IPN Progress Report 42-173 • May 15, 2008

Lossless Compression of Seismic Data into Fixed-Length Packets

Aaron B. Kiely*

We present an algorithm to losslessly compress a sequence of integers. The intended application is to encode seismic data at nodes in a network of seismometers. The algorithm performs predictive compression, using adaptive linear filtering to predict sample values and encoding variable numbers of samples into fixed-length packets. To accommodate packet losses, the packets include sufficient overhead data to ensure that samples in each packet can be decoded without requiring data from preceding packets. Compression results are presented for seismic test data sets.

I. Introduction

We would like to efficiently losslessly compress the output of a one-dimensional data source that produces integer-valued samples $x_1, x_2, \ldots$. The data has a dynamic range of b bits, and without loss of generality, we may assume that each sample value is in the range $[-2^{b-1}, 2^{b-1} - 1]$.

Our intended application is compression of seismometer data in a 16-node sensor network as part of the Optimized Autonomous Space In-situ Sensor-web (OASIS) project.[1] The network is designed to collect real-time volcano status data from Mount St. Helens. Each node in the network includes a single-component 100 Hz seismometer with dynamic range of $b = 16$ bits and is controlled by a wireless iMote2 mote with a microprocessor operating in low frequency mode (13 MHz) to conserve power. Because of the modest computational power, our compression approach is designed to have relatively low complexity.

Encoded sample values are to be transmitted using fixed-length packets, and so we would like to encode as many samples as possible in each packet. A significant additional problem is that packets are often lost on the channel. For this reason we impose the additional constraint that decoding of a received packet must not depend on the contents of other

*Communications Architectures and Research Section

[1] http://sensorweb.vancouver.wsu.edu/research/oasis.html

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

packets. We assume that time stamp information is already included with each packet so the decoder can properly synchronize received sample values once they are decoded.

Our compression approach relies on adaptive linear prediction of sample values and entropy coding of prediction residuals.

II. Prediction

A. Adaptive Linear Prediction

We maintain a running estimate of the mean input signal value $\hat{\mu}_i$. This estimate is used to compute a "de-biased" version of the source samples

$$d_i = x_i - \hat{\mu}_i.$$

We apply Mth order adaptive linear prediction to the de-biased signal d_i. I.e., the predicted value $\hat{d}_i$ is a linear combination of the preceding M de-biased values,

$$\hat{d}_i = \sum_{j=1}^{M} w_j \cdot d_{i-j} = \mathbf{w}_i^{\mathrm{T}} \mathbf{u}_i. \tag{1}$$

Here $\mathbf{w}_i = [w_1, w_2, \ldots, w_M]^{\mathrm{T}}$ is a vector of weight coefficients that are adapted to the source and $\mathbf{u}_i = [d_{i-1}, d_{i-2}, \ldots, d_{i-M}]^{\mathrm{T}}$ is the vector of the preceding M de-biased sample values. The predicted sample value is

$$\hat{x}_i = \hat{\mu}_i + \hat{d}_i.$$

The estimation error, or prediction residual, is

$$e_i = x_i - \hat{x}_i = d_i - \hat{d}_i.$$

The prediction $\hat{x}_i$ is used to losslessly encode x_i using a a variable length coding scheme described in Section IV. The entropy coding procedure takes into consideration the fact that x_i is an integer while $\hat{x}_i$ is usually not.

After encoding x_i, we use the sign algorithm [1] to update the weight vector:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha \cdot \mathbf{u}_i \cdot \mathrm{sign}(e_i)$$

and we update the mean value estimate via

$$\hat{\mu}_{i+1} = \hat{\mu}_i + \beta \cdot (x_i - \hat{\mu}_i).$$

Here α and β are parameters that control the adaptation of the weight vector and mean estimate to the source statistics.

It might seem more natural to perform mean estimation as part of the sign algorithm instead of as a separate step. Under this alternative, one could define extended vectors $\mathbf{w}'_i = [w'_0, w'_1, \ldots, w'_M]^{\mathrm{T}}$ and $\mathbf{u}'_i = [\kappa, x_{i-1}, x_{i-2}, \ldots, x_{i-M}]^{\mathrm{T}}$ where κ is some fixed constant. Then we could predict the value of sample x_i directly as $\hat{x}'_i = \mathbf{w}'^{\mathrm{T}}_i \mathbf{u}'_i$. We do not adopt this alternative approach because prediction of the first sample value in a packet becomes less straightforward and because compression effectiveness becomes somewhat more sensitive to parameter selection.

