Dynamic Online Performance Optimization in Streaming Data Compression

J. Kade Gibson*, Dongeun Lee*, Jaesik Choi† and Alex Sim‡
*Department of Computer Science
Texas A&M University-Commerce, Commerce, Texas 75428
Email: dongeun.lee@tamuc.edu (equal contribution)
†School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology, Ulsan 44919, Korea
Email: jaesik@unist.ac.kr
‡Scientific Data Management Group
Lawrence Berkeley National Laboratory, Berkeley, California 94720
Email: asim@lbl.gov

***Abstract*—Compression is essential to high bandwidth applications such as scientific simulations and sensing applications to reduce resource burden such as storage, network transmission, and more recently I/O. Existing lossy compression methods attempt to minimize the Euclidean distance between original data and reconstructed data, which significantly limits either compression performance or reconstruction quality since original and reconstructed data sequences should be aligned. Substituting the Euclidean distance for a statistical similarity maximizes the compression performance while retaining essential data features. By implementing this methodology, IDEALEM has recently demonstrated compression ratios far exceeding 100:1, better than best-known compression methods, while preserving reconstruction quality. This work proposes an online algorithm for streaming data compression which takes account of generally concave trend of compression ratio curve, and optimizes key operation parameters. We demonstrate that the proposed algorithm successfully adapts one of the key parameters in IDEALEM to the optimal value and yields near maximum compression ratios for time series data.**

***Keywords*-compression performance optimization; multistage random sampling; lossy compression; Kolmogorov-Smirnov test; time series data**

I. Introduction

High bandwidth applications such as scientific simulations and sensing applications generate huge volumes of data. This poses a challenge as to how we can handle and manage data at an unprecedented scale. Data compression reduces the volume of data at the cost of computational resources, which has found its usage in reducing storage and transmission burden. More recently, data compression attracts attention again due to streaming data applications and the increasing disparity between computational speed and I/O rates.

Among many types of streaming data generated by various applications, floating-point data takes a significant portion due to widespread usage of sensing applications such as power grid monitoring. However, floating-point data is known to be especially hard to compress because of its random and noisy nature [1]. Notwithstanding, there have been research efforts for the compression of this seemingly incompressible data type [1]–[3]. In order to accommodate the randomness and noisiness in data, these techniques are in general based on lossy compression which allows removing less important information, including randomness and noisiness. In particular, they allow slack in original data values in terms of the Euclidean distance to simplify the representation of the original data. In fact, the Euclidean distance is a popular measure for data quality to this day, when we consider the trade-off between data size and quality [4], [5].

However, relying on the Euclidean distance measure places significant restrictions on the performance of data compression techniques. To remove the randomness and noisiness inherent in floating-point data more effectively, we recently proposed a new class of lossy compression method based on statistical similarity, dubbed IDEALEM (Implementation of Dynamic Extensible Adaptive Locally Exchangeable Measures) [6]–[8], which was shown to be effective in capturing essential characteristics of data with very high compression performance. In particular, IDEALEM showed its capability of identifying representative examples of data a domain expert would recognize and extract. IDEALEM specializes in the compression of streaming floating-point data, which further enables online data analysis thanks to its simple encoded stream structure. Since IDEALEM is targeted for streaming data, its impact on computational resources such as CPU and memory is minimal, which enables IDEALEM to be deployed on any resource-constrained devices.

This paper proposes an online algorithm which optimizes the parameters of streaming data compression method such as IDEALEM. Leveraging the generally concave trend of compression ratio curve, this algorithm dynamically finds and converges quickly on parameters which can yield near maximum compression ratios. Here, the compression ratio is defined by the ratio of the original data size to the compressed data size.

As a concrete example, IDEALEM has a few key pa-

rameters a user can control to optimize its compression performance. However, one of these parameters, the block size which determines the number of samples in a data block, has remained unclear about its effect on the performance. This is because two different forces act in opposite directions: increasing the block size in theory should yield high compression performance, but it also incurs difficulty in compression with the statistical similarity measure, Kolmogorov-Smirnov test (KS test). As a result, a user had to find an optimal block size through trial and error.

We test the proposed algorithm on IDEALEM and show that it adapts the block size parameter to the optimal value for the maximum compression of time series data. This approach can be applicable to other online parameter optimization problems as well when there is any trade-off between parameters such as in parallel data I/O [9] and concurrent network data transfers [10], where results show certain concave or convex patterns. The algorithm can be further extended to other applications involving online optimizations such as experimental testbeds to determine change point alerts for data collection performance [11] and the autonomous control of the sensor device settings depending on surrounding conditions [12].

II. Statistical Similarity Based Data Reduction

Various application scenarios which generate floating-point values can be explained by random number generations: devices such as sensors might be measuring background noise during their operation time, and network monitoring devices would be observing random traffic.

