Dynamic Online Performance Optimization in Streaming Data Compression

J. Kade Gibson*
Texas A&M University-Commerce
Department of Computer Science
Commerce, Texas, USA
kade.gibson@tamuc.edu

Dongeun Lee*
Texas A&M University-Commerce
Department of Computer Science
Commerce, Texas, USA
dongeun.lee@tamuc.edu

Jaesik Choi
Ulsan National Institute of Science and Technology
School of Electrical and Computer Engineering
Ulsan, Korea
jaesik@unist.ac.kr

Alex Sim
Lawrence Berkeley National Laboratory
Scientific Data Management Group
Berkeley, California, USA
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 optimizes one of the key operation parameters known as block size, which determines the number of samples in a data block. We propose an online algorithm which takes account of generally concave trend of compression ratio curve, and adapts the block size to the optimal value which yields the maximum compression ratio.

CCS CONCEPTS

• **Theory of computation → Optimization with randomized search heuristics**; **Online learning algorithms**; *Data compression*; *Sketching and sampling*; • **Information systems** → *Data streaming*; • **Mathematics of computing** → Statistical paradigms;

KEYWORDS

Compression performance optimization, multistage random sampling, lossy compression, Kolmogorov-Smirnov test, time series data

*Both authors contributed equally to the paper.

ACM Reference Format:
J. Kade Gibson, Dongeun Lee, Jaesik Choi, and Alex Sim. 2018. Dynamic Online Performance Optimization in Streaming Data Compression. In *Proceedings of ACM Woodstock conference (SSDBM'18).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 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 [6]. Notwithstanding, there have been research efforts for the compression of this seemingly incompressible data type [2, 5, 6, 12]. 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 [7, 8].

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) [9–11], 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. IDEALEM has a few key parameters 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, still remains 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.

This paper proposes an online algorithm which optimizes the parameters of IDEALEM. Leveraging the generally concave trend of compression ratio curve, this algorithm dynamically finds and converges quickly on a block size which yields a near maximum compression ratio.[1] This approach can be applicable to other online parameter optimization problems such as data I/O performance parameters or network data transfers for the number of concurrent transfers, where the results show certain patterns such as a concave or convex curve. Other online optimization problems may include experimental testbeds to determine change point alerts for data collection performance, or autonomous control of the sensor device settings depending on the surrounding conditions.

2 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.[2]

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 [9]. 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

[1]The compression ratio is defined by the ratio of the original data size to the compressed data size.

[2]Note the term *exchangeability* here is used in somewhat wide sense. Rigorously speaking, random variables having the identical distribution are not necessarily exchangeable, although the converse is true [1].

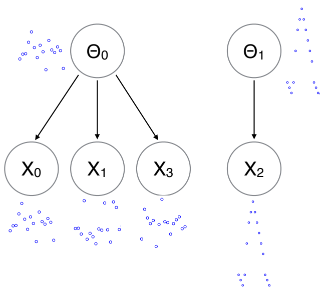


Figure 1: Input streaming data is divided into fixed-size blocks, each of which can be treated as an instantiation of a random variable X_i. IDEALEM compresses the input data by learning common probability distributions behind the group of random variables, which are represented by latent random variables 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.

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 of compression performance, without compromising reconstruction quality much [9]. This is possible because in many applications, if not all, an exact reproduction of a random fluctuation such as background noise is unnecessary.

The idea of learning common probability distributions has been recently extended to accommodate IDEALEM to non-stationary data with certain trends [10, 11]. With transformation methods such as residual transformation and delta transformation, we can remove trends in data and allow resulting sequences to be compared through the statistical similarity. The code of IDEALEM is available at http://datagrid.lbl.gov/idealem.

2.1 Similarity Measure

IDEALEM adopts the well-known KS test as its statistical similarity measure [13–15]. 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:

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

This standardized distance (2) converges to the inverse of the Kolmogorov distribution. As $\widetilde{D}_{n_i,n_j}$ grows, the value of the complementary cumulative distribution function (ccdf) for the Kolmogorov distribution yields a smaller value, which is dubbed the *p-value* [16].