B. Prediction Using Integer Arithmetic

To eliminate floating-point operations in the basic algorithm of Section II-A, we use rational approximations to real-valued quantities to produce a version of the algorithm that requires only integer arithmetic. Specifically, the real-valued quantities $\hat{d}_i$, $\hat{x}_i$, $\hat{\mu}_i$, e_i, $\mathbf{w}_i$ are replaced with rational values:

$$\begin{aligned} \hat{d}_i &= \hat{D}_i/2^R \\ \hat{x}_i &= \hat{X}_i/2^R \\ \hat{\mu}_i &= \hat{\Omega}_i/2^R \\ e_i &= E_i/2^R \\ \mathbf{w}_i &= \frac{1}{2^R}\mathbf{W}_i \end{aligned}$$

Here R is some fixed integer ($R = 14$ in our experiments), $\hat{D}_i$, $\hat{X}_i$, $\hat{\Omega}_i$, E_i are integer variables, and $\mathbf{W}_i$ is a vector of integers. In this version of the algorithm, we perform round-off operations that result in d_i being integer-valued, and, consequently, $\mathbf{u}_i$ being a vector of integers. The adaptation parameters α and β are chosen to be $\alpha = 2^{-A}$, $\beta = 2^{-B}$ for some integers A and B so that the multiplications needed to perform the updates can be accomplished via bit-shift operations.

Each iteration of the integer version of the prediction algorithm consists of the following steps:

1. Compute
$$\hat{D}_i = \mathbf{W}_i^{\mathrm{T}} \mathbf{u}_i. \tag{2}$$

2. Compute
$$\hat{X}_i = \hat{D}_i + \hat{\Omega}_i.$$

3. Encode the integer sample value x_i using the rational predicted value $\hat{x}_i = \hat{X}_i/2^R$.

4. Compute the (integer) de-biased value d_i
$$d_i = x_i - \left\lfloor (\hat{\Omega}_i + 2^{R-1} - 1)/2^R \right\rfloor.$$

5. Compute the prediction error
$$E_i = d_i \cdot 2^R - \hat{D}_i.$$

6. Update the weight vector
$$\mathbf{W}_{i+1} = \mathbf{W}_i + \operatorname{sign}(E_i) \left\lfloor (2^R \mathbf{u}_i + (2^{A-1} - 1)\mathbf{1})/2^A \right\rfloor$$
where $\mathbf{1}$ denotes a vector of ones, and the floor operation is applied to each component of the vector.

7. Update the mean value estimate:
$$\hat{\Omega}_{i+1} = \hat{\Omega}_i - \left\lfloor (\hat{\Omega}_i - x_i \cdot 2^R + 2^{B-1} - 1)/2^B \right\rfloor.$$

III. Encoding into Independent Packets

To ensure that samples in a packet can be decoded without requiring preceding packets to be available to the decoder, we make the following algorithm modifications that occur at the start of each packet:

1. We place encoded quantized versions of $\hat{\mu}_i$ and $\mathbf{w}_i$ at the beginning of each packet. The values of $\hat{\mu}_i$ and $\mathbf{w}_i$ are set to these quantized versions in both the encoder and decoder. Quantization is uniform, using Q_μ bits of resolution for the value of $\hat{\mu}_i$ and Q_w bits for each component of $\mathbf{w}_i$.
2. We alter the prediction approach for the first M samples in the packet so that it does not rely on sample values in the preceding packet.

We describe these modifications in further detail below.

A. Quantization

The range of possible values of $\hat{\mu}_i$ is, at least in principle, equal to the range of possible sample values x_i. This range is uniformly partitioned into 2^{Q_μ} bins, and the index of the quantizer bin containing the value of $\hat{\mu}_i$ is encoded in the packet using Q_μ bits. The quantizer index could be encoded a little more efficiently using a variable length code since smaller magnitude values of $\hat{\mu}_i$ are presumably more likely than larger magnitudes, but we did not investigate such a scheme.