The main idea of IDEALEM is to find and store the data sequence patterns which are distinct from previous data sequences in terms of statistical similarity. To this end, IDEALEM breaks an incoming data stream into blocks of a fixed size and represents statistically similar blocks with a data block which appears earlier in the data stream. If we assume each data block is an instantiation of a random variable, we can consider an *exchangeability* of these random variables, where the exchangeability can be assumed if these random variables share an identical distribution as their data source.

Fig. 1 represents a graphical model for a streaming data with 64 samples. IDEALEM groups statistically similar sample data blocks together and represents them with a single random variable, which leads to the compression of data [6]. In particular, IDEALEM keeps the first occurrence of similar data blocks to store as a latent random variable. During the encoding stage, IDEALEM learns these common probability distributions behind data blocks. Therefore new data sequences should be generated from the learned distributions in the decoding stage, where it is impossible to reconstruct the same data sequence as the original except the learned distribution itself. However, relaxing the order of data sequence makes IDEALEM very effective in terms

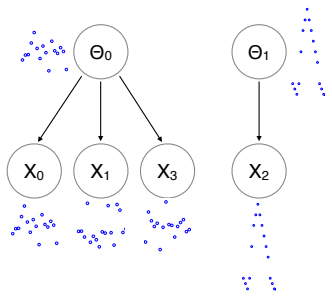


Figure 1. Input streaming data is divided into fixed-size blocks, each of which can be treated as an instantiation of a random variable X_i. IDEALEM compresses the input data by learning common probability distributions behind the group of random variables, which are represented by latent random variables Θ_0 and Θ_1. These distributions are non-parametric, allowing any shapes of distributions. Here, three data blocks and corresponding random variables X_0, X_1, and X_3 can be governed by Θ_0; but the data block corresponding to X_2 is dissimilar and can be governed by Θ_1.

of compression performance, without compromising reconstruction quality much [6]. This is possible because in many applications, if not all, an exact reproduction of a random fluctuation such as background noise is unnecessary.

A. Similarity Measure

IDEALEM adopts the well-known KS test as its statistical similarity measure [13], [14]. KS test, especially two-sample KS test, is a non-parametric statistical hypothesis testing method which can test whether two underlying one-dimensional probability distributions of random variables differ or not.

Since the KS test is non-parametric, it can compare two random variables from any arbitrary distributions without parameters. In particular, the maximum distributional distance D_{n_i,n_j} between two random variables X_i and X_j is defined by

$$D_{n_i,n_j} := \sup_x |F_{X_i,n_i}(x) - F_{X_j,n_j}(x)|, \tag{1}$$

where $F_{X_i,n_i}(\cdot)$ and $F_{X_j,n_j}(\cdot)$ are empirical (cumulative) distribution functions of X_i and X_j; n_i and n_j are the numbers of samples for X_i and X_j respectively; sup is the supremum. The distance (1) is also called the *test statistic*, which is subsequently *standardized* with respect to n_i and n_j as follows:

$$\widetilde{D}_{n_i,n_j} := D_{n_i,n_j}\sqrt{\frac{n_i n_j}{n_i + n_j}}. \tag{2}$$

As this standardized distance (2) grows, the *p-value* [15] gets smaller.

A small p-value indicates the *null hypothesis*, i.e., two random variables are from the same distribution, is more likely to be wrong, which automatically supports its logical complement, i.e., two random variables are *not* from the same distribution. In practice, a threshold α is specified by

the user, which is also called the *significance level*. If a p-value is less than or equal to a chosen α, we reject the null hypothesis, supporting its logical complement. IDEALEM interprets this α as a threshold for similarity, so as to remove redundancy from original data [6].

B. A Key Parameter in Question

IDEALEM has three key parameters which affect its compression performance. The number of buffers b controls how many learned distributions Θ_j are simultaneously stored in memory for comparison, where each buffer holds a single Θ_j. It is apparent more buffers promise higher compression ratios because there is a higher chance of finding a similar distribution stored in buffers when we encounter new X_i.

The similarity threshold α explained in Section II-A is used when comparing new X_i to Θ_j stored in buffers via the KS test. Thus, a lower α results in a higher compression ratio, allowing more X_i's to be declared exchangeable. On the other hand, lowering the bar for similarity impairs the reconstruction quality, as it would also include not-so-similar sequences under the same Θ_j.

Finally, the block size n determines the number of samples in an individual data block. An incoming time series is broken down into blocks with each of them having n elements. However, its effect on compression performance is involved due to the characteristics of the KS test and the design of IDEALEM [6].

We can observe the scaling factor $\sqrt{n_i n_j/(n_i+n_j)}$ in (2) grows with increasing numbers of samples. For instance, this factor is simply $\sqrt{n/2}$ when $n_i = n_j = n$. Therefore, larger n_i and n_j can yield the same $\widetilde{D}_{n_i,n_j}$ with a smaller D_{n_i,n_j}.

Fig. 2 shows the plot of the p-value versus the test statistic (1) with various n's ($n_i = n_j = n$). Given a distance $D_{n,n}$, a larger n leads to a smaller p-value. Thus even a small distance with a large n could lead to a small p-value. In other words, the same p-value may correspond to different test statistics depending on n.

