Novel Data Reduction Based on Statistical Similarity

Dongeun Lee*
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA
eundong@lbl.gov

Alex Sim
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA
asim@lbl.gov

Jaesik Choi
Ulsan National Institute of
Science and Technology
Ulsan, 44919, Korea
jaesik@unist.ac.kr

Kesheng Wu
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA
kwu@lbl.gov

ABSTRACT

Applications such as scientific simulations and power grid monitoring are generating so much data quickly that compression is essential to reduce storage requirement or transmission capacity. To achieve better compression, one is often willing to discard some repeated information. These lossy compression methods are primarily designed to minimize the Euclidean distance between the original data and the compressed data. But this measure of *distance* severely limits either reconstruction quality or compression performance. We propose a new class of compression method by redefining the distance measure with a statistical concept known as exchangeability. This approach reduces the storage requirement and captures essential features, while reducing the storage requirement. In this paper, we report our design and implementation of such a compression method named IDEALEM. To demonstrate its effectiveness, we apply it on a set of power grid monitoring data, and show that it can reduce the volume of data much more than the best known compression method while maintaining the quality of the compressed data. In these tests, IDEALEM captures extraordinary events in the data, while its compression ratios can far exceed 100.

CCS Concepts

•**Theory of computation** → **Data compression;** •**Mathematics of computing** → Bayesian nonparametric models; Time series analysis; •**Information systems** → Data streaming; Similarity measures;

Keywords

Floating-point data, locally exchangeable measure, lossy compression, online algorithm, time series data

1. INTRODUCTION

Compression is a common technique for reducing storage requirement or transmission capacity. As computer systems and devices gather more and more data, our ability to store or transmit data records is significantly challenged. In these cases, compressing the data could provide an effective path to address such a challenge. To reduce the compressed data volume, we may drop some information in the original data. These techniques are known as lossy compressions. The quality of these techniques is judged currently by the Euclidean distance between the original data and the decompressed version of the compressed data. This focus on a single quality measure has imposed a significant limitation on the effectiveness of the compression methods. To break this limitation, we propose a new type of compression method based on a statistical concept known as exchangeability. This approach allows us to capture common data blocks in a very compact form while preserving key statistical properties. With a carefully designed algorithm, it can capture the essential characteristics of the data while using only a very small fraction of bytes needed by the original data records.

In general, compression takes advantage of repeated patterns or common features in the original data. In this regard, our approach is no different from others. What is different is that we consider the distribution of the values as a basic pattern. To quantify the similarity of two data blocks, we extend the concept of exchangeability with the locally exchangeable measure (LEM) [8]. When the LEM value passes a given threshold, we consider two blocks as exchangeable and choose to store only one of them as a representative. If new blocks are found to be exchangeable with the stored block, it is unnecessary to store the new blocks.

Since our approach relies on statistical properties of data values, it can only work on numerical values. Numerical values, especially floating-point values, are known to be particularly hard to compress [4, 11]. Recently, there has been a flurry of publications on compressing floating-point values [15, 10, 11, 4]. However, all these techniques are designed to reduce the Euclidean distance between the original data and the decompressed data. ZFP [15], which is one of those

*Dongeun Lee is jointly affiliated with Ulsan National Institute of Science and Technology.

SSDBM '16, July 18–20, 2016, Budapest, Hungary

DOI: http://dx.doi.org/10.1145/2949689.2949708

techniques, has been demonstrated to be able to reduce the volume of data by 100-fold, while maintaining good quality. In this case, ZFP relies heavily on continuity present in 3D simulation data. There are many applications where such continuity is not present; rather, variations in data values might be in some sense small but apparently random. Intuitively, such data should be compressible, but even the state-of-the-art floating-point value compression techniques fail to achieve good compression in such cases. We believe the key limitation of the existing compression methods is that they attempt to reproduce every rise and fall in the original data records, where preserving key statistical properties might be sufficient for the applications.

By focusing on the statistical distribution of data, our approach does not aim to reproduce the original data with small Euclidean distances, instead we produce decompressed data that have the same distribution as the original data. This is a significant departure from the common practice in designing compression techniques. In this work, we propose a practical algorithm to realize this unique approach. This design also allows us to capture rare events precisely, which can be very useful for analysis tasks. Our design additionally allows analysis operations on the compressed data without decompression. This is a useful feature that we plan to explore in future work.

In this work, we provide an effective implementation of the algorithm, and demonstrate the effectiveness of the new compression scheme on a set of power grid monitoring data. Experimental results show that our new scheme can reduce the data volume by more than 100-fold, while retaining key features of the original data.

The rest of this paper is organized as follows. In Section 2, we briefly review related work and discuss the key design considerations of the new algorithm. In Section 3, we discuss a similarity measure that could be used with the LEM concept. The details of the IDEALEM implementation is presented in Section 4, and an extensive evaluation of IDEALEM is given in Section 5. We conclude with a brief summary and the discussion of future work in Section 6.

