentially small) subset of all possible length n binary strings. In the problem above, however, we wish to successfully reconstruct any transmitted string.

For some channels, or error models, our problem is easy. For example, if only bit flips occur then the ith bit of the transmitted string t can be deduced by taking the majority of the ith bits of the received strings. Even for constant bit flip probability, $O(\log n)$ transmissions suffices with high probability. A similar approach works when the received strings are generated by the erasure channel. However, when the channel can insert and delete bits, as is possible in the our channel, it is no longer clear which bits in the received string should be used to deduce the ith bit of the transmitted string.

There have been numerous similar problems considered in the literature but almost all of these consider received strings generated by combinatorial channels defined by the maximum number of errors of a given type that they can introduce. These include [8], [9] which considers how many distinct traces are necessary to uniquely infer the original string and [3] in which we start with all the substrings (of a given length) and their multiplicities. We are, however, interested probabilistic channels and how many transmissions are required such that, with high probability, we can reconstruct the transmitted string. Note that probabilistic channels are considered in a section of [9] but that the channels considered do not include the on described above.

Firstly we show that for almost all transmitted strings, if the deletion/insertion error probability is $q = O(1/\log^2 n)$ then with $m = O(\log n)$ traces we can exactly reconstruct the transmitted string with high probability. Furthermore we can still reconstruct in the presence of additional noise that flips each bit with constant probability p. Secondly, for arbitrary strings (with no run of length $> n^\epsilon$) we show that with a constant number of received strings we can reconstruct when

[1] Here, and throughout this paper, *high probability* means that the probability can be made arbitrarily close to 1 for sufficiently large values of n.

the deletion/insertion probability is $q = O(1/n^{1/2+\epsilon})$. This paper continues work initiated in [2] which considered only deletion errors. The results presented in the following theorem.

Theorem 1.1 ([2]): There exist algorithms such that

1) For all but a polynomially small fraction of possible strings t, reconstruct the string t from $m = O(\log n)$ received strings if the deletion probability $q = O(1/\log n)$ (with high probability.)
2) For an arbitrary strings t, reconstruct a string that has identical structure to t from $m = O(1/\epsilon)$ received strings if the deletion probability $q = O(1/n^{1/2+\epsilon})$ (with high probability.)

There are numerous rationales for our setting in which the information represented by t is not encoded before errors are introduced by the channel. For example our setting can be viewed as the study of an idealized biological evolutionary process where the DNA string undergoes point mutations, deletions and insertions. We would like to understand at what mutation rates, a small number of observed samples can be correctly aligned to reconstruct the parent string.

A primary motivation for our problem comes from computational biology, in particular, the multiple sequence alignment problem. In a typical biological scenario, we observe related DNA or protein sequences from different organisms. These sequences are the product of a random process of evolution that inserts, deletes, and substitutes characters in the sequences. The multiple sequence alignment problem is commonly used to deduce conserved subpatterns from a set of sequences known to be biologically related [4]. In particular, one would like to deduce the common ancestor of these related sequences. In reality, each of the observed sequences is not produced independently by this evolution process. Sequences from organisms that are evolutionarily very closely related undergo common evolution (identical changes) for the most part and only diverge and undergo independent evolution for a small period of time.

The multiple sequence alignment problem is one of the most important problems in computational biology and is known to be NP-hard. An *alignment* of k strings is obtained by inserting spaces into (or at either end of) each string so that the resulting strings have same length, say, l. Then, the strings are put in an array with k rows of l columns each. Typically, a score is assigned to an alignment to measure its quality. Different scoring schemes are proposed in the literature. In one common family of schemes, the score of an alignment is taken to be the sum of the scores of the columns; the score of a column is defined as some function of the symbols in the column. In standard versions, this function has a high value when all the symbols in the column agree and its value drops off as there is greater and greater variation in the column. The objective is to find an alignment with the maximum score. Note that in the case of related sequences, it is not clear how these scoring schemes serve the purpose of discovering the common ancestor, from which each sequence is generated. In fact, it is easy to construct examples where the optimum alignment will not produce the common ancestor.

Note: We assume that the transmitted string is a binary string because strings from a larger alphabet can actually be inferred more easily. Specifically, if the actual transmitted string comes from an alphabet Σ, one can consider $|\Sigma|$ different mappings from Σ to $\{0,1\}$, each of which maps exactly one letter in Σ to 1, solve the induced inference problems on $\{0,1\}$-sequences and from these solutions reconstruct the solution for the original problem.