The p-value is interpreted as the probability of obtaining a result equal to or more extreme than what was actually observed, assuming the *null hypothesis*, i.e., two random variables are from the same distribution, is true. Therefore, a small p-value indicates the null hypothesis 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. This does not directly assert whether two random variables are from the same distribution or not; rather, it is a way of identifying similar random variables from the perspective of data compression.

2.2 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 2.1 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 obscure due to the characteristics of the KS test and the design of IDEALEM [9].

We can observe the scaling factor $\sqrt{n_i n_j/(n_i + n_j)}$ in the standardized distance (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}. In other words, we can be more confident about the identity of two underlying probability distributions with a larger number of samples.

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.

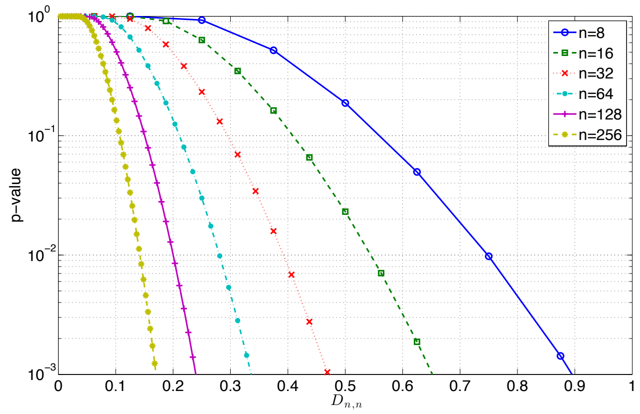


Figure 2: Effects of numbers of samples on the p-value and corresponding test statistics (distances) $D_{n,n}$, where the y-axis is drawn in log scale. Six numerical results are shown, where each has n sample points for two random variables. As n grows, it becomes more difficult to exchange random variables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

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 [9].) 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.

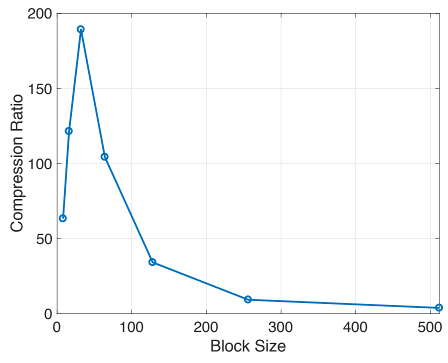
Considering these two aspects together, the effect of the block size n on the compression performance is not straightforward. Although the compression ratio should increase with n in principle, it becomes also difficult to pass the KS test as n increases, as shown in Fig. 2.

2.3 Concavity in Compression Ratio

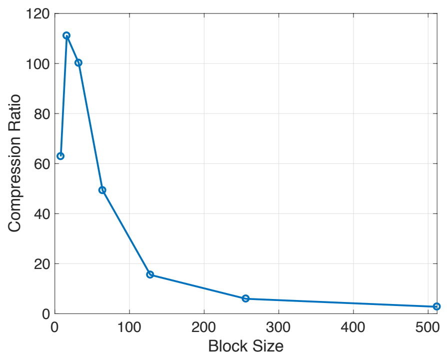
The two different forces acting on the compression ratio with reference to the block size n result in a *concave trend* in the compression ratio: as n increases, the compression ratio also increases thanks to its linearly growing relationship with n; but after a certain point, the difficulty of passing 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 three different similarity thresholds α. In order to see how the number of buffers b affects the compression ratio, Fig. 4 displays the contour plot of compression ratio with varying n's and b's, with $\alpha = 0.01$. In Fig. 4, we see more buffers increase a compression ratio. More interestingly, with any given b, the compression ratio always follows the concave trend with varying n.

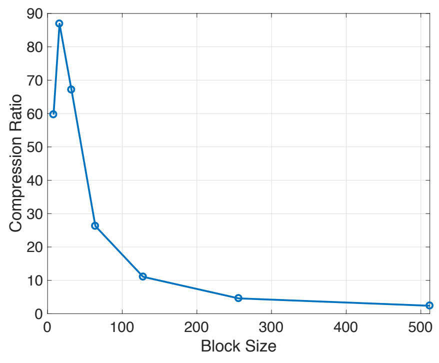
In an ideal condition where an entire data stream can be represented by a single distribution, the compression ratio would follow the theoretical upper bound $8n$ and b would be essentially irreverent. On the other hand, for a data stream where every data block is dissimilar from each other, the compression ratio would be less than 1 regardless of b, due to the index overhead in the encoded stream structure as explained in Section 3. However, in practice, these two



(a) $\alpha = 0.01$



(b) $\alpha = 0.05$



(c) $\alpha = 0.1$

Figure 3: Compression ratio variations of IDEALEM with power grid monitoring data when $b = 255$. Depending on the similarity threshold α, maximum compression ratios and corresponding block sizes n vary; however, all of their trends are concave within a certain block size range.

