A Compression Algorithm Using Mis-aligned Side-information[1]

Nan Ma, Kannan Ramchandran and David Tse

Wireless Foundations, Dept. of Electrical Engineering and Computer Sciences
University of California at Berkeley

***Abstract*—We study the problem of compressing a source sequence in the presence of side-information that is related to the source via insertions, deletions and substitutions. We propose a simple algorithm to compress the source sequence when the side-information is present at both the encoder and decoder. A key attribute of the algorithm is that it encodes the edits contained in runs of different extents separately. For small insertion and deletion probabilities, the compression rate of the algorithm is shown to be asymptotically optimal.**

I. Introduction

In [1], we have studied the problem of compressing a source sequence with the help of mis-aligned decoder-only side-information, where the source and side-information are the input and output of a deletion channel, respectively. The minimum rate is shown to correspond to the amount of information in the deleted content plus the locations of the deletions, minus the uncertainty in the locations given the source and side-information. We refer to the latter as "nature's secret". This is the information that the encoder and decoder can never find out. It represents the over-counting of information in the locations of the deletions. For example, if the input and output of a deletion channel and are $(0, 0)$ and (0), the encoder and decoder will never know and never need to know whether the first or the second bit is deleted. An interesting question is: how to construct a practical compression algorithm with the optimal compression rate, where the encoded bits do not reveal "nature's secret"? In this paper we provide such a construction for a simpler problem where the side-information is available at both the encoder and decoder. Although the availability of the side-information is changed, the minimum rate remains the same.

In this paper, we study the problem of compressing a source sequence, $\mathbf{X}$, with the help of side-information, $\mathbf{Y}$, which is available at both the encoder and the decoder. The side-information is related to the source via insertions, deletions and substitutions. See Figure 1 for an illustration of the system. The objective of this work is to construct an encoding/decoding algorithm to achieve the optimal compression rate defined as the minimum number of encoded bits per source bit.

[1]This material is based upon work supported by the US National Science Foundation (NSF) under grants 23287 and 30149 and by a gift from Qualcomm Inc.. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

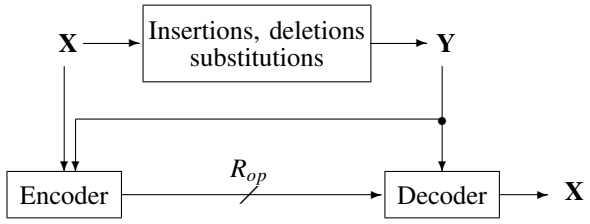

Fig. 1. *Structure of the system*

Here is an example of the source and side-information:

$$\mathbf{X} = (0, 0, 1, 1, 0, 1)$$
$$\mathbf{Y} = (0, 1, 0, 0, 1, 1)$$

In order to compare these two sequences, we can insert some gaps, which are denoted by '$-$', to align them as follows.

$$\mathbf{X}^* = (0, 0, 1, 1, 0, 1, -)$$
$$\mathbf{Y}^* = (0, -, 1, 0, 0, 1, 1)$$

This alignment explains the $\mathbf{X}$ with respect to $\mathbf{Y}$ with an insertion, a substitution and a deletion: X_2 is inserted between Y_1 and Y_2; X_4 substitutes Y_3; Y_6 is deleted. The encoder needs to describe the above editing information using the minimum number of bits.

The problem of synchronizing edited sequences has been studied by [2]–[4] assuming the number of edits is a constant that does not increase with the length of the sequence. Upper and lower bounds on the minimum number of encoded bits were provided as functions of the number of edits and the length of the sequence. In [5], an interactive, low-complexity and asymptotically optimal scheme was proposed. In comparison, in this paper, we consider the case that a fraction of source bits, rather than a constant number of bits, is edited, which makes the problem more general. There are also practical synchronization algorithms. such as RSYNC [6] for generic files and VSYNC [7], which targets video applications. In the special case when the source and the side-information differ only by substitutions (side-information is aligned), a universal compression algorithm has been proposed by [8].