II. Problem and Definitions

We start with a length n binary transmitted string t. We transmit this string m times over a channel $\mathcal{C}$ that introduces errors. In this way we generate m received strings $r_1, \ldots r_m$. The channel we consider is a binary channel with acts on each bit independently as follows:

$$\begin{aligned} \mathcal{C} : \mathrm{F}_2 &\rightarrow \cup_{0 \le v} \mathrm{F}_2^v \\ b &\rightarrow Sg(b) \end{aligned}$$

where S a random binary string of length k, k is a random variable distributed as a Geometric random variable with parameter q_2, and

$$g(b) = \begin{cases} b & \text{with probability } 1 - p - q_1 - pq_1 \\ \bar{b} & \text{with probability } p(1-q_1) \\ \emptyset & \text{with probability } q_1 \end{cases}$$

where $\emptyset$ donates the empty string and $\bar{b}$ denotes bit-flipping, i.e. $\bar{0} = 1$ and $\bar{1} = 0$. Note that $q_1 = q_2 = 0$ corresponds to the channel BSC_p and $q_2 = p = 0$ corresponds to a deletion channel with deletion probability q_1. In what follows $q_1 = q_2$ so we will denote $q := q_1 = q_2$. Let $f_i(s)$ be the binary string output by the channel on the ith transmission of a substring s of t.[2] In particular $r_i = f_i(t)$.

III. Almost All Transmitted Strings

In this section we consider the error probabilities $0 \le p < 1/2$ and $q = O(1/\log^2 n)$. We wish to show that for all but an arbitrary small fraction of transmitted strings t, $m = O(\log n)$ transmissions is sufficient to exactly reconstruct t (with high probability.) To prove this our analysis will assume that t is chosen at random for all length n binary strings. Then, we show that probability (over the random choice of t *and* the random noise introduced by the channel) of not reconstructing t exactly can be made arbitrarily small by ensuring that q is less than some small constant fraction of $1/\log^2 n$ and that m is bigger than some large constant multiple of $\log n$. Integral to our analysis is that fact that, because t can be viewed as a random string, all substrings of length longer that $\Omega(\log n)$ have large Hamming distance from each other. In what follows we show that they can therefore be used to help align the received strings. We start with some necessary definitions.

Definition 3.1: We say two strings of equal length are a *semi-match* if the Hamming distance between them is less

[2]We use the convention throughout the paper that a substring of a string is a subsequence of t in which all the bits are contiguous. Furthermore we denote as $t[i,j]$, the substring of a string t starting in position i of t and ending at position j of t.

than $p - p^2 + 1/4$. Imagine that the bits of the original string t are tagged with indices 1 through n. We say that a length l substring s of r_i and a length l substring s' of r_j are *non-overlapping* if there does not exist a k, $1 \leq k \leq l$ such that the kth bit of s has the same tag as the kth bit of s'.

Our algorithm works by first finding a length l substring a of r_1 with the following properties:

1) There exists a length l substring t^m of t, ie. $t = t^1 t^m t^2$ such that $a = f_1(t^m)$.
2) Each received string r_i can be split into $r_i = r_i^1 m_i r_i^2$ where $r_i^1 = g^r(f_i(t^1))$ where m_i is a semi-match to a and g^r is function that inserts or deletes at most l bits from the right side. Unless g^r is the identity function, we call the right end of r_i^1 a *dirty end*. Similarly with $r_i^2 = g^l(f_i(t^2))$.
3) t^m = average of $(m_i)_{2 \leq i \leq m}$ and a. (By "average" we mean the string whose jth bit takes the same value as the majority of the jth bits of the strings $(m_i)_{2 \leq i \leq m}$ and a.)

We call such a string an *anchor*. Once an anchor has been found we recurse on the strings either side of the anchor, ie. we now try to deduce t_1 from $r_1^1, \ldots r_m^1$ and t_2 from $r_1^2, \ldots r_m^2$. To find an anchor we consider the middle kl bits of $r = r_1$. This middle section consists of k potential length l anchors, $a_1, ...a_k$.