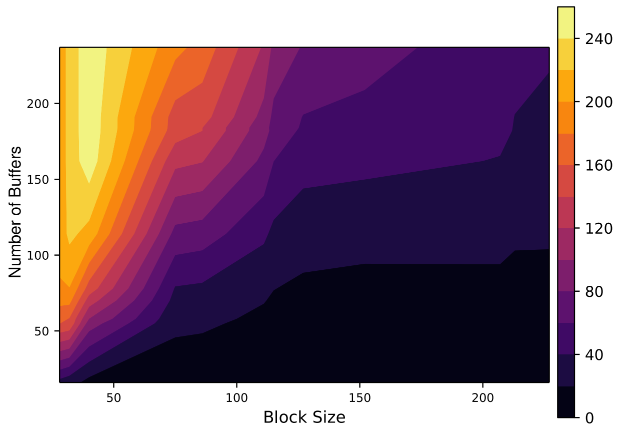


Figure 4: Contour of compression ratio variations with power grid monitoring data (100 MB) at $\alpha = 0.01$ over $b \in [3, 255]$ and $n \in [8, 255]$. More buffers tend to increase a compression ratio. With any given b, the compression ratio tends to follow the concave trend with varying n.

extreme conditions do not typically occur. Thus, we assume the trend of compression ratio is always concave in this work.

3 ENCODED STREAM STRUCTURE

Fig. 5 illustrates an example of the modified encoded stream structure from [9]. 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 starting block size n in 1 byte so the decoder knows the initial block size. Thus, $0 \leq n \leq 255$ with this design.

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.[3] 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 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 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.

[3]Counting starts from 0.

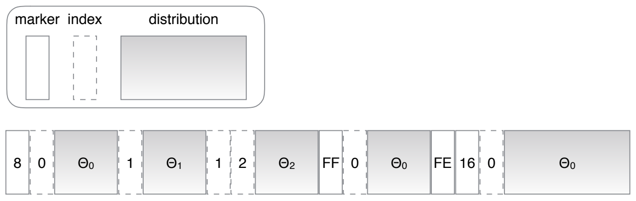


Figure 5: An example of the modified encoded stream structure from original IDEALEM to incorporate dynamic block size changes. The block size changes from $n = 8$ to $n = 16$ and $b = 3$. A solid box represents a marker in 1 byte; a dotted box an index j in 1 byte; a solid box with gradation a distribution Θ_j in $8n$ bytes. The marker is used for signaling buffer overwrite `0xFF`, the block size change `0xFE`, and a new n.

4 BLOCK SIZE OPTIMIZATION

The goal of our algorithm is to find the vertex of compression ratio curve shown in Fig. 3, as quickly and accurately as possible, minimizing exploration time and maximizing compression performance. However, this is not straightforward due to the difficulty of learning compression ratio at a given block size n. Note Fig. 3 and Fig. 4 show the ground-truth results of compression ratios for different combinations of parameters. Because they require the entire dataset to derive, and are unique to each dataset, they cannot be used directly for calculating some cost to optimize on.

4.1 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 3, 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.[4]

Specifically, we call 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:

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

[4]When every buffer is empty, i.e., in the very beginning of the encoded stream or right after the 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.

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.

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:

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

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

Proposition 4.1. *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. 6 shows the growth of $\rho_{\min}$ over $r \in [1, 100]$ for $n = 14, 46, 75, 80$. In Fig. 6, $\rho_{\min}$ grows continuously and the rank of each n changes frequently in this range. Although the approximation accuracy of $\rho_{\min}$ increases as r approaches infinity in principle, more trials are required to stabilize $\rho_{\min}$, especially the ranks of different block sizes n.