On the other hand, the theoretical upper bound of achievable compression ratio increases linearly with the block size n, which is especially $8n$ when IDEALEM handles data in IEEE 754 double precision floating-point format. (See Proposition 1 in [6].) This is because we can represent each data block in $8n$ bytes with a pointer in 1 byte in the ideal case where all data blocks have the same latent distribution behind.

Considering these two aspects together, the effect of the block size n on the compression performance is not straightforward. Although the compression ratio should increase with n in principle, it becomes also difficult to pass the KS test as n increases, as shown in Fig. 2.

C. Concavity in Compression Ratio

The two different factors acting on the compression ratio with reference to the block size n result in a *concave trend*

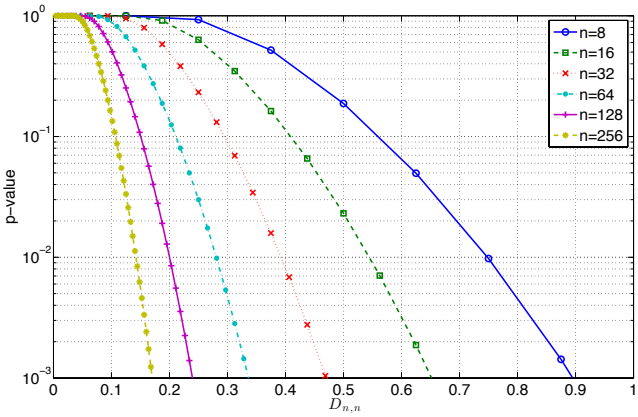


Figure 2. Effects of numbers of samples on the p-value and corresponding test statistics (distances) $D_{n,n}$, where the y-axis is drawn in log scale. Six numerical results are shown, where each has n sample points for two random variables. As n grows, it becomes more difficult to exchange random variables due to lower p-values for a given distance.

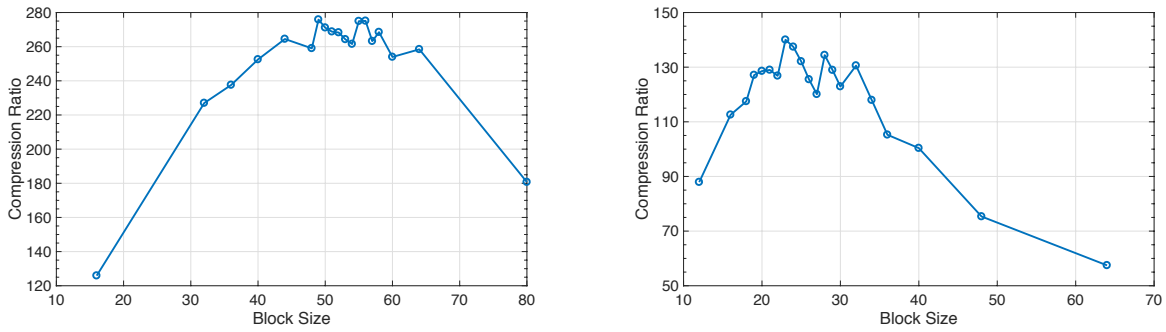
in the compression ratio: as n increases, the compression ratio also increases thanks to its linearly growing relationship with n; but after a certain point, the difficulty of passing the KS test dominates and increasing n has an adverse effect on the compression ratio.

Fig. 3 shows the concavity of compression ratio with varying block sizes n, for four different datasets. Here we use approximately two weeks of power grid monitoring data collected by two different sensors (A6BUS1 and BANK514) installed at Lawrence Berkeley National Laboratory (LBNL). Each sensor monitors three-phase voltages and currents, collecting magnitude (MAG) and phase angle (ANG) measurements. Specifically, this work uses phase 1 MAG data for both current (C) and voltage (L). We observe compression ratios in Fig. 3 vary substantially across different n's; even small errors in n's may lead to drastic deflation in the compression ratios.

In an ideal condition where an entire data stream can be represented by a single distribution, the compression ratio would follow the theoretical upper bound $8n$ and the difficulty of passing the KS test would not occur. On the other hand, for a data stream where every data block is dissimilar from each other, the compression ratio would be less than 1 regardless of n, due to the index overhead in the encoded stream structure as explained in Section II-D. However, in practice, these two extreme conditions do not typically occur and we assume the trend of compression ratio is always concave in this work.