Lemma 3.1 (Anchors can be reliably located): The probability that there exists an a_i that is falsely identified as an anchor is less than $kne^{-l(1/2-2p+2p^2)/4}$. The probability that none of the a_i are anchors is less than,

$$\left(mql + m \left[\frac{e^\delta}{(1+\delta)^{1+\delta}} \right]^{(2p-2p^2)l} \right)^k ,$$

where $\delta = \frac{1}{2} - \frac{1}{8(p-p^2)}$.

Proof: The probability that a_i is falsely identified as an anchor is bounded above by the probability that a_i has a semi-match with a non-overlapping substring in r_j for some $j \neq 1$. Since the strings are non-overlapping, the probability of them being a semi-match is $< e^{-l(1/2-2p+2p^2)/4}$ by an application of the Chernoff bound. Applying the union bound gives the first result.

The probability that a_i is an anchor is at least the probability that the following both hold true,

1) no insertion/deletion errors occured in the m transmissions of $f^{-1}(a_i)$
2) no more than $p - p^2 + 1/4$ bit-flip errors occurred in each of the m transmissions of $f^{-1}(a_i)$

The probability that an insertion or deletion error occurred among the m transmissions of $f^{-1}(a_i)$ is less than qml. Using the Chernoff bound, the probability that more than $p - p^2 + 1/4$ bit-flip errors occurred in any of the m transmissions of $f^{-1}(a_i)$ can be bounded above by

$$\left[\frac{e^\delta}{(1+\delta)^{1+\delta}} \right]^{(2p-2p^2)l}$$

, where $\delta = \frac{1}{2} - \frac{1}{8(p-p^2)}$. Hence, the probability that there does not exist an anchor among the k possible anchors is $(mql + m \left[\frac{e^\delta}{(1+\delta)^{1+\delta}} \right]^{(2p-2p^2)l})^k$ as the event of each a_i being an anchor is independent. ∎

We recurse until the strings are of length $3kl$. The probability that we have erred up until this point when identifying anchors is less than,

$$n \max \left\{ \left(mql + m \left[\frac{e^\delta}{(1+\delta)^{1+\delta}} \right]^{(2p-2p^2)l} \right)^k , \; kne^{-l(1/2-2p+2p^2)/4} \right\} ,$$

where $\delta = \frac{1}{2} - \frac{1}{8(p-p^2)}$. We can make this quantity arbitrarily small by setting $l = O(\log n)$, $k = O(\log n)$ and $q = O(1/\log^2 n)$ and choosing the constants appropriately

We now argue that for each segment t^i of t, the majority of the strings being used to infer t^i differ from t^i only as a result of bit-flips. First note that,

1) Dirty ends are rare, ie. the probability that a given string has a dirty end is bounded above by $2q$
2) Each string has a probability $3klq$ of actually having had bits deleted or inserted by the channel

and that each of these probabilities is a small constant for our chosen values of l, k and q. Hence, since $m = O(\log n)$, using the Chernoff bound we can show that with probability $\geq 1 - \frac{1}{n^2}$, the majority of the corrupted strings being used to deduce a given segment of t are only corrupted by bit-flip errors. We therefore conclude that this is that case for all the $O(n)$ segments of t.

We have now proved the main theorem of this section.

Theorem 3.1: For all but an arbitrarily small fraction of possible strings t, reconstructing the string t from $m = O(\log n)$ received strings if the probability of insertion or deletion is $q = O(1/\log^2 n)$ (with high probability.)

IV. Arbitrary (no long runs) Transmitted Strings

The technique for reconstructing arbitrary strings will be to find a set of *promises* that will be kept with high probability during the transmission of the m strings. Given these promises, we embark on a case analysis to verify that the transmitted string will be correctly reconstructed given these promises. Now we consider the error probabilities $p = 0$ and $q = O(1/n^{1/2+\epsilon})$. We start with some definitions.

Definition 4.1 (Runs and Alternating Sequences): A *run* of 1's (also called a *1-run*) in a string t is a maximal substring of consecutive 1's. A run of 0's is defined analogously. We denote the ith run of string t by L_i and the length of L_i by l_i. A run is called *long* if its length is at least n^ϵ and is *short* otherwise. An *alternating sequence* in a string is a sequence of length at least two such that each run in the sequence has length 1 (e.g., $010101\ldots$). A bit is called the *first bit* of an alternating sequence if it is a run of length 1 that either follows a run of length greater than 1 or it is the first bit in the transmitted string. A *delimiting run* for an alternating sequence is the first run of length at least two that follows the alternating sequence.