Each component of $\mathbf{w}_i$ is clipped as needed to ensure that its magnitude does not exceed some cap 2^C (we use $C = 2$ in our experiments). Each component is then quantized using a uniform quantizer with 2^{Q_w} bins spanning the range $[-2^C, 2^C]$. We make use of a simple variable length coding scheme to exploit the fact that the components of $\mathbf{w}_i$ are usually nonincreasing in magnitude and alternating in sign, i.e.,

1. $|w_1| \geq |w_2| \geq |w_3| \geq \dots$
2. $\text{sign}(w_j) = (-1)^{j+1}$

We say that a quantized weight vector is *ordinary* if it satisfies these conditions.

We use a single bit to indicate whether the quantized weight vector is ordinary. If it is not ordinary, the weight components are sent uncoded using an additional $M \cdot Q_w$ bits. If the weight vector is ordinary, we encode $|w_1|$ directly using $Q_w - 1$ bits, and for $j > 1$, we encode the value of $|w_j|$ using $\lceil \log_2 |w_{j-1}| \rceil$ bits.

B. Predicting the Initial Samples in a Packet

Since the procedure for decoding a packet cannot depend on the contents of another packet, for the first M samples in a packet we must modify the prediction approach as we do not have enough data to perform the calculations of equations (1) or (2) directly. Instead, we do the following:

1. The first de-biased sample value in a packet is predicted to be zero. I.e., for the first sample, prediction makes use of the quantized bias estimate but not the weight vector.
2. For subsequent samples in the packet, when the calculation in equations (1) or (2) would require the use of samples from the preceding packet, the first de-biased sample value is repeated enough times to artificially produce M de-biased sample values to fill the vector $\mathbf{u}_i$.
3. Updates of the weight vector $\mathbf{w}_i$ are not performed for the first M samples in a packet (the samples for which the modified prediction strategy is in effect).

C. Packet Overhead and Parameter Tradeoffs

Compression-related packet overhead consists of the following:

1. The quantized value of $\hat{\mu}_i$ (encoded using Q_μ bits)
2. The quantized value of $\mathbf{w}_i$ (encoded using at most $M \cdot Q_w + 1$ bits)
3. The value of an index indicating which variable length code was used to encode prediction residuals (encoded using $\lceil \log_2 b \rceil$ bits); see Section IV.

We summarize the tradeoffs involved in selecting compression parameters:

- Higher order prediction (a larger value of M) allows for more accurate prediction, but increases the number of components of $\mathbf{w}_i$, thus generally increasing the amount of overhead used to encode $\mathbf{w}_i$ at the start of each packet.

- Higher resolution quantization of $\hat{\mu}_i$ and $\mathbf{w}_i$ (i.e., larger values of Q_μ and Q_w) increases prediction accuracy but increases packet overhead.

- Larger adaptation step sizes (larger values of α, β) allow for faster initial adaptation and adaptation to a dynamically changing source, but provide worse steady-state performance.

IV. Entropy Coding

Source sample values are collected in a buffer. After each source sample arrives, we determine whether the encoded bit cost of the samples in the buffer exceeds the available space in the packet. If not, then we proceed to the next sample. Otherwise, we encode the samples in the buffer (excluding the newest one) using the entropy coding procedure described in the remainder of this section and reset the buffer to contain only the newest sample.

Thus, with the arrival of each new sample, we must determine whether the accumulated samples fit within a packet. Since the coding option used for a packet can change as new samples arrive, explicitly computing the encoded length of the samples in the buffer is not always as simple as incrementing an encoded bit count with the cost of the new sample. The obvious brute-force approach is to simply apply the entropy coding procedure to the samples in the buffer.

Fortunately, we can often avoid the brute-force calculation by bounding the encoded bit cost to quickly identify cases where the packet can accommodate the accumulated samples. For example, the entropy coding procedure guarantees that the average bit cost to encode n samples is no more than $n \cdot b$ bits. As another example, if we have computed the bit cost to encode the first $n-1$ samples, we can bound the cost to encode n samples based on a bound on the incremental cost to encode a single sample. We omit further details of our approach to bounding the encoded bit cost.