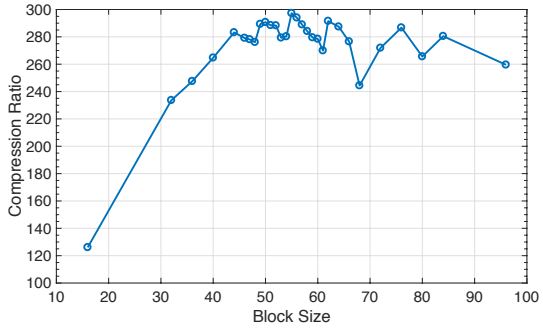
D. Encoded Stream Structure

Fig. 4 illustrates an example of the modified encoded stream structure from IDEALEM [6]. This example shows a single block size change from $n = 8$ to $n = 16$, when there are three buffers ($b = 3$). Every encoded stream starts with

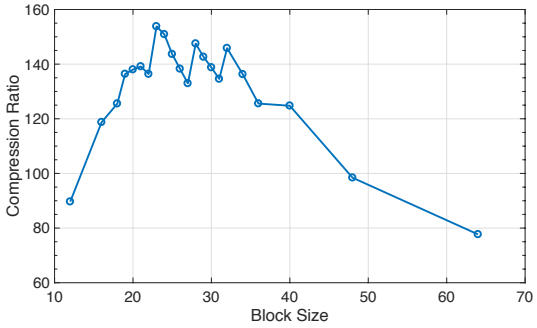


(a) A6BUS1C1MAG

(b) A6BUS1L1MAG



(c) BANK514C1MAG



(d) BANK514L1MAG

Figure 3. Compression ratio variations of IDEALEM with power grid monitoring data collected by A6BUS1 and BANK514 when $\alpha = 0.01$ and $b = 255$. Overall, current magnitude data (C1MAG) show higher compression ratios than voltage magnitude data (L1MAG) do. Despite many peaks and valleys around global maxima, all of their trends are concave within certain block size ranges.

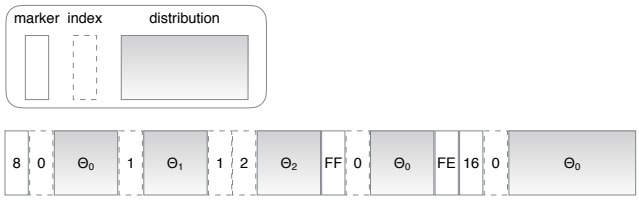


Figure 4. An example of the modified encoded stream structure from original IDEALEM to incorporate dynamic block size changes. A solid box represents a marker in 1 byte; a dotted box an index j in 1 byte; a solid box with gradation a distribution Θ_j in $8n$ bytes.

a starting block size n in 1 byte so that the decoder knows the initial block size.

Next, the first data block in an input stream is outputted *as is*, along with the corresponding index j in 1 byte which precedes the block, where counting starts from 0. Note this data block is also stored in a buffer as the *distribution* Θ_0, where each buffer occupies $8n$ bytes. Then, the second block is encountered and compared against the first block in the buffer. In this example, it is not exchangeable, so the data block itself is written on the encoded stream as well as the corresponding index. It is also stored in a buffer as the distribution Θ_1. The third block is first compared with Θ_0, but not exchangeable. It is next compared with Θ_1, and found to be exchangeable. So the index 1 is solely outputted.

The fourth block is not exchangeable with any of two stored distributions. So it is again written on the encoded stream as is with the corresponding index. This data block also occupies the last remaining buffer as the distribution Θ_2. The fifth block is compared with previous three buffers, but not exchangeable with any of them. Therefore this distribution should be stored in a buffer, which is not immediately possible since all three buffers are occupied. IDEALEM discards an existing buffer in first-in-first-out (FIFO) manner. Thus the fifth data block overwrites the oldest distribution Θ_0. This overwriting should be signaled on the encoded stream so that the decoder can recognize it. To this end, IDEALEM uses a marker `0xFF`. This marker is first outputted, and then the index and the data block itself is written on the encoded stream.

So far, $n = 8$ for data blocks and distributions. But our online algorithm dynamically changes the block size to reach to the maximum compression ratio. To signal the block size change, IDEALEM uses a marker `0xFE`, which is followed by another marker denoting a new $n = 16$. At this point, IDEALEM flushes every buffer; thus the next data block is outputted to the encoded stream with the preceding index 0. This data block also occupies the new buffer as the distribution Θ_0. Note Θ_0 was previously 64 bytes, but it is

now 128 bytes.

This new encoded stream structure is designed in a backward-compatible way that the decoder can easily handle both the new structure and the original structure without the block size change. Due to the usage of the signaling markers `0xFF` and `0xFE`, the number of buffers b can increase up to 254.

III. Online Parameter Optimization

The goal of our algorithm is to find the vertex of concave or convex such as the compression ratio curve shown in Fig. 3, as quickly and accurately as possible, thus minimizing exploration time and maximizing performance. However, this is not straightforward due to the difficulty of learning the curve in real time for streaming data. For IDEALEM, it is impossible to know in advance every compression ratio at every block size n. Note that Fig. 3 shows the ground-truth results of compression ratios for different n's, which require the entire dataset and are unique to each dataset.