In this paper, we propose a simple compression algorithm, for which the compression rate is asymptotically optimal when the editing probability is small. The key ideas are: (1) describing the locations of insertions and deletions by

978-1-4673-2579-0/12/$31.00 ©2012 IEEE

specifying the runs[2] of side-information in which they appear, and (2) separately encoding the edits that appears in runs of different extents. To explain idea (1), consider the example where the side-information is $\mathbf{Y} = (0, 0, 1, 0)$ and the source is $\mathbf{X} = (0, 1, 0)$. Neither the encoder nor the decoder knows whether the first bit or the second bit is deleted. Therefore the encoder needs to describe the location of the deletion only up to a run, which consists of the first two bits in this example, but not further. To explain idea (2), consider the example where the side-information $\mathbf{Y} = (0, 0, 1, 0)$ and the source is $\mathbf{X} = (0, 1)$. These sequences can be explained by two deletions, in the first run and the third run of $\mathbf{Y}$, respectively. If the deletion process is memoryless and stationary, the longer first run is more likely to contain a deletion than the shorter third run. Therefore the two deletion events should be encoded separately, using entropy coders with different target distributions, or using a universal entropy coder.

Our compression algorithm can find applications in a number of settings, for example, to compress genomic sequences, as in [9].[3] The difference between the genomic sequences from two individuals of the same species is a small fraction of a whole sequence, and is in the form of insertions, deletions and substitutions. If one of the genomic sequences can be used as side-information, the algorithm can be used to compress the other sequence. The algorithm can also be used in distributed file backup or file sharing systems, where different source nodes have different versions of the same file differing by a small number of edits including insertions, deletions and substitutions. Here, an old version can be used as side-information that is mis-aligned to the new version of the same file.

The rest of this paper is organized as follows. In Section II we formally setup the problem. In Section III we consider a simple case where the source sequence is obtained from side-information by pure deletion. We present the algorithm and analyze the performance. In Section IV we present the algorithm in the general setup. All the proofs are provided in [10] due to the page limit.

Notation: Symbols in boldface represent sequences or matrices, and the symbols in non-boldface represent scalars. The binary entropy function is denoted by $h_2(\cdot)$. The notation $\{0,1\}^n$ denotes the n-fold Cartesian product of $\{0, 1\}$, and $\{0,1\}^*$ denotes $\left(\bigcup_{k\in\mathbb{Z}^+}\{0,1\}^k\right)\bigcup\{\emptyset\}$.

II. Problem Setup

We will define two sequences $\mathbf{X}$ and $\mathbf{Y}$, which differ by insertions, deletions, and substitutions.

First, consider an auxiliary length-n sequence $\mathbf{Z}_X = (Z_{X,1}, \ldots, Z_{X,n}) \in \{0,1\}^n \sim$ iid Bernoulli(p), where $p \in (0, 1)$. Pass $\mathbf{Z}_X$ through a binary symmetric channel with crossover probability q to get $\mathbf{Z}_Y$.

[2]A run is the maximal length sequence of a repeated symbol. The extent, or length, of a run is the number of times the symbol repeats.

[3]We would like to thank Dr. Tsachy Weissman for introducing us to this application.

We will then make deletions in $\mathbf{Z}_X$ and $\mathbf{Z}_Y$ to construct $\mathbf{X}$ and $\mathbf{Y}$, respectively. Let the deletion pattern $\mathbf{D}_X$ be a length-n sequence $\sim$ iid Bernoulli(d_X), which is independent of $\mathbf{Z}_X$ and $\mathbf{Z}_Y$. The deleted sequence $\mathbf{X} \in \{0,1\}^*$ is a subsequence of $\mathbf{Z}_X$, which is derived from $\mathbf{Z}_X$ by deleting all those $Z_{X,i}$'s with $D_{X,i} = 1$. Similarly, the deletion pattern $\mathbf{D}_Y \sim$ iid Bernoulli(d_Y) describes the deletion process from $\mathbf{Z}_Y$ to $\mathbf{Y}$.

Since the editing process from $\mathbf{Z}_X$ to $\mathbf{X}$ is a deletion process, the inverse process from $\mathbf{X}$ to $\mathbf{Z}_X$ can be regarded as an insertion process. Therefore from $\mathbf{X}$ to $\mathbf{Y}$ there are insertions (from $\mathbf{X}$ to $\mathbf{Z}_X$), substitutions (from $\mathbf{Z}_Y$ to $\mathbf{Z}_Y$) and deletions (from $\mathbf{Z}_Y$ to $\mathbf{Y}$).