Moreover, 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, which means we discard all the previously collected distributions Θ_j, along with h and r. As shown in (3) and Fig. 6, when the trial count r is low, much uncertainty exists in the success probability p and therefore $\rho_{\min}$ increases both unstably and quickly as we have more trials. In other words, resetting mature buffers is a very expensive operation and should be carefully done in *correct timing*, because it is irrevocable and repeating trials for the same n is unreasonable.

Thus we should obtain a number of r for a given n to have a relatively stable $\rho_{\min}$. On the other hand, too many trials impedes the convergence of our online algorithm, which leads to degradation in overall compression performance. It is therefore important to switch to another n not too early, but not too late. Once the algorithm finds the optimal n, it stops and no longer changes the block size.

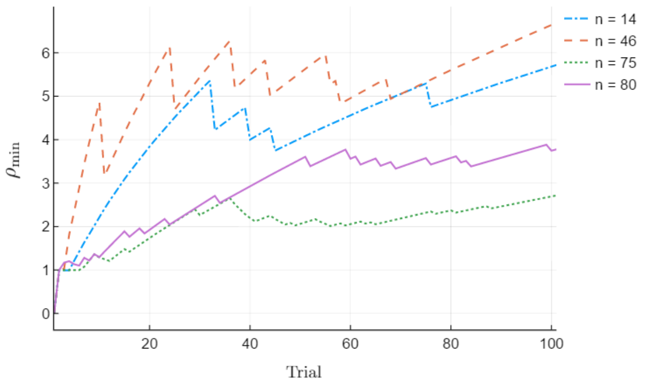


Figure 6: Growth of $\rho_{\min}$ over $r \in [1, 100]$ with $b = 254$, $\alpha = 0.01$, and $n = 14, 46, 75, 80$. Here, $\rho_{\min}$ grows continuously and the rank of n changes frequently.

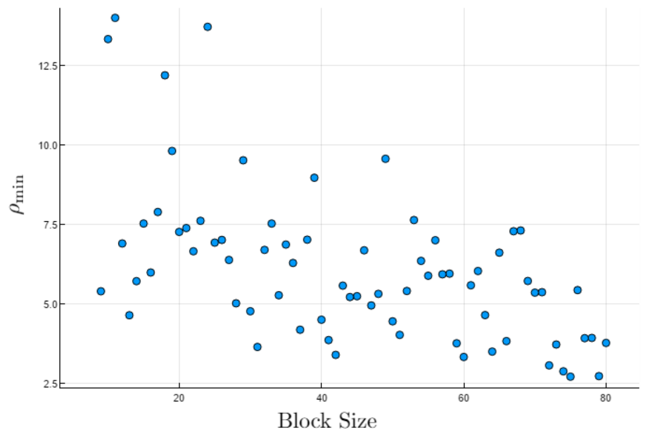


Figure 7: The guaranteed minimum running compression ratio $\rho_{\min}$ approximates the final compression ratio for $n \in [8, 80]$ at $r = 100$, $b = 254$, and $\alpha = 0.01$.

4.2 Multistage Random Sampling

Fig. 7 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 $r = 100$, which helps considerably when modeling the concave trend. As shown in Fig. 7, 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, 7]. Compared with uniform sampling, the random sampling requires a smaller number of samples to capture the same amount of information. In particular, it is shown one needs $O(K \log N)$ samples to recover an original data sequence, where K is the number of sparse components in a known basis; N is the number of samples in the original data sequence. In our scenario, we have prior knowledge of general concavity and especially assume $K = 1$ in the last stage where we model the concavity with a quadratic polynomial (second-degree polynomial).

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 are subject to noise [3]. 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 the range where the concavity lies, but it does not describe it sufficiently for modeling a concave curve. To this end, we use a small size for this initial sample.

After the initial sampling, another is taken using a window of size w_p around the n with the highest corresponding $\rho_{\min}$ from the previous stage. This process is repeated some 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.

Algorithm 1 shows the procedure 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 a 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 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 sample, it constructs the next samples within a window w_p around the best performing n from the previous samples. This selection and sample construction process repeats a predefined number of times l called stages. When all samples 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 $[n_{\text{best}} - w_c, n_{\text{best}} + w_c]$ by $\rho_{\min}$'s, from least-squares fitting. If the vertex of the curve is outside a small window w_n around the previous n, the value is assigned to n; if within the window a new n is not picked. The generally concave relationship between $\rho_{\min}$ and n within the $n_{\text{best}} \pm w_c$ window allows a second-degree polynomial to closely approximate it, which makes this method efficient and effective.