Our entropy coding problem then is to efficiently encode a length-n sequence of integer-valued samples x_i given real-valued predictions $\hat{x}_i$. To do this, we map each sample value to a non-negative integer and then encode the resulting sequence of non-negative integers using a Golomb code. This general strategy is used in the Rice entropy coding algorithm [2, 3, 4] and the LOCO-I image compressor [5], among myriad other applications.

A. Mapping

It is sensible to refine the predicted value $\hat{x}_i$ to take into account the fact that the true sample value x_i is an integer and is constrained by the instrument dynamic range. Accordingly, we define

$$[\hat{x}_i] = \min\{\max\{\mathrm{round}(\hat{x}_i), x_{\min}\}, x_{\max}\},$$

where $x_{\min} = -2^{b-1}$ and $x_{\max} = 2^{b-1} - 1$ are the minimum and maximum possible sample values. We use this refined prediction to calculate the integer-valued prediction residual

$$\tilde{e}_i = x_i - [\hat{x}_i].$$

We map the signed integer quantity $\tilde{e}_i$ to a nonnegative integer f_i using a slight variation on the mapping used in [3, 4]:

$$f_i = \begin{cases} 2|\tilde{e}_i| - \delta_i, & \text{if } |\tilde{e}_i| \leq \theta \\ |\tilde{e}_i| + \theta, & \text{otherwise.} \end{cases}$$

Here we define

$$\theta = \min\{[\hat{x}_i] - x_{\min}, x_{\max} - [\hat{x}_i]\}$$

and

$$\delta_i = \begin{cases} 1, & \text{if } \operatorname{sign}(\tilde{e}_i) = \operatorname{sign}(\hat{x}_i - [\hat{x}_i]) \\ 0, & \text{otherwise.} \end{cases}$$

This mapping is invertible and ensures that $f_i \in [0, 2^b - 1]$, i.e., f_i is a nonnegative integer with dynamic range that matches that of the original source. More significantly, the mapping assigns smaller magnitude residuals $\tilde{e}_i$ to smaller values of f_i. Since smaller magnitude prediction residuals should occur more frequently than larger magnitudes, we would like to encode f_i using a variable length code that assigns shorter codewords to smaller integers, such as the codes that we discuss next.

B. Variable Length Coding

For positive integer m, the mth Golomb code [6] defines a reversible prefix-free mapping of nonnegative integers to variable length binary codewords. We restrict our choices to codes for which $m = 2^k$ for some nonnegative integer k. As noted in [6], coding in this case becomes especially simple. The codeword for the integer j consists of the unary representation of $\lfloor j/2^k \rfloor$ (that is, $\lfloor j/2^k \rfloor$ zeros followed by a one) concatenated with the k least significant bits of the binary representation of j. Following the convention of [5], we refer to this special case as a Golomb-power-of-2 (GPO2) code with parameter k.

The samples in a packet are either all sent uncoded, using b bits for each sample, or they are all encoded using the same GPO2 code with some fixed parameter k. The coding option selected is explicitly encoded as part of the packet overhead, as described in Section III-C. For a source with b-bit dynamic range, the cost of using code parameter $k \geq b - 1$ is always at least as large as the cost of sending the samples uncoded [7]. Thus, in our application, when we use a GPO2 code, it must have parameter k satisfying

$$0 \leq k \leq b - 2.$$

This gives us $b - 1$ GPO2 code choices, along with the uncoded option, so our selected code can be indicated using $\lceil \log_2 b \rceil$ bits of overhead.

The coded samples do not usually perfectly fill a packet; following the last encoded sample in the packet, any remaining unused bits in the packet (fill bits) are set to zero. Because each GPO2 codeword begins with a unary-encoded value, these fill bits will never be mistaken for a coded sample value, and the decoder can correctly determine the number of samples encoded in the packet.