Both sequences $\mathbf{X}$ and $\mathbf{Y}$ are available to the encoder and $\mathbf{Y}$ is available only to the decoder as side-information. All the other sequences, $\mathbf{Z}_X$, $\mathbf{Z}_Y$, $\mathbf{D}_X$, and $\mathbf{D}_Y$ are available to neither the encoder nor the decoder. The encoder encodes $\mathbf{X}$ in the presence of $\mathbf{Y}$ and sends a bit string of variable length to the decoder so that the decoder can reproduce $\mathbf{X}$ without any error. The sequences $\mathbf{X}$ and $\mathbf{Y}$ are called the source sequence and the side-information, respectively. Please see Fig. 2 for the structure of the system together with the source model.

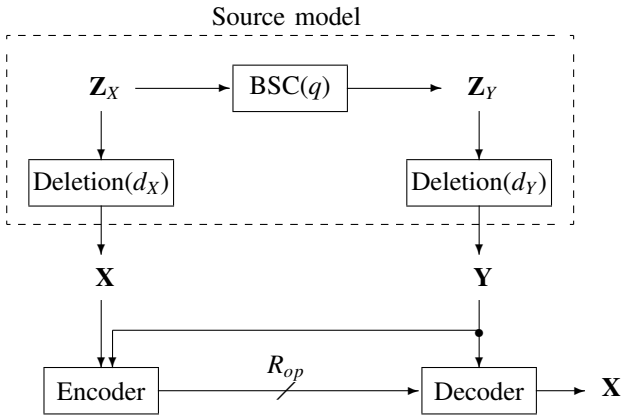

Fig. 2. *Structure of the system with the source model*

The performance of the encoder and the decoder is measured by the expected operational rate, which is defined as $R_{op} := \lim_{n\to\infty} \mathbb{E}[L_M/L_Y]$, where L_M is the length of encoded bit string, and L_Y is the length of Y.

If there is no computational constraint, the generally optimal approach for this conditional compression problem is to enumerate all possible source sequences $\mathbf{x}$'s for each realization of the side-information sequence $\mathbf{y}$ and compute $p_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y})$, and then construct a Huffman code to encode the $\mathbf{x}$'s. However, the complexity is exponential in n and hence impractical in most applications. The objective of this work is to find a practical encoder and decoder which minimize the expected operational rate.

III. Algorithm for the Pure Deletion Case

In order to provide a clear presentation of our algorithm, we start by considering a special case of the general problem, where the source sequence $\mathbf{X}$ is derived from the side-information $\mathbf{Y}$ only by deletion, but not substitution or in-

sertion. Formally speaking, $q = 0$ and $d_Y = 0$, which imply $\mathbf{Z}_X = \mathbf{Z}_Y = \mathbf{Y}$. For the sake of simplicity, in this section, we drop the subscript X in d_X and $\mathbf{D}_X$ and denote them as d and $\mathbf{D}$, respectively.

A. Algorithm for pure deletion

The encoder has the following three stages.

1) Alignment: In this stage we insert some gaps in $\mathbf{X}$ to get $\mathbf{X}^*$, which has the same length as $\mathbf{Y}$. The following greedy alignment algorithm described in [11, Section 3.1] is used.
   Read $\mathbf{X}$ and $\mathbf{Y}$ from left to right. Take the first bit of $\mathbf{X}$, and match it with the leftmost appearance of this bit in $\mathbf{Y}$; then take the second bit of $\mathbf{X}$, and match it with the subsequent leftmost appearance of this bit in $\mathbf{Y}$; and so on. All the bits in $\mathbf{Y}$ that are not matched with bits from $\mathbf{X}$ are matched with gaps denoted by '$-$'. Let $\mathbf{X}^*$ be the aligned version of $\mathbf{X}$ with gaps inserted. The alignment implies a reconstructed deletion pattern $\widehat{\mathbf{D}}$, which can explain the deletion process from $\mathbf{Y}$ to $\mathbf{X}$, but is in general different from $\mathbf{D}$.
2) Describing the deletions with respect to runs:
   Let the maximum extent of the runs in $\mathbf{Y}$ be $L_{\max}$. For IID sequence $\mathbf{Y}$, $\mathbb{E}[L_{\max}] = \Theta(\log n)$ [12].
   The encoder performs the following:
   - For $l = 1, \ldots, L_{\max}$, do:
     - Compute U_l, the number of runs of extent l in $\mathbf{Y}$.
     - For $i = 1, \ldots, U_l$, compute $\widehat{V}_{l,i}$, the number of deletions in the i-th run of extent l in $\mathbf{Y}$ according to $\widehat{\mathbf{D}}$.
3) Entropy coding: For each $l = 1, \ldots, L_{\max}$, compress the sequence $\{\widehat{V}_{l,i}\}_{i=1}^{U_l}$ using an entropy coder. Note that $\widehat{V}_{l,i}$ with $l = 1, \ldots, L_{\max}$ have different distributions.