A. Challenges in Measuring Performance

Due to the online nature of our algorithm, the performance of a given n can be only evaluated by an approximation of compression ratio, which is denoted by a running compression ratio ρ. To this end, we employ the concept of *Bernoulli process* to compute ρ. When the IDEALEM encoder encounters a new data block in an input data stream as explained in Section II-D, it tries to find an exchangeable block by searching through existing buffers. If this succeeds, we simply output an index j; otherwise we need to output the data block itself to the encoded output stream. We model these *success* and *failure* with a Bernoulli trial.[1]

Specifically, we consider the success of the trial *hit* and keep track of a hit count h and a trial count r during the encoding process. Therefore the hit ratio, i.e., the estimator for the success probability of the Bernoulli trial, is denoted by $\hat{p} = h/r$. In practice, we are only given a limited number of trial samples. Specifically, when $r < 30$, it is common to utilize Student's t-distribution to compute the *confidence interval* of the population mean. However, we empirically found the running compression ratio ρ is extremely sensitive to changes in $\hat{p}$. Thus we only consider the case of $r \geq 30$, where we can represent the confidence interval of the true success probability p assuming the *central limit theorem* as follows:

$$\hat{p} - z^* \frac{s}{\sqrt{r}} \leq p \leq \hat{p} + z^* \frac{s}{\sqrt{r}}, \tag{3}$$

where z^* is the critical value which can be found on the normal distribution table according to the confidence level; s is the sample standard deviation.

[1]When every buffer is empty, i.e., in the very beginning of the encoded stream or right after a block size change, a new data block does not have a chance to test the exchangeability; thus this case is not counted as a trial.

In (3), s is bounded at 0.5 when $\hat{p} = 0.5$; thus $s \leq 0.5$ in any case. In order to account for the uncertainty introduced when r is low, we take the lower bound in (3) and substitute $\min(0.5, s)$ for s to yield a *guaranteed* minimum $p_{\min}$ as follows:

$$p_{\min} \coloneqq \hat{p} - z^* \frac{\min(0.5, s)}{\sqrt{r}}. \tag{4}$$

Using (4), we are interested in a guaranteed minimum running compression ratio $\rho_{\min}$, which is presented by the following proposition.

Proposition 1: The guaranteed minimum running compression ratio $\rho_{\min}$ is $8n/(1 + 8n - 8np_{\min})$.

Proof: The hit ratio $\hat{p}$ indicates how often we can represent a data block with a 1 byte index. With r trials and the assumption of IEEE 754 double precision floating-point data format, the original data size (in bytes) can be represented by $8n + 8nr$, where $8n$ is the size of the first data block; $8nr$ is the size of the entire data stream excluding the first data block. On the other hand, the compressed data size is represented by $1 + (1 + 8n) + h + (1 + 8n)(r - h)$, where 1 is the size of the starting block size; $(1 + 8n)$ is the size of the initial index and Θ_0; h is the size of index representations; $(1+8n)(r-h)$ is the size of index and data block representations.

If continuous streaming of data is assumed, the limiting compression ratio is given by

$$\lim_{r\to\infty} \frac{8n + 8nr}{1+(1+8n)+h+(1+8n)(r-h)} = \frac{8n}{1+8n-8n\hat{p}}. \tag{5}$$

If we plug $p_{\min}$ into $\hat{p}$ in (5), then we have the guaranteed minimum running compression ratio $\rho_{\min} = 8n/(1 + 8n - 8np_{\min})$. ∎

Fig. 5 shows the growth of $\rho_{\min}$ over $r \in [1, 100]$ for $n = 14, 46, 75, 80$. In Fig. 5, $\rho_{\min}$ grows unstably and the rank of each n changes frequently in this range. Since the approximation accuracy of $\rho_{\min}$ increases as r approaches infinity in principle, more trials are required to stabilize $\rho_{\min}$ and the ranks of different block sizes n. This is because when the trial count r is low, much uncertainty exists in the success probability p as shown in (3).

The nature of IDEALEM is to accumulate buffers to compare with incoming data blocks. Each time the block size n changes, every buffer must be reset and IDEALEM starts from scratch, discarding all the previously collected distributions Θ_j, along with h and r. In other words, resetting mature buffers is a very expensive operation and should be carefully done in *correct timing*, because it is irrevocable and later repeating trials for the same n is unreasonable.

Thus we should obtain an enough number of r for a given n to have a relatively stable $\rho_{\min}$. On the other hand, too many trials impede the convergence of our online algorithm, which leads to degradation in overall compression performance. It is therefore important to switch to another

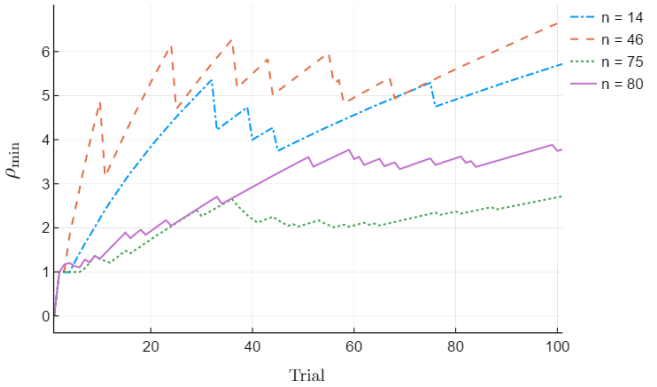


Figure 5. Growth of $\rho_{\min}$ over $r \in [1, 100]$ with $b = 254$, $\alpha = 0.01$, and $n = 14, 46, 75, 80$. Here, $\rho_{\min}$ grows continuously and the rank of n changes frequently.