We now turn our attention to the problem of selecting a coding option that efficiently encodes some number n of non-negative integers $f_1, f_2, \ldots, f_n$. The traditional solution to this problem is the Rice algorithm's brute-force approach: explicitly compute the coding cost of each option and select the best one [2, 3, 4]. But it was shown in [7] that the brute-force approach is unnecessarily complex; if we simply compute the sum

$$F = \sum_{i=1}^{n} f_i$$

then the mean value F/n allows us to narrow the possible optimum code choices to at most three candidates. Furthermore, by simply comparing this mean value to a list of pre-defined thresholds, we can perform code selection in a way that gives compression effectiveness that is quite close to that obtained under optimum code selection.[2]

Our code selection procedure uses the following steps (further details and mathematical background can be found in [7]):

1. If the mean value F/n is sufficiently large, the f_i are sent uncoded. Specifically, we first check if
$$\frac{F}{n} > \mu_b^\dagger \triangleq \frac{1}{2^{2^{2-b}} - 1}$$
($\mu_{16}^\dagger \approx 23637$). If this condition is satisfied, then the uncoded option is selected. Otherwise, proceed to the next step.

2. Compute K as follows. If $\frac{F}{n} + \frac{49}{128} < 1$ then $K = 0$, otherwise K is the unique nonnegative integer satisfying
$$2^K < \frac{F}{n} + \frac{49}{128} \leq 2^{K+1}.$$
This can be implemented in C source code as:
```
for (K=0; (n<<(K+1)) <= F+(n*49>>7); K++)
    ;
```

3. Assign $k = \min\{K, b-2\}$.

4. Compute the bit cost of using the GPO2 code with parameter k to encode

[2]The Rice algorithm also differs from our coding problem in that the Rice coder encodes a fixed number of samples into a variable number of encoded bits, whereas in our problem we encode a variable number of samples into fixed-length packets.

$f_1, f_2, \ldots, f_n$. If this cost exceeds the uncoded cost (which is $n \cdot b$ bits) then we use the uncoded option, otherwise we select the GPO2 code with parameter k.

Analysis in [7] shows that the GPO2 code parameter k selected under this strategy is always within 1 of the optimum parameter value. Our experiments with seismic data samples suggest that the increased bit rate due to occasional suboptimum code selection is negligible.

V. Results

We now present some compression results for the proposed algorithm. For comparison, we also evaluate the performance of two alternative compression approaches:

1. The *simple* algorithm. The first sample in each packet is included directly in the output, and the remaining samples in each packet are compressed by applying the block-adaptive GPO2 coding approach of Section IV using the previous sample value as the predictor.
2. The *continuous* algorithm. We perform the adaptive linear prediction of Section II-A and the sample mapping of Section IV-A to compute the mapped sample values f_i which are encoded using GPO2 codes. The code parameter k is chosen separately for each f_i using the selection approach of Section IV-B applied with the mean mapped value F/n replaced by a decaying running average of past mapped samples. This encoding approach yields a more compact representation of the data because no overhead information (such as the value of k or information about prediction parameters) is encoded and no quantization of prediction parameters is performed. However, the decoder cannot recover from the loss of even a single packet. This approach is intended to illustrate the compression performance that could be obtained if we did not need to ensure that packets can be decoded independently.

We apply the compressors to five test sets of 12-bit seismic data, each consisting of 30 minutes of 100 Hz data (180100 samples each). The test sets include an earthquake swarm ("eswarm"), a storm, and low level activity ("background"). For our proposed compressor, we use algorithm parameters $R = 14$, $B = 8$, $A = 15$ for all results shown here. We also use 56-byte packets[3] and quantizer resolution $Q_w = 5$, $Q_\mu = 11$, as our defaults except where noted otherwise.

Table 1 shows the compressed bit rates achieved on the test data using the simple algorithm, the continuous algorithm, and our proposed approach with default parameters. With a good choice of compression parameters, the proposed approach provides a performance benefit over the simple compressor. We observe also that for a fixed

[3]Here and in the sequel, the nominal packet size refers to the space available in each packet for compressed data including the overhead described in Section III-C, but not counting the cost of a timestamp or other auxiliary data.

compression approach, the compressed bit rate that can be achieved on the test data sets varies by about a factor of two. Clearly one might expect significant fluctuations in compressed bit rate when using this compression algorithm in a network of seismometers.

Table 1. Compressed bit rates (bits/sample) achieved on five seismic data sets encoded using 56-byte packets using the simple algorithm, the proposed algorithm with default quantizer resolution, and the continuous algorithm.