The encoded string generated by the encoder is the output of the entropy coder in stage 3).

The decoder has the following two stages.

1) Entropy decoder: Reconstruct $\{\widehat{V}_{i,l}\}_{i=1}^{U_l}$ for each l.
2) Locate deletions up to runs: For each l and each i, find the i-th run of extent l in $\mathbf{Y}$, and delete $\widehat{V}_{i,l}$ bits in that run. The outcome is the reconstruction of $\mathbf{X}$.

Since the total number of entries in $\{V_{i,l}\}$ is the total number of runs in $\mathbf{Y}$, which is no larger than n, the size of memory the algorithm takes is $O(n)$. Since the greedy alignment, the generation and coding of $\{V_{i,l}\}$ take $O(n)$ operations, the algorithm takes $O(n)$ operations.

B. Example

Let the side-information, the hidden deletion pattern, and the source sequence be as follows for example:

$$\begin{aligned}\mathbf{Y} &= (1,0,1,1,0,0,0,1,0,1,1)\\ \mathbf{D} &= (1,0,0,1,0,1,0,0,0,1,0)\\ \mathbf{X} &= (0,1,0,0,1,0,1).\end{aligned}$$

On the encoder side:

Stage 1): The greedy alignment algorithm aligns $\mathbf{X}$ and $\mathbf{Y}$ and generates $\widehat{\mathbf{D}}$ as follows.

$$\begin{aligned}\mathbf{Y} &= (1,0,1,1,0,0,0,1,0,1,1)\\ \mathbf{X}^* &= (-,0,1,-,0,0,-,1,0,1,-)\\ \widehat{\mathbf{D}} &= (1,0,0,1,0,0,1,0,0,0,1).\end{aligned}$$

Stage 2): The maximum extent of the runs in $\mathbf{Y}$ is $L_{\max} = 3$. There are $U_1 = 4$ runs of extent 1, $U_2 = 2$ runs of extent 2, and $U_3 = 1$ run of extent 3. For the four extent-1 runs, '1', '0', '1' and '0', only the first one is deleted according to $\widehat{\mathbf{D}}$, therefore we have

$$(V_{1,1}, V_{1,2}, V_{1,3}, V_{1,4}) = (1,0,0,0).$$

For the two extent-2 runs, '1, 1' and '1, 1', there is a deletion in each of them. Therefore we have

$$(V_{2,1}, V_{2,2}) = (1,1).$$

For the only extent-3 run, '0, 0, 0', there is a deletion in it. Therefore we have

$$(V_{3,1}) = (1).$$

Stage 3): The entropy encoder compresses $((V_{1,1}, V_{1,2}, V_{1,3}, V_{1,4}), (V_{2,1}, V_{2,2}), (V_{3,1})) = ((1,0,0,0),(1,1),(1))$. Note that each entry in $(V_{1,1}, V_{1,2}, V_{1,3}, V_{1,4})$ is more likely to be 0 than $(V_{2,1}, V_{2,2})$ and $(V_{3,1})$. Therefore we should use entropy encoder with different target distributions to encode them, when the sequences are long.

On the decoder side:

Stage 1): The entropy decoder reconstructs $((V_{1,1}, V_{1,2}, V_{1,3}, V_{1,4}), (V_{2,1}, V_{2,2}), (V_{3,1})) = ((1,0,0,0),(1,1),(1))$.