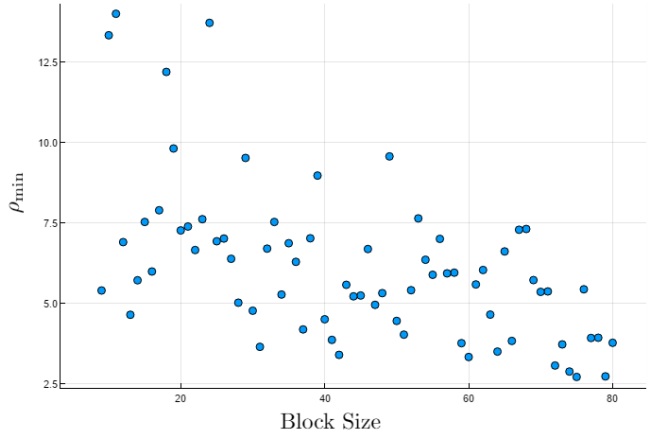


Figure 6. The guaranteed minimum running compression ratio $\rho_{\min}$ approximates the final compression ratio for $n \in [8, 80]$ at $r = 100$, $b = 254$, and $\alpha = 0.01$.

n not too early, but not too late. Once the algorithm finds the optimal n, it stops and no longer changes the block size.

B. Multistage Random Sampling

Fig. 6 illustrates the relationship between n and $\rho_{\min}$, which shows $\rho_{\min}$ approximates the compression ratio with 100 trials. Although 100 trials may not be enough for $\rho_{\min}$ to stabilize, we empirically found the rank of n rarely changed at $r \geq 100$, which helps considerably when modeling the concave trend. In Fig. 6, the small area of concavity within some window around the n with a maximum $\rho_{\min}$ is a consistent feature of the relationship across different datasets. Our algorithm works to localize this known feature, rather than discover and model the relationship of the entire range from scratch.

In particular, we employ *multistage random sampling* in our algorithm to quickly find an area of apparent concavity. Random sampling is a way to effectively capture inherently sparse information with a relatively small number of samples [4], [16], [17]. Compared with uniform sampling, the random sampling requires a smaller number of samples to capture the same amount of information.

It is worth noting that the random sampling employed in our algorithm has been also used in random sample consensus (RANSAC) to estimate parameters of a curve when data is subject to noise [18]. However, RANSAC is not directly applicable to our work due to two reasons: (1) it assumes observed data points are already given prior to its operation, which is unsuitable for online performance optimization; (2) its voting mechanism to filter out outliers in data points requires at least a few iterations over the same data points, which is too expensive for an online algorithm.

Basically, random samples serve to locate the known concavity in the relationship between n and $\rho_{\min}$ with minimal exploration time. Initially, we acquire random samples of n across all possible block sizes. This initial sample serves to locate a range where the concavity lies, but it does not illustrate sufficiently for modeling a concave curve. To this end, we use a small size of initial random samples.

After initial sampling, another is taken using a window of size w_{p} around n with the highest corresponding $\rho_{\min}$ from the previous stage. This process is repeated for a predefined number of times. Each stage further pinpoints the location of the vertex and describes in more detail the concavity we are searching for. Finally in the last stage, we have the area of apparent concavity amenable to being modeled by a *second-degree polynomial*.

Fig. 7 shows the algorithm for deciding whether to select a new n and if so, what its value should be. After a data block X_i is processed (indexed if similar to a Θ_j in buffer, added to the buffer if unique), it checks whether n should be switched. Each time a new n is selected, a switch counter k is incremented. The switch decision is made by checking if k is less than the maximum number of switches $k_{\max}$ and if r is greater than or equal to the minimum number of trials $r_{\min}$. Initially switching block sizes n are selected from random samples (without replacement), which are constructed from a discrete uniform distribution across all possible values of n.

When there are no random values remaining in the initial set of samples, the algorithm constructs the next $size$ samples within a window w_{p} around the best performing block size n_{best} from the previous samples. This selection and sample construction process repeats a predefined number of times l called stages.

When all samples from l stages are exhausted, it models a curve of n's by $\rho_{\min}$'s within a window w_{c} around n_{best} which is the best performing n with the largest $\rho_{\min}$ from all samples we have considered thus far. Here, the curve is a second-degree polynomial through $[n_{\mathrm{best}} - w_{\mathrm{c}}, n_{\mathrm{best}} + w_{\mathrm{c}}]$ by $\rho_{\min}$'s, from least-squares fitting.