		Proposed Algorithm				Continuous Algorithm			
	Simple	Prediction Order M				Prediction Order M			
Data Set	Algorithm	2	3	4	5	2	3	4	5
(a) SEP.EHZ.background	5.21	5.21	5.27	5.22	**5.10**	4.56	4.51	4.49	4.49
(b) SEP.SHZ.eswarm	7.10	6.91	6.74	**6.59**	6.67	6.56	6.30	5.97	5.93
(c) YEL.EHZ.background	7.77	7.25	7.24	**7.17**	7.37	6.72	6.61	6.46	6.46
(d) YEL.SHZ.eswarm	8.76	8.53	8.42	**8.16**	8.41	7.80	7.47	7.11	7.04
(e) YEL.EHZ.storm	10.76	10.47	10.36	**10.34**	10.47	9.75	9.53	9.22	9.22

To provide an indication of the improvement that could be obtained by making a better choice of quantizer resolution, Table 2 gives the minimum bit rate achieved by the proposed approach if we selected the best quantizer resolution settings (Q_w, Q_μ) for each choice of data set and prediction order. (Note that under the current algorithm it is not possible to optimize quantizer resolution on individual packets.) The table also shows the portion of the bit rate that is due to overhead bits and the choice of quantizer resolution that achieves this minimum rate. Optimizing quantizer resolution tends to yield an improvement of about 0.1 bits/sample over the default choice on the test data sets.

Table 2. Compression results for the proposed algorithm when quantizer resolution is adjusted to minimize bit rate achieved for 56-byte packets. The table gives the minimum compressed bit rate (bits/sample), the rate cost due to overhead (in bits/sample), and the choice of quantizer resolution (Q_w, Q_μ) that minimizes bit rate.

	Bit Rate				Overhead Cost				Quantizer Resolution			
Data	Prediction Order M				Prediction Order M				Prediction Order M			
Set	2	3	4	5	2	3	4	5	2	3	4	5
(a)	**4.95**	5.00	5.09	5.08	0.37	0.45	0.56	0.33	(10,12)	(10,11)	(10,11)	(5,9)
(b)	6.90	6.69	**6.48**	6.50	0.32	0.36	0.35	0.35	(5,9)	(5,9)	(4,10)	(4,9)
(c)	7.20	7.13	7.17	**7.10**	0.37	0.37	0.53	0.36	(7,7)	(4,11)	(5,11)	(4,8)
(d)	8.26	8.04	**7.96**	8.02	0.33	0.42	0.44	0.46	(7,3)	(7,3)	(5,3)	(5,3)
(e)	10.26	10.15	**10.13**	10.18	0.28	0.39	0.43	0.46	(4,3)	(5,3)	(4,3)	(4,3)

Packet size has a significant impact on compression effectiveness. To illustrate this, Table 3 shows the compressed bit rate achieved and the rate cost due to overhead bits when we double the packet size from the default of 56 bytes to 112 bytes while using the choice of quantizer resolution shown in Table 2.

Comparing the performance of the proposed and continuous algorithms in Table 1 shows that producing packets that can be decoded independently incurs a significant penalty in compressed bit rate. Comparing Tables 2 and 3 shows that this penalty is particularly

Table 3. Compressed bit rate and rate cost due to overhead (in bits/sample) for the proposed algorithm using 112-byte packets and the values of (Q_w, Q_μ) shown in Table 2

	Bit Rate				Overhead Cost			
	Prediction Order M				Prediction Order M			
Data Set	2	3	4	5	2	3	4	5
(a)	4.74	4.72	4.75	4.89	0.18	0.21	0.26	0.16
(b)	6.71	6.48	6.27	6.25	0.16	0.17	0.17	0.26
(c)	7.28	6.90	6.90	6.93	0.19	0.18	0.27	0.23
(d)	8.03	7.74	7.55	7.63	0.16	0.20	0.22	0.22
(e)	10.00	9.84	9.71	9.76	0.13	0.19	0.22	0.25

significant when packets are smaller. Sources of this penalty include:

- The cost of overhead bits. Tables 2 and 3 demonstrate that this cost is significant and that, as expected, doubling packet size while keeping other compression parameters fixed reduces the overhead rate cost by about half.

- Quantization of $\mathbf{w}_i$ and $\hat{\mu}_i$ performed at the start of each packet reduces prediction accuracy, thus reducing compression effectiveness. The weight vector and mean estimate continue to adapt during the course of encoding a packet, and so the penalty for quantization should generally decrease slightly as packet length increases.