It should be noted that it is not possible to reconstruct some transmitted strings with only a small number of transmissions. It was noted in [2] that distinguishing between two strings, one that starts with $n/2$ 1's followed by $n/2$ 0's and one that starts with $n/2+1$ 1's followed by $n/2-1$ 0's, requires $\Omega(nq(1-q))$ transmissions even if there are only deletions. When there were only deletions, most strings that necessitate many transmissions did so because they had long runs. When there are both insertions and deletions, alternating sequences can also cause trouble. In what follows we make one restriction about how arbitrary a transmitted string may be - it must not contain any long runs. We also will only estimate the length of alternating sequences that are longer that $\sqrt{n}$.

We now list the promises that we will assume are kept in the course of the transmissions. The proof of this lemma is largely straight forward.

Lemma 4.1 (Transmission Promises Lemma): A collection of $m = O(1)$ received strings generated by deleting and inserting from t with probability q, with high probability satisfies the following:

(P1) The first bit in the transmitted string is the first bit in each received string.

(P2) Among all the m transmissions, at most 1 error occurred in the transmission of any 4 consecutive runs. (In particular, no two consecutive bits are inserted or deleted.)

(P3) For all maximal alternating sequence of length $l > \sqrt{n}$, if an error occurs at the start of the alternating sequence (in any of the m transmissions) then in all the m transmissions, there are no errors during the transmission of the final $\log n\sqrt{l}$ bits of the maximal alternating sequence and the next two bits of the delimiting run.

(P4) For all maximal alternating sequence, if an error occurs at the start of the alternating sequence (in any of the m transmissions) then in all the m transmissions, there are no errors during the transmission of the final n^ϵ (or the rest of the alternating sequence if the length of the alternating sequence is less than n^ϵ) bits of the maximal alternating sequence and the next two bits of the delimiting run.

(P5) For each $\sqrt{n}$ substring x of t,

$$\{i \in [m] : f_i(x) = x\} > m/2$$

i.e. in the majority of transmissions, x is transmitted without errors.

(P6) For each substring x of t of length $> n^\epsilon$, for each i, there are fewer than $q|x|\log n$ errors in the ith transmission of $|x|$.

Our algorithm for determining the transmitted string processes the received strings from left to right and will maintain a pointer p_i to a bit in each of received string r_i. Let $r_i^* = r_i[p_i, |r_i|]$, the suffix of r_i starting with the bit pointed to by p_i. Initially the pointers point to the first bit of their respective received strings, ie. initially $r_i^* = r_i$. There will exist the following invariant regarding the positions of each pointer:

Invariant: *In each received string we are pointing to the first bit transmitted from the same run, ie. for some j, all p_i point to the left most bit of $f_i(L_j)$.*

In what follows we assume that the transmission promises listed in the above lemma are kept. By P1, we start with the invariant holding true. Assume that the invariant holds and that each pointer points to the first bit of $f_i(L_j)$. Consider the next $\sqrt{n}$ bits to the right of each pointer. By P5, the majority of these are equal and correspond to the length $\sqrt{n}$ substring of t starting with the jth run. Call this substring t'. We therefore learn the length of the runs in the next $\sqrt{n}$ bits of t. What we have to do now is update the pointers to ensure that we maintain the invariant. Without loss of generality let the first bit of t' be 1. Let the length of the first run be y. Now consider r_i^* and let x^i be the length of the first run assuming it is a 1-run (and $x^i = 0$ otherwise.) Making considerable use of P2, we can make various deductions about the transmissions

1) If $x^i = y$, we conclude that no errors have occurred in the ith transmission of L_j.
2) If $x^i = y+1$, we conclude that either one "1" was inserted in the ith transmission of L_j or that, on the condition that $l_{j+1} = l_{j+2} = 1$, one "0" was deleted from L_{j+1}.
3) If $x^i > y+1$, we conclude that one "0" was deleted in the ith transmission of L_{j+1}.
4) If $x^i = y-1$, we conclude that either one "1" was deleted in the ith transmission of L_j or that, on the condition that $l_{j+1} = l_{j+2} = 1$, one "0" was inserted before the last bit of L_j was transmitted.
5) If $x^i < y-1$, we conclude that one "0" was inserted into L_j.