Algorithm 1 Block Size n Switch and Selection Procedure

```
1: # $k$ is count of $n$ switches
2: # $r$ is count of trials
3: # $w_c$ is window size for curve
4: # $w_p$ is window size for sample
5: # $w_n$ is denial window size for new $n$ from curve
6: while incoming data stream to process do
7:     compare $X_i$ to $\Theta_j$'s in buffer
8:     $r \leftarrow r + 1$
9:     if $k < k_{\max}$ and $r \geq r_{\min}$ then
10:        if $k \geq$ length(sample) $\cdot\, l$ then
11:            curve $\leftarrow$ 2dPolynomial($\{n_{best} \pm w_c\}$, $\{\rho_{\min}\}$)
12:            $n_{\text{new}} \leftarrow$ argmax (curve)
13:            if $n_{\text{new}} < n - w_n$ or $n_{\text{new}} > n + w_n$ then
14:                $n \leftarrow n_{\text{new}}$
15:                reset buffer
16:            end if
17:        else if isempty(sample) then
18:            sample $\leftarrow$ rand($n_{\text{best}} \pm w_p$, size)
19:            $n \leftarrow$ pop(sample)
20:            reset buffer
21:        else
22:            $n \leftarrow$ pop(sample)
23:            reset buffer
24:        end if
25:        $k \leftarrow k + 1$
26:    end if
27: end while
```

5 RESULTS

For performance testing we use approximately two weeks of power grid monitoring data collected by two different sensors (A6BUS1 and BANK514). Each sensor monitors three-phase voltages and currents, collecting magnitude (MAG) and phase angle (ANG) measurements. In this work, we demonstrate the performance of our algorithm on phase 1 MAG data for both current (C) and voltage (L).

To find the ground truth for the maximum compression ratio of IDEALEM and the optimal block size n for each dataset, we searched the entire space of n's by running IDEALEM with block sizes in the range of 10 to 100, as shown in Fig. 8. We observe compression ratios in Fig. 8 vary substantially across different n's; even small errors in n's may lead to drastic deflation in the compression ratios.