Stage 2): Since $(V_{1,1}, V_{1,2}, V_{1,3}, V_{1,4}) = (1,0,0,0)$, the decoder deletes the first run of extent-1, i.e., the first bit. Since $(V_{2,1}, V_{2,2}) = (1,1)$, the decoder deletes a bit from each of the two runs of extent-2. It does not matter which bit to delete in each run. Since $(V_{3,1}) = (1)$, the decoder deletes a bit in the only extent-3 run. The deletions are represented by $\widetilde{\mathbf{D}}$ and the reconstruction of the source sequence is denoted by $\widetilde{\mathbf{X}}$.

$$\begin{aligned}\mathbf{Y} &= (1,0,1,1,0,0,0,1,0,1,1)\\ \widetilde{\mathbf{D}} &= (1,0,1,0,0,0,1,0,0,0,1)\\ \widetilde{\mathbf{X}} &= (0,1,0,0,1,0,1).\end{aligned}$$

Since $\widetilde{\mathbf{X}} = \mathbf{X}$, the reconstruction is correct.

C. Performance of the algorithm

Let $\mathbf{U} := \{U_l\}_{l=1}^{L_{\max}}$ and $\widehat{\mathbf{V}} := \{\widehat{V}_{l,i}\}_{l=1,i=1}^{L_{\max},U_l}$. In the limit as the lengths of the sequences tends to infinity, the operational rate of this algorithm is $R_{op} = \lim_{n\to\infty} H(\widehat{\mathbf{V}})/n$. The optimal rate is $\lim_{n\to\infty} H(\mathbf{X}|\mathbf{Y})/n$. When the probability of deletion d is small, the following theorem shows that the algorithm is asymptotically optimal.

Theorem 1: The gap between the operational rate of the algorithm described in Section III-A and the optimal rate

satisfies: $\lim_{n\to\infty}[H(\widehat{\mathbf{V}})/n - H(\mathbf{X}|\mathbf{Y})/n] = O(d^{2-\epsilon})$, for any $\epsilon > 0$.

The proof is provided in [10, Appendix 1], which can be intuitively explained as follows. When d is small, the deletions are typically far away from each other. Therefore the intervals between the deletions are so long that can be used to synchronize segments of $\mathbf{X}$ to segments of $\mathbf{Y}$. As a result, the deletions can be located within the correct runs with high probability. The exact positions of the deletions within the runs are impossible to find based on only $\mathbf{X}$ and $\mathbf{Y}$. Since the goal is to reconstruct $\mathbf{X}$, describing the positions within runs is unnecessary. Moreover, the description of the locations of the deletions, $\widehat{\mathbf{V}}$, is almost independent of the decoder side-information $\mathbf{Y}$. Therefore sending $\widehat{\mathbf{V}}$ is approximately optimal in terms of rate. See Section III-D-2 for more discussions about the independence between $\widehat{\mathbf{V}}$ and $\mathbf{Y}$. The deletions cannot be located within the correct runs only if two or more deletions are in the same run or adjacent runs, which occurs with the probability in the order of $O(d^2)$. This event affects the performance of the algorithm when d is not small. Therefore the gap between the operational rate and the optimum is in the order of $O(d^{2-\epsilon})$.

Remark 1: In [1], we have shown that when $p = 1/2$, for any $\epsilon > 0$, $\lim_{n\to\infty} H(\mathbf{X}|\mathbf{Y})/n = h_2(d) - cd + O(d^{2-\epsilon})$, where $c := \sum_{l=1}^{\infty} 2^{-l-1} l \log_2 l \approx 1.29$.[4] It captures the asymptotic expansion of the optimal rate to the precision of $\Theta(d)$ with a remainder term $O(d^{2-\epsilon})$. Due to Theorem 1, $R_{op} = h_2(d) - cd + O(d^{2-\epsilon})$, which also matches the optimal rate to the precision of $\Theta(d)$.

Remark 2: In [1], we have shown that $\lim_{n\to\infty} H(\mathbf{X}|\mathbf{Y})/n$ is also the minimum rate when the side-information is only available available at the decoder but not the encoder. Although the minimum rate is the same, constructing an explicit algorithm to implement the distributed compression at the asymptotically optimal rate remains an open problem.

D. Comparison to other compression algorithms

Let us compare the algorithm described in Section III-A with two simpler but suboptimal algorithms in the simple case $\mathbf{Y} \sim$ iid Bernoulli(1/2) ($p = 1/2$). The comparison reveals more intuition on why the algorithm is asymptotically optimal.