If, $x^i = y$ for all received strings, then we move the pointers in each received string y positions to the right. Note that there can be at most one received string r_{error} such that $x^{\text{error}} \neq y$. If $x^{\text{error}} > y+1$, we also move the pointer in each received string y positions to the right. If $x^{\text{error}} < y-1$, we more the pointer in r_{error}, $y+1$ positions to the right and for all other received strings, we move the pointer y positions to the right.

The two remaining cases, $x^{\text{error}} = y-1$ or $y+1$ are more tricky. As we noted earlier, these cases can only arise if $l_{j+1} = l_{j+2} = 1$, ie. if there is an alternating sequence. We will attempt to update the pointers p_i to point to $f_i(L_k)$ where L_k is the delimiting run of the alternating sequence.

From t' we know that either the length of this alternating sequence is either $< \sqrt{n}$ or $\geq \sqrt{n}$. We consider each case in turn and conclude that we can correctly update the pointers. Firstly if the length is $\geq \sqrt{n}$ then we know by P3 that in each received string, there exists a length ≥ 2 run corresponding to the delimiting run. Further more, by P3 and P6 we can ensure that we do not incorrectly identify the delimiting run in a received string by requiring that the identified runs lie in close proximity in the received strings. We can bound the area in which we expect to see the delimiter and guarantee that no other errors occur in this region that might result in a substring in the received string that could be mistaken for the delimiter.

Secondly if the length is $< \sqrt{n}$ then we know by P4 that if there are any errors in the transmission of the alternating

sequence, then there are no errors during the transmission of the last bit of the alternating sequence and the next run. Hence in r_{error} the delimiter is transmitted intact. Again, appealing to P6 ensures that we do not incorrectly identify the delimiter because we can bound the region where we expect to find this delimiter (by bounding the number of errors we expect to see in the alternating sequence) and guarantee that no other errors occur in this region that might result in a substring in the received string that could be mistaken for the delimiter.

We have now proved the main theorem of this section.

Theorem 4.1: If all runs are of length $< n^{\epsilon}$, we can reconstruct t from $m = O(1)$ received strings if the probability of insertion or deletion is $q = O(1/n^{1/2+\epsilon})$ (with high probability.)

V. Conclusion and Future Work

We have considered the problem of reconstructing strings from traces generated by a probabilistic channel. Note that while the majority of strings are very easy to reconstruct, we still do not have a good technique for reconstructing arbitrary strings and this clearly merits further work. Even for the majority of "easy" strings, we have thus far only considered error rates that decrease as n increases. It would be desirable to be able to cope with constant deletion and insertion rates. Finally, in regard to some of the biological motivation for this problem, being able to relax the assumption of total independence in the generation of the received strings would have benefits when it came to applying this work in practice.

Acknowledgments

Sampath Kannan was supported by NSF CCR98-20885 and NSF CCR01-05337. Andrew McGregor was supported by NSF ITR02-05456.

References

[1] N. Alon, J. Edmonds, and M. Luby, *Linear time erasure codes with nearly optimal recovery*, 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 512–519.

[2] T. Batu, S. Kannan, S. Khanna, and A. McGregor, *Reconstructing strings from random traces*, Proc. of SODA 2004 (2004), 903–911.

[3] M. Dudik and L. J. Shulman, *Reconstruction from subsequences*, J. Combinatorial Theory Series A (2003), no. 103, 337–348.

[4] D. Gusfield, *Algorithms on strings, trees, and sequences*, Algorithms on Strings, Trees, and Sequences, 1997.

[5] V. I. Levenshtein, *Binary codes capable of correcting spurious insertions and deletions of ones*, Problems of Information Transmission **1** (1965), no. 1, 8–17.

[6] ———, *Binary codes capable of correcting deletions, insertions and reversals*, Soviet Physics Dokl. **10** (1966), no. 8, 707–710.

[7] ———, *On perfect codes in the deletion/insertion metric*, Discrete Mathematics and Applications **3** (1992), no. 1, 241–258.

[8] ———, *Efficient reconstruction of sequences from their subsequences and supersequences*, J. Combinatorial Theory Series A (2001), no. 93, 310–332.

[9] ———, *Efficient reconstruction of sequences*, IEEE Trans. Inform. Theory **47** (2001), no. 1, 2–22.

[10] L. J. Shulman and D. Zuckerman, *Asymptotically good codes correction insertions, deletions and transpositions*, Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms **45** (1999), no. 7, 2552–2557.