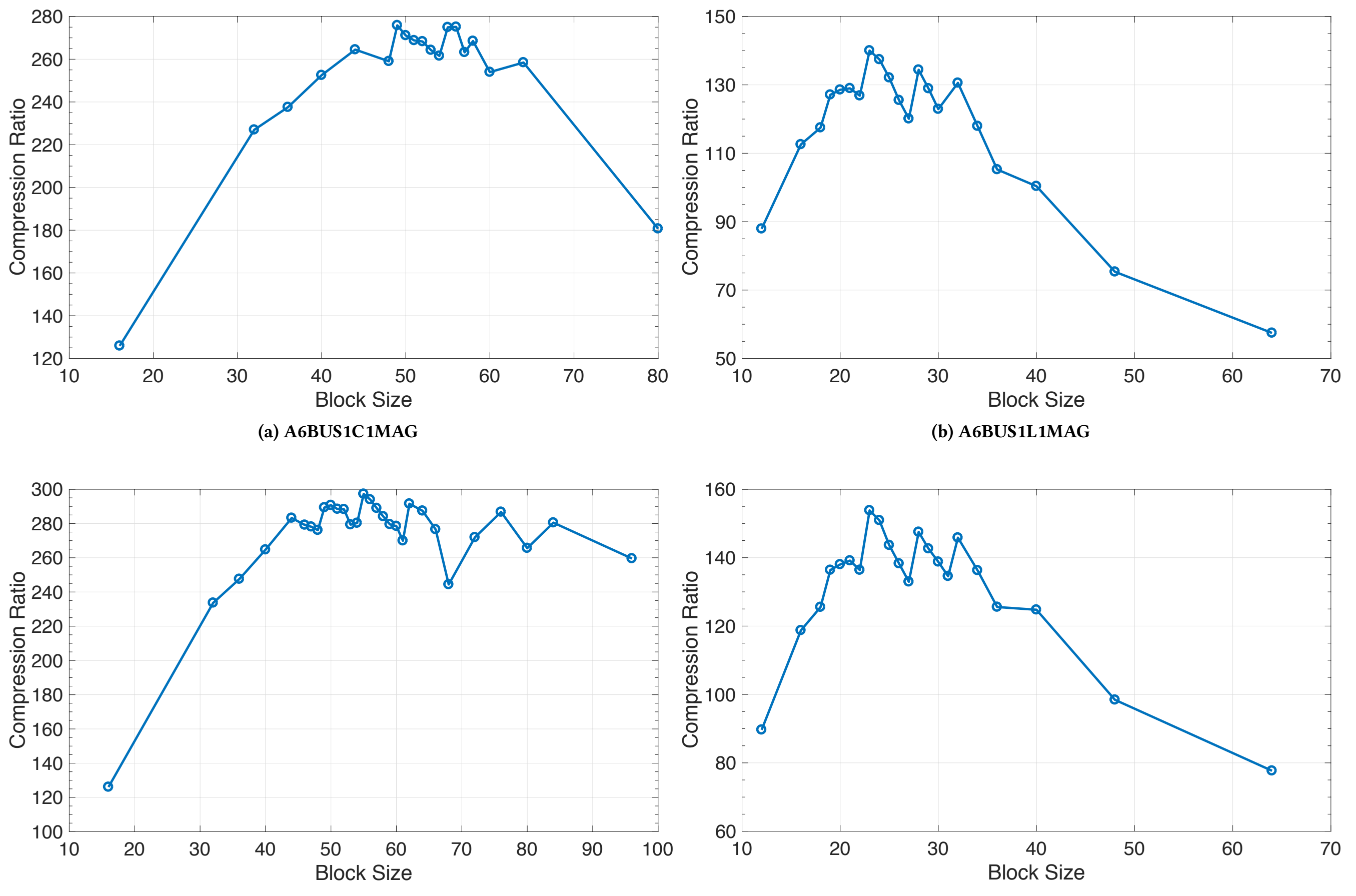
We use true optimal n's found in Fig. 8 to evaluate the effectiveness of the optimization algorithm in Table 1. Here, Dataset is the data used for testing, $r_{\min}$ is the minimum number of trials per n, Model is the n predicted by the optimization algorithm, True is the ground-truth n mentioned above, and $n_{\text{diff}} = |\text{Model} - \text{True}|$ is the absolute error of predicted n. Model and True compression ratios are also shown. All IDEALEM results use $\alpha = 0.01$ and $b = 254$. All IDEALEM with block size optimization results use $w_c = w_p = w_n = 20$, random sample size $= 8$, $l = 3$, $n_{\min} = 8$, $n_{\max} = 80$, and $k_{\max} = \text{length(sample)} \cdot l + 4$.

Three factors contribute to deflating the compression ratio of the optimizer here. (1) We use 100MB data for these tests, so higher values of $r_{\min}$ reduce compression ratio due to the samples consuming more data. For instance, where $r_{\min} = 2800$ the sample tends to consume ~15MB data, so the selected n is only used for ~ 85% of the input data, as explained in Section 4.2. (2) The buffer with $b = 254$ is not being filled with only 100MB – random sample consumption data available, as explained in Section 4.1 and illustrated by Fig. 4. (3) While small, the addition of the block size index to the encoded stream introduced in Section 3 also reduces the compression ratio. Due to this compression ratio deflation, n_{diff} is more telling than the compression ratios when evaluating optimization performance.