If the block size yielding the vertex of the curve n_{new} is

```
# k is n switch count
# r is trial count
# w_c is window size for curve
# w_p is window size for sample
# w_n is denial window size for new n from curve
while incoming data stream to process do
    compare X_i to Θ_j's in buffer
    r ← r + 1
    if k < k_max and r ≥ r_min then
        if k ≥ |{sample}| · l then
            {curve} ← 2dPolynomial({n_best ± w_c}, {ρ_min})
            n_new ← argmax{curve}
            if n_new < n − w_n or n_new > n + w_n then
                n ← n_new
                reset buffer
                r ← −1
            else
                exit while loop
            end if
        else if {sample} = ∅ then
            {sample} ← rand(n_best ± w_p, size)
            n ← pop({sample})
            reset buffer
            r ← −1
        else
            n ← pop({sample})
            reset buffer
            r ← −1
        end if
        k ← k + 1
    end if
end while
```

Figure 7. Block size n selection using multistage random sampling.

outside a small window w_{n} around the previous n, n_{new} is assigned to n; if within the window, n_{new} is not picked and the algorithm terminates. The generally concave relationship between $\rho_{\min}$ and n within the $n_{\mathrm{best}} \pm w_{\mathrm{c}}$ window allows a second-degree polynomial to closely approximate it.

IV. Experimental Results

For performance testing, we use power grid monitoring data whose compression ratios by IDEALEM are shown in Fig. 3, where we searched the space of n's in the range of 10 to 100.

We use true optimal n's found in Fig. 3 to evaluate the effectiveness of the optimization algorithm in Table I. Here, **Dataset** is the data used for testing; $\boldsymbol{r_{\min}}$ is the minimum number of trials; **Model** is the n predicted by the optimization algorithm; **True** is the ground-truth n mentioned above; and $\boldsymbol{n_{\mathrm{diff}}}$ is the absolute error of predicted n. Model and True compression ratios are also shown for reference.

Table I
Predicted vs. True Optimal n and Compression Ratio (CR)

	Block Size				CR	
Dataset	$r_{\min}$	**Model**	**True**	n_{diff}	**Model**	**True**
A6BUS1C1MAG	100	22	49	27	145.08	275.1
	500	28		21	196.49	
	2800	39		10	178.79	
A6BUS1L1MAG	100	18	23	5	90.81	140.02
	500	21		2	97.19	
	2800	26		3	73.91	
BANK514C1MAG	100	68	62	6	149.18	291.54
	500	51		11	170.42	
	2800	67		5	90.44	
BANK514L1MAG	100	17	23	6	100.56	154
	500	20		3	102.07	
	2800	25		2	85.27	

All IDEALEM results use $\alpha = 0.01$ and $b = 254$. The multistage random sampling algorithm uses $w_{\mathrm{c}} = w_{\mathrm{p}} = w_{\mathrm{n}} = 20$, $size = 8$, $l = 3$, $n \in [8, 80]$, and $k_{\max} = |\{sample\}| \cdot l + 4$.

We report tests where $r_{\min} = 100, 500, 2800$, showing generally closer to optimal outcomes as larger $r_{\min}$ allows $\rho_{\min}$ to better approximate the true compression ratio. Large $r_{\min}$ generally increases compression ratio and the accuracy of predicted n, therefore reducing n_{diff}. The absolute error n_{diff} decreases 12% on average when $r_{\min}$ increases from 100 to 500, 23% when $r_{\min}$ increases from 500 to 2800, and 47% when $r_{\min}$ increases from 100 to 2800. These changes correspond to 0.02% decrease in n_{diff} for each unit increase in $r_{\min}$.

A. Discussion

In Table I, n_{diff} grows for A6BUS1L1MAG and BANK514C1MAG when $r_{\min}$ increases. These anomalies are likely due to uncertainty in $\rho_{\min}$ and the inherent stochastic nature of the optimization algorithm.

Interestingly, the optimizer performed better on A6BUS1L1MAG and BANK514L1MAG, where the compression ratio of IDEALEM is lower, than on A6BUS1C1MAG and BANK514C1MAG, where the compression ratio is higher. This suggests $\rho_{\min}$ may be an overly conservative approximation of compression ratio.

Among the parameters of the algorithm which are empirically found and set, $k_{\max} > |\{sample\}| \cdot l$ to give the optimizer ability to settle an optimal n by a few switches before algorithm convergence. If the denial window rejects n_{new} during this period, the algorithm terminates immediately by exiting the while loop. Decreasing $k_{\max}$ may prevent last-minute improvements for n; while increasing $k_{\max}$ may prevent the algorithm convergence.