1) Sending $\widehat{\mathbf{D}}$ directly: A simple and the most natural algorithm to compress $\mathbf{X}$ given $\mathbf{Y}$ is first running a greedy alignment to obtain $\widehat{\mathbf{D}}$ (as in stage 1)) and then compressing $\widehat{\mathbf{D}}$ using an entropy coder (similar to stage 3)). As the lengths of the sequences tend to infinity, the operational rate is $\lim_{n\to\infty} H(\widehat{\mathbf{D}})/n$. If we approximate $H(\widehat{\mathbf{D}})$ by $H(\mathbf{D})$[5], the operational rate is approximately $h_2(d) = -d\log_2 d + d\log_2 e + O(d^2)$. Therefore for small d, the operational rate of this simple algorithm matches the optimal expression up to the $-d\log_2 d$ term. But for the $\Theta(d)$ term, there is a gap $cd \approx 1.29d$. That is, this compression algorithm wastes 1.29 bits per deletion bit on average. When d is not very small, $-d\log_2 d$ and d can be in the same order of magnitude. Therefore the gap may not be negligible in practice.

The above strategy is suboptimal because $\widehat{\mathbf{D}}$ specifies the exact positions of the deletions. Note that after specifying the runs that contain the deletions and specifying the number of deletions in each run, $\mathbf{X}$ can already be deduced from $\mathbf{Y}$. However, this strategy goes further and specifies the exact positions within the runs, which are redundant in terms of reconstructing $\mathbf{X}$. Therefore this strategy over-describes the positions of the deletions beyond what is necessary to represent $\mathbf{X}$. The amount of over-description, $H(\mathbf{D}|\mathbf{X},\mathbf{Y})$, is called "nature's secret" in [1], because only the hypothetical party "nature" has access to $\mathbf{D}$, but the encoder and decoder do not.

2) Locating deletions up to runs: The analysis of the previous strategy suggests that the encoder should specify the location of the deletions with respect to runs. Therefore a better algorithm than the one described in Section III-D-1 is first defining a sequence $\widehat{\mathbf{W}}$ such that $\widehat{W}_i$ is the number of deletions in the i-th run of $\mathbf{Y}$ according to $\widehat{\mathbf{D}}$, then compressing $\widehat{\mathbf{W}}$ at the entropy rate.

Since the average extent of a run in an iid Bernoulli(1/2) sequence is 2, the length of $\widehat{\mathbf{W}}$ is approximately half of that of $\widehat{\mathbf{D}}$. It can be shown[6] that the operational rate can be approximated by $(h_2(d) - d)$. There is still a linear d gap between this rate and the optimal one, given by $(c-1)d \approx 0.29d$. That is, this algorithm wastes 0.29 bit per deletion bit.

Why is this algorithm suboptimal? The reason is because $\widehat{\mathbf{W}}$ is significantly correlated with $\mathbf{Y}$. If the deletion process is iid, then the longer runs of $\mathbf{Y}$ tend to contain more deletions and the shorter runs tend to contain less deletions. Therefore $\mathbf{Y}$ reveals a certain amount of information about $\widehat{\mathbf{W}}$, that is about 0.29 bit per deletion bit. The algorithm described above does not use this amount of information and thus is suboptimal.

The algorithm described in Section III-A, however, treats the deletions contained in runs of different extents differently. As a result the operational rate matches the optimal rate for the $\Theta(d)$ term.

Table I provides a comparison among the performance of the two algorithms in Section III-D and the one in Section III-A for $n = 1000$kb and $d = 0.01$. Note that when $\mathbf{Y}$ has biased bits ($p = 0.1$), the benefit of the proposed algorithm in Section III-A is more significant than when $p = 0.5$. The reason is that when $p = 0.1$, the runs of $\mathbf{Y}$ are longer and it pays to exploit the information from the run-lengths.

TABLE I
Performance of compression algorithms for $n = 1000kb$, $d = 0.01$.

p	No SI	Sec. III-D-1	Sec. III-D-2	Sec. III-A
0.5	990kb	81kb	71kb	68kb
0.1	469kb	81kb	63kb	46kb

IV. Algorithm for the General Case

The algorithm described in Section III-A can be extended to the general problem where $\mathbf{Y}$ is related to $\mathbf{X}$ by insertions,

[4] In [1], $\mathbf{Y}$ is defined as the deleted version of $\mathbf{X}$. Therefore the expression $H(\mathbf{X}|\mathbf{Y})$ in this paper corresponds to $H(\mathbf{Y}|\mathbf{X})$ in [1].