We report tests where $r_{\min} = 100, 500$ and 2800, showing generally closer to optimal outcomes as higher values allow $\rho_{\min}$ to better approximate the true compression ratio. $r_{\min}$ behaves similarly to b, where increasing it generally increases compression ratio and the accuracy of predicted n, therefore reducing n_{diff}, as the input size grows infinitely. n_{diff} decreases 12% on average where $r_{\min}$ is increased from 100 to 500, 23% where $r_{\min}$ is increased from 500 to 2800, and 47% where $r_{\min}$ is increased from 100 to 2800. These changes represent a 0.03%, 0.01%, 0.02%, and average 0.02% per-unit decrease in n_{diff} for each unit increase in $r_{\min}$. This has two implications: 1) $\rho_{\min}$ generally becomes significantly more accurate as $r_{\min}$ increases and 2) $\rho_{\min}$ should not suffer from diminishing returns as the relationship is generally linear. However, n_{diff} increases for A6BUS1L1MAG and BANK514C1MAG where $r_{\min} = 2800$ and 500 compared to $r_{\min} = 500$ and 100, respectively. These anomalies are likely due to epistemic uncertainty in $\rho_{\min}$ and the inherent stochastic nature of the optimization algorithm. Interestingly, the optimizer performed better on A6BUS1L1MAG and BANK514L1MAG, data where the compression ratio of IDEALEM is comparatively lower, than on A6BUS1C1MAG and BANK514C1MAG, where the compression ratio of IDEALEM is comparatively higher. This suggests $\rho_{\min}$ may be an overly conservative approximation of compression ratio. This is also supported by the optimizers bias toward smaller n's, where the KS-test is easier to pass and hit ratios are higher, particularly as shown in the predicted n's for A6BUS1C1MAG.

As w_c increases more $(n_i, \rho_{\min_i})$ points outside the concavity are included when curve fitting, which prevents the vertex of the curve from approximating the optimal n. The converse is true as w_c decreases, and has similarly negative results. As w_n increases or decreases it expands or contracts the algorithm's ability to explore new n's. When w_n is too large the first n selected by the curve is the last, as all future selections are contained in the range $[n_{\text{new}} - w_n, n_{\text{new}} + w_n]$; when too small predicted n jumps between adjacent n's frequently until hitting $k_{\max}$, which prevents a mature buffer from accumulating and $\rho_{\min}$ from accurately approximating the true compression ratio. As w_p increases the range of n's available to random sampling stages 2 through l approaches $[n_{\min}, n_{\max}]$. As w_p decreases the range of n's available approaches $[n_{\text{best}}, n_{\text{best}}]$. Both extremes impede concavity localization and description, and distort the modeled curve.

Concavity localization and description performance is hampered as random sample size is reduced. Increasing random sample size toward covering $n_{\min}$ to $n_{\max}$ improves performance, but increases exploration time and data consumption. Empirically we found total random sample size of 24 split evenly between three stages works well for the n range used. We set $n_{\min}$ and $n_{\max}$ based on finding the concavity in the relationship between n and $\rho_{\min}$ is consistently



(a) A6BUS1C1MAG (b) A6BUS1L1MAG (c) BANK514C1MAG (d) BANK514L1MAG

Figure 8: Ground truth compression ratios 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 the generally concave trend of compression ratio curves, they are not smooth due to many peaks and valleys around global maxima.

present where $8 \leq n \leq 80$. As the range of possible n's expands stage one random sample size should be increased, as it serves primarily to localize the concavity. Sample size for stages $2 \ldots l$ need little or no change, as they primarily serve to describe the concavity. While shrinking the $[n_{\text{min}}, n_{\text{max}}]$ range decreases the samples needed to localize the concavity, it risks excluding the optimal n from the search space.

We use a small k_{max} because the algorithm tends to select and retain an n a few switches into curve fitting. Decreasing it further prevents the small improvements which are made to this selection. Increasing k_{max} allows the algorithm to retain an n for tens of thousands of trials, then select a new one, resulting in loosing mature buffers, which can severely negatively impact compression ratio. The negative effects of reseting mature buffers were discussed in Section 4.1.

6 CONCLUSIONS

In this paper we describe our design and implementation of an online optimization algorithm for the statistical similarity based stream compressor IDEALEM. IDEALEM breaks a data stream into blocks of some size and uses the KS-test for similarity between them. When found similar, blocks are index as a member of another distribution. This work presents an algorithm for optimizing the block size while minimizing trials. In theory, compression ratio increases with block size. In practice, the difficulty of passing the KS-test overwhelms large block sizes. This results in a concavity in the relationship between block size and compression ratio. Our algorithm is designed to localize and describe the concavity through a series of increasingly accurate random block size sampling stages, then model it as a two dimensional polynomial. The performance of each block size is calculated as guaranteed minimum running compression ratio, which approximates final compression ratio and accounts for uncertainty from minimized trials by modeling hits and

Table 1: Predicted vs. True Optimal n and Compression Ratio

		Block Size			**Compression Ratio**	
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	

misses as Bernoulli trials. We find this strategy effectively approximates optimal block size, but is sensitive to aggressively minimizing trials. Additionally, the initial stochasticity of this method gives non-deterministic results.