It should be noted that in Table I, Model compression ratios are somewhat low despite the near optimal performance of n_{diff}. In fact, True compression ratio shown in

Table I follows the ground-truth results presented in Fig. 3; whereas Model compression ratio by Model n does not. This is because the algorithm keeps switching n until it converges, which in turn reduces the final compression ratio. When we consider continuous streaming data, though, the adversarial effect of switching n in the beginning would diminish. In addition, as shown in Fig. 3, compression ratio curves are not smooth and even very small n_{diff} can lead to a great loss of compression ratio. Finally, while small, the modified encoded stream structure of IDEALEM explained in Section II-D introduces overhead due to new markers.

V. Conclusion

We described our design and implementation of an online optimization algorithm for the parameters of streaming data compression method. We tested our algorithm on the statistical similarity based data compression method called IDEALEM to optimize its block size parameter. Our algorithm is designed to localize and describe a concavity curve resulting from a trade-off relationship through a series of increasingly accurate random sampling stages. The performance of each block size is calculated as a guaranteed minimum running compression ratio, which approximates the final compression ratio and accounts for uncertainty from minimized trials by modeling hits and misses as Bernoulli trials. We find this strategy effectively approximates an optimal block size. We plan to explore similar optimization problem for different applications with the same approach, as well as other unsupervised optimization schemes which function with given limited trials and potentially high uncertainty.

Acknowledgment

This work was supported by the Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was also supported by Institute for Information and communications Technology Promotion (IITP) grant (No. 2017-0-01779 statistical inference framework for explainable artificial intelligence) and Basic Science Research Program through the National Research Foundation of Korea (NRF) grant (NRF-2017R1A1A1A05001456) funded by the Korea government, Ministry of Science and ICT.

References

[1] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "ISABELA for effective in situ compression of scientific data," *Concurr. Comput. Pract. Exp.*, vol. 25, no. 4, pp. 524–540, 2013.

[2] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *Proc. Int'l Parallel Distrib. Process. Symp. (IPDPS '16)*, 2016, pp. 730–739.

[3] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec. 2014.

[4] D. Lee and J. Choi, "Learning compressive sensing models for big spatio-temporal data," in *Proc. Int'l Conf. Data Min. (SDM '15)*, 2015, pp. 667–675.

[5] D. Lee, J. Choi, and H. Shin, "A scalable and flexible repository for big sensor data," *IEEE Sensors J.*, vol. 15, no. 12, pp. 7284–7294, Dec. 2015.

[6] D. Lee, A. Sim, J. Choi, and K. Wu, "Novel data reduction based on statistical similarity," in *Proc. Int'l Conf. Scient. Stat. Database Manag. (SSDBM '16)*, 2016, pp. 21:1–21:12.

[7] ——, "Expanding statistical similarity based data reduction to capture diverse patterns," in *Proc. Data Compression Conf. (DCC '17)*, 2017, p. 445.

[8] ——, "Improving statistical similarity based data reduction for non-stationary data," in *Proc. Int'l Conf. Scient. Stat. Database Manag. (SSDBM '17)*, 2017, pp. 37:1–37:6.

[9] B. Behzad, S. Byna, S. M. Wild, Prabhat, and M. Snir, "Dynamic model-driven parallel I/O performance tuning," in *Proc. Int'l Conf. Clust. Comput. (CLUSTER '15)*, 2015, pp. 184–193.

[10] A. L. Chervenak, A. Sim, J. Gu, R. E. Schuler, and N. Hirpathak, "Adaptation and policy-based resource allocation for efficient bulk data transfers in high performance computing environments," in *Proc. Int'l Workshop Netw. Data Manag. (NDM '14)*, 2014, pp. 1–8.

[11] C. Dao, X. Liu, A. Sim, C. Tull, and K. Wu, "Modeling data transfers: change point and anomaly detection," in *Proc. Int'l Conf. Distributed Comput. Syst. (ICDCS '18)*, 2018, pp. 1589–1594.

[12] R. Kettimuthu, Z. Liu, I. Foster, P. H. Beckman, A. Sim, K. Wu, W.-K. Liao, Q. Kang, A. Agrawal, and A. Choudhary, "Towards autonomic science infrastructure: architecture, limitations, and open issues," in *Proc. Int'l Workshop Auton. Infrastructure Sci. (AI-Science '18)*, 2018, pp. 2:1–2:9.

[13] F. J. Massey Jr., "The Kolmogorov-Smirnov test for goodness of fit," *J. Am. Stat. Assoc.*, vol. 46, no. 253, pp. 68–78, Mar. 1951.

[14] G. Marsaglia, W. W. Tsang, and J. Wang, "Evaluating Kolmogorov's distribution," *J. Stat. Softw.*, vol. 8, no. 18, pp. 1–4, Nov. 2003.

[15] R. L. Wasserstein and N. A. Lazar, "The ASA's statement on p-values: context, process, and purpose," *Am. Stat.*, vol. 70, no. 2, pp. 129–133, Jun. 2016.

[16] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Springer, 2013.

[17] D. Lee and J. Choi, "Low complexity sensing for big spatio-temporal data," in *Proc. Int'l Conf. Big Data (BigData '14)*, 2014, pp. 323–328.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.