[5] It can be made rigorous using the techniques that are similar to those used to prove argument (iii) in [10, Appendix 1].

[6] Using the techniques that are similar to those used to prove argument (iii) in [10, Appendix 1].

deletions and substitutions.

A. Algorithm for insertions, deletions and substitutions

The encoder has the following stages.

1) Alignment: align $\mathbf{X}$ and $\mathbf{Y}$ using the minimum total number of insertions, deletions and substitutions. If there are multiple such alignments, pick any one of them. This can be done by the Needleman-Wunsch algorithm [13] with the gap penalty and the substitution penalty equal to 1, with computation complexity of order $O(n^2)$. The algorithm generates two sequences $\mathbf{X}^*$ and $\mathbf{Y}^*$, which are $\mathbf{X}$ and $\mathbf{Y}$ with gaps, respectively. Then construct $\widehat{\mathbf{Z}}_X$ and $\widehat{\mathbf{Z}}_Y$ by replacing the gaps in $\mathbf{X}^*$ and $\mathbf{Y}^*$ by the corresponding bits in $\mathbf{Y}^*$ and $\mathbf{X}^*$, respectively.

2) Describing the insertions (from $\mathbf{Y}$ to $\widehat{\mathbf{Z}}_Y$): The edits from $\mathbf{Y}$ to $\widehat{\mathbf{Z}}_Y$ can be viewed as insertions. The locations of the insertions are specified by the gaps in $\mathbf{Y}^*$. The content of the insertions is specified by the corresponding bits in $\widehat{\mathbf{Z}}_Y$.

All the insertions can be categorized into isolated insertions with only one bit per insertion event, and bursts of insertions with two or more consecutive bits per insertion event. For each insolated insertion, if the inserted bit is equal to the bit on the left (or right) side, the insertion is extending the run to the left (or right). If the inserted bit is not equal to the bits on either side, it is breaking an existing run and creating a new run. We will describe the isolated insertions that extend runs, then the insertions that break runs, then the bursts of insertions.

- In order to describe the insertions that extend runs, the encoder does the following.
  - For $l = 1, \ldots, L_{\max}$ ($L_{\max}$ is the the maximum extent of the runs in $\mathbf{Y}$), for $i = 1, \ldots, U_l$ (U_l is the number of runs of extent l in $\mathbf{Y}$), let $\widehat{V}^{ins}_{l,i} := 1$ if the i-th run of extent l in $\mathbf{Y}$ is extended by one bit, and $\widehat{V}^{ins}_{l,i} := 0$ otherwise.

  Having made such insertions, $\mathbf{Y}$ becomes $\mathbf{Y}'$.
- In order to describe the insertions that break runs, the encoder does the following.
  In the sequence $\mathbf{Y}'$, a slot between two bits is a potential location to break a run only if the two bits are the same. The slots before the first bit and after the last bit are also potential locations to create new runs. Let U_0 denote the total number of such potential locations in $\mathbf{Y}'$. For $i = 1, \ldots, U_0$, let $\widehat{V}^{ins}_{0,i} := 1$ if a bit is inserted in the i-th potential location, and $\widehat{V}^{ins}_{0,i} := 0$ otherwise.
  Having made such insertions, $\mathbf{Y}'$ becomes $\mathbf{Y}''$. Let $\widehat{\mathbf{V}}^{ins}$ denote all the descriptions up to this step: $\{\widehat{V}^{ins}_{l,i}\}_{l\geq 0}$.
- In order to describe the bursts of insertions, the encoder creates a sequence $\widehat{\mathbf{V}}^{burst}$ from $\widehat{\mathbf{Z}}_Y$ by keeping the bursts of inserted bits and replacing the other bits by '$*$'. $\widehat{\mathbf{V}}^{burst}$ describes the insertions needed to construct $\widehat{\mathbf{Z}}_Y$ from $\mathbf{Y}''$.