In future work, we will explore methods for improving the approximation of guaranteed minimum running compression ratio given limited trials; specifically to reduce epistemic uncertainty at higher compression ratios. It would be interesting to include further metaparameter optimization such as detecting optimal w_c, w_n, w_p, random sample size, l, and $r_{\min}$ through statistical measures of observed data. We also plan to explore similar optimization problem for different applications with the same approach, as well as other unsupervised optimization schemas which function with given limited trials and potentially high uncertainty.

ACKNOWLEDGMENTS

The authors gratefully acknowledge helpful discussions with K. John Wu, Kristofer Bouchard, David Donofrio, Emma Stewart, Sean Peisert, Chuck McParland, Reinhard Gentz, Mahdi Jamei, Ciaran Roberts, and Sebastian Ainslie. 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 Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (NRF-2014R1A1A1002662, NRF-2014M2A8A2074096). This research used resources of the National Energy Research Scientific Computing Center.

REFERENCES

[1] Jaesik Choi, Kejia Hu, and Alex Sim. 2013. *Relational dynamic Bayesian networks with locally exchangeable measures.* Technical Report LBNL-6341E. Lawrence Berkeley National Laboratory.
[2] Sheng Di and Franck Cappello. 2016. Fast error-bounded lossy HPC data compression with SZ. In *Proc. Int'l Parallel Distrib. Process. Symp. (IPDPS '16).* 730–739.
[3] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (June 1981), 381–395.
[4] Simon Foucart and Holger Rauhut. 2013. *A Mathematical Introduction to Compressive Sensing.* Springer.
[5] Jeremy Iverson, Chandrika Kamath, and George Karypis. 2012. Fast and effective lossy compression algorithms for scientific datasets. In *Proc. Int'l Eur. Conf. Parallel Distributed Comput. (Euro-Par '12).* 843–856.
[6] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Seung-Hoe Ku, Choong-Seock Chang, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. 2013. ISABELA for effective in situ compression of scientific data. *Concurr. Comput. Pract. Exp.* 25, 4 (2013), 524–540.
[7] Dongeun Lee and Jaesik Choi. 2015. Learning compressive sensing models for big spatio-temporal data. In *Proc. Int'l Conf. Data Min. (SDM '15).* 667–675.
[8] Dongeun Lee, Jaesik Choi, and Heonshik Shin. 2015. A scalable and flexible repository for big sensor data. *IEEE Sensors J.* 15, 12 (December 2015), 7284–7294.
[9] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. 2016. Novel data reduction based on statistical similarity. In *Proc. Int'l Conf. Scient. Stat. Database Manag. (SSDBM '16).* 21:1–21:12.
[10] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. 2017. Expanding statistical similarity based data reduction to capture diverse patterns. In *Proc. Data Compression Conf. (DCC '17).* 445.
[11] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. 2017. Improving statistical similarity based data reduction for non-stationary data. In *Proc. Int'l Conf. Scient. Stat. Database Manag. (SSDBM '17).* 37:1–37:6.
[12] Peter Lindstrom. 2014. Fixed-rate compressed floating-point arrays. *IEEE Trans. Vis. Comput. Graphics* 20, 12 (December 2014), 2674–2683.
[13] George Marsaglia, Wai W. Tsang, and Jingbo Wang. 2003. Evaluating Kolmogorov's distribution. *J. Stat. Softw.* 8, 18 (November 2003), 1–4.
[14] Frank J. Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *J. Am. Stat. Assoc.* 46, 253 (March 1951), 68–78.
[15] Leslie H. Miller. 1956. Table of percentage points of Kolmogorov statistics. *J. Am. Stat. Assoc.* 51, 273 (March 1956), 111–121.
[16] Ronald L. Wasserstein and Nicole A. Lazar. 2016. The ASA's statement on p-values: context, process, and purpose. *Am. Stat.* 70, 2 (June 2016), 129–133.