- The cost of fill bits. On average we would expect the number of fill bits in a packet to be about half the average codeword length. So if the compressed bit rate is r bits/sample, then we would expect about $\frac{1}{2}r$ fill bits per packet. The average number of samples encoded in a packet consisting of P bits is about P/r, and thus the average rate due to fill bits is approximately

$$\frac{\frac{1}{2}r}{P/r} = \frac{r^2}{2P}.$$

  Empirical results confirm that this is a good approximation, generally within about 0.01 bits/sample of the actual rate cost of fill bits. This cost is relatively small, generally about 0.1 bits/sample or less on the test data sets.

- Prediction is not as effective for the first M samples in a packet (for which we use less than the full power of the Mth-order predictor), and so these samples are not encoded as effectively. This accounts for a larger fraction of samples in smaller packets.

Comparing Tables 2 and 3, we see that doubling packet size while keeping quantizer resolution fixed generally yields a reduction in bit rate that is about equal to the rate savings in overhead and fill bits for the test data sets. One exception is data set (c) with 2nd order prediction, bit rate actually *increases* by 0.08 bits/sample despite a reduced overhead rate of 0.18 bits/sample. It's unclear why this is happening, though we note that a 2nd order predictor is not particularly good for this data set. Other exceptions are data set (b) with 5th order prediction, and data sets (d) and (e) with 4th and 5th order prediction. In each of these cases there is an additional savings of about 0.15 bits/sample. In fact, when given the opportunity to double the packet size, we would likely use higher resolution quantizers and thus see slightly larger improvements than indicated from Tables 2 and 3.

When the packet loss rate is sufficiently small, increasing packet size to improve compression effectiveness appears to be a prudent thing to do, up to a point. Our coding approach is not designed for very long packets; coding is constrained to use the same entropy coding option for all samples in a packet, and so short packets have the potential to offer faster adaptivity to changing source behavior. Allowing the entropy coding option to vary during the encoding of a packet could offer some improvement, but we suspect that this only occurs for packets much larger than the ones of interest in our application.

We conclude by mentioning two areas for potential improvement. First, perhaps the most obvious improvement would be to devise a more effective strategy for encoding overhead information than the one currently employed. Along these lines, would could envision strategies for joint quantization and encoding of overhead information. For example, we might alter the quantization of $\mathbf{w}_i$ depending on the magnitude of the bias estimate or the value of the GPO2 parameter k.

Second, the algorithm would clearly be more practical if it were modified to independently adaptively choose quantizer resolution based on observation of the data, rather than requiring a user to make this selection.

Acknowledgments

The author would like to thank Matthew Klimesh for several insightful discussions, Sharon Kedar for the invitation to work on this problem and for providing the seismic test data, and WenZhan Song at Washington State University for helpful feedback.

References

[1] A. Gersho, “Adaptive Filtering with Binary Reinforcement,” *IEEE Transactions on Information Theory*, vol. IT-30, pp. 191-199, Mar. 1984.

[2] R. F. Rice, “Some Practical Universal Noiseless Coding Techniques,” Tech. Rep. JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, Mar. 1979. http://ntrs.nasa.gov, Document ID 19790014634

[3] R. F. Rice, “Some Practical Universal Noiseless Coding Techniques, Part III,” JPL Publication 83-17, Jet Propulsion Laboratory, Pasadena, CA, Mar. 1983. http://ntrs.nasa.gov, Document ID 19830019798

[4] Consultative Committee for Space Data Systems, *CCSDS 121.0-B-1: Lossless Data Compression*, Blue Book, issue 1, May 1997.
http://public.ccsds.org/publications/archive/121x0b1c2.pdf

[5] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, August 2000.

[6] S. W. Golomb, "Run-Length Encodings," *IEEE Transactions on Information Theory*, vol. IT-12, no. 3, pp. 399–401, July, 1966.

[7] A. Kiely, "Selecting the Golomb Parameter in Rice Coding," *IPN Progress Report*, vol. 42-159, pp. 18, November 15, 2004.
http://ipnpr.jpl.nasa.gov/progress_report/42-159/159E.pdf