3) Describe the substitutions (from $\widehat{\mathbf{Z}}_Y$ to $\widehat{\mathbf{Z}}_X$):

The edits from $\widehat{\mathbf{Z}}_Y$ to $\widehat{\mathbf{Z}}_X$ can be viewed as substitutions, which can be described by $\widehat{\mathbf{V}}^{sub} := \widehat{\mathbf{Z}}_Y \oplus \widehat{\mathbf{Z}}_X$.

4) Describe the deletions (from $\widehat{\mathbf{Z}}_X$ to $\mathbf{X}$) as in stage 2) of Section III-A. Denote the description by $\widehat{\mathbf{V}}^{del}$.

5) Entropy coding: Use an entropy coder to compress $\widehat{\mathbf{V}}^{ins}$, $\widehat{\mathbf{V}}^{burst}$, $\widehat{\mathbf{V}}^{sub}$ and $\widehat{\mathbf{V}}^{del}$.

The decoder decodes $\widehat{\mathbf{V}}^{ins}$, $\widehat{\mathbf{V}}^{burst}$, $\widehat{\mathbf{V}}^{sub}$ and $\widehat{\mathbf{V}}^{del}$ by an entropy decoder, and then follow the stages 2) to 4) to construct $\mathbf{X}$ from $\mathbf{Y}$.

B. Performance analysis

The operational rate of the above algorithm can be analyzed for small probability of insertion, deletion and substitution as follows.

Theorem 2: The gap between the operational rate of the algorithm described in Section IV-A and the optimal rate satisfies: $\lim_{n\to\infty}[H(\widehat{\mathbf{V}}^{ins}, \widehat{\mathbf{V}}^{burst}, \widehat{\mathbf{V}}^{sub}, \widehat{\mathbf{V}}^{del})/n - H(\mathbf{X}|\mathbf{Y})/n] = O(d^{2-\epsilon})$, for any $\epsilon > 0$, where $d = \max\{d_X, d_Y, q\}$.

The proof is similar to that of Theorem 1 and is provided in [10, Appendix 2].

V. Concluding Remarks

We have proposed an algorithm to compress the source sequence given the mis-aligned side-information at both the encoder and decoder. For small editing probability, the compression rate of the algorithm is asymptotically optimal. Directions for future work include (1) developing algorithms for bursty edits, and (2) developing distributed algorithms to compress a source sequence when the reference sequence is only available at the decoder side.

References

[1] N. Ma and K. Ramchandran and D. Tse, "Efficient file synchronization: A distributed source coding approach," in *Proc. IEEE Int. Symp. Information Theory*, St. Petersburg, Russia, Jul. 31–Aug. 5 2011, pp. 583–587.

[2] V. L. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965.

[3] A. Orlitsky and K. Viswanathan, "One-way communication and error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1781–1788, 2003.

[4] S. Agarwal, V. Chauhan, and A. Trachtenberg, "Bandwidth efficient string reconciliation using puzzles," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 11, pp. 1217 –1225, nov. 2006.

[5] R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions," in *Proc. of 48th Annual Allerton Conference onCommunication, Control, and Computing*, 2010, pp. 1412 –1419.

[6] A. Tridgell and P. Mackerras, "The rsync algorithm," aNU Technical report, TR-CS-96-05, Jun 1996.

[7] H. Zhang, C. Yeo, and K. Ramchandran, "VSYNC: a novel video file synchronization protocol," *ACM Multimedia*, pp. 757–760, 2008.

[8] H. Cai, S. Kulkarni, and S. Verdu, "An algorithm for universal lossless compression with side information," *Information Theory, IEEE Transactions on*, vol. 52, no. 9, pp. 4008 –4016, sept. 2006.

[9] M. C. Brandon, D. C. Wallace, and P. Baldi, "Data structures and compression algorithms for genomic sequence data," *Bioinformatics*, vol. 25, no. 14, pp. 1731–1738, 2009.

[10] N. Ma, K. Ramchandran, and D. Tse, "A compression algorithm using mis-aligned side-information," http://arxiv.org/abs/1202.0865.

[11] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys*, vol. 6, pp. 1–33, 2009.

[12] M. F. Schilling, "The longest run of heads," *The College Mathematics Journal*, vol. 21, no. 3, pp. 196–207, 1990.

[13] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.