2. MOTIVATION: NEW PERSPECTIVE ON DATA COMPRESSION

Data compression reduces the space needed to represent some information. This is accomplished by identifying and using structures that exist in the data [20]. A data compression method is categorized into two broad classes: **lossless coding** where a reconstruction of compressed data is identical to the original data; and **lossy coding** where a reconstruction is different from the original data. Typically, a lossy coding can provide much better compression than a lossless one. When one can tolerate a certain amount of distortion [22, 19], the quality of data can be adjusted in favor of better compression with the lossy coding [13]. Next, we briefly review related compression methods and highlight two design considerations that drive our work on the new compression method named IDEALEM (Implementation of Dynamic Extensible Adaptive Locally Exchangeable Measures) [2].[1] The first one is redefining the distance (similarity) measure to relax the order of values to increase the possibility of compression, and the second is allowing analysis to be performed directly on the compressed data.

[1]Code is available.

2.1 Relaxing Order of Values

To quantify information loss, lossy coding methods treat incoming data as numbers. Most of the lossy coding methods could be described as a transformation of these numbers to produce a compact form. Since our work is originally motivated by time series of measurements, our review of lossy codings will concentrate on those designed for time series or arrays of values. A common transformation used by lossy coding methods is quantization [20]. Some of the most effective compression techniques, such as ZFP [15] and SQE [10], are based on variant forms of quantization.

The information loss due to compression is generally measured by the Euclidean distance (ℓ_2 distance) between reconstructed data and the original data. Given some constraints such as the number of bits to be used per value, the most successful lossy compression methods are designed to minimize objectives such as the mean squared error (MSE) or to maximize objectives such as the signal-to-noise ratio (SNR) [18, 20, 12, 13]. One fundamental limitation of this approach is that the order of the values is preserved. For example, if values are gradually increasing in the original input, the reconstructed data will also be roughly increasing. In some applications, the order of these values is not important, for example, when the original data values are assumed to be generated from random number generators, then only probability distributions that explain the source of data are the essential character of the data. Giving up preserving the order of the incoming values should lead to better compression.

Of course, application data could not be explained by random numbers. However, in some situations, devices such as sensors might be measuring background noise during the majority of their operation time. For example, during normal operations of a computer network, monitoring devices will be observing *random* traffic. Similarly, during normal operations of a transformer on a power grid, a micro phasor measurement unit (μPMU) will be measuring *random* fluctuations. In these cases, faithfully reproducing the random fluctuations is not necessary. Section 5 demonstrates the reconstruction results of IDEALEM, which shows relaxing the order of data sequence is a reasonable approach.

2.2 Allowing Analysis without Decompression

Widely used general-purpose compression tools, such as gzip [1], are based on lossless coding methods [28, 20]. However, these methods are known to not perform well on numerical values, especially on floating-point numbers from scientific simulation or high-precision sensor measurements [11, 15]. The key reason for this poor compression is that the numerical values are usually different from each other, even though the differences might be small. When the differences are relatively small, the lossy coding could capture the essential character of data without losing much information and therefore are good alternatives to the lossless coding.

Lossless coding methods similar to IDEALEM lie in the area of biological sequence compression, especially deoxyribonucleic acid (DNA) compression [5]. Some DNA sequences are highly repetitive, but they are not exactly identical to the original sequence as nucleotides can be changed, inserted, or deleted. This is the reason why most conventional dictionary-based algorithms [28, 25, 20] fail to compress DNA data, as they all try to look up the same recurring

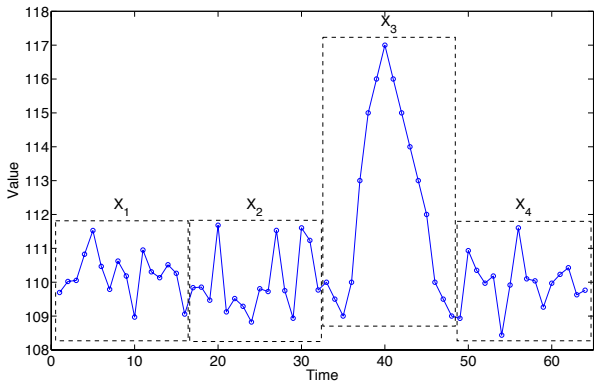


Figure 1: An example time series and corresponding random variables. All random variables except X_3 were generated from the normal distribution $\mathcal{N}(110, 1)$. X_1, X_2, and X_4 look visually similar, as they share the same underlying probability distribution.

pattern stored in the dictionary. To handle this, substitution approaches exploiting approximate repeats have been proposed [6, 7]. IDEALEM also reconstructs data from learned patterns during the encoding process that do not need to be identical to the original sequence.

It would be desirable if these compression methods permit certain analysis operations to be directly performed on compressed data without decompression, because these methods in the broad sense preserve selected values from the input data, along with side information for the reappearance of these values. However, the complexities of encoded data structures hamper direct analysis on the compressed data. On the other hand, IDEALEM is designed to have a simple encoded structure that allows certain analysis to be performed on the compressed data without decompression.

3. SIMILARITY MEASURE

Fig. 1 shows time series data of total 64 samples. If we assume that each sequence of 16 samples is an instantiation of a random variable X_i $(i = 1, \ldots, 4)$, we can consider similarities between these random variables. In Fig. 1, X_1, X_2, and X_4 all look visually similar; whereas X_3 looks different from other random variables. The design of IDEALEM is based on these observations: we may represent X_1, X_2, and X_4 using a single random variable, claiming that three random variables have an identical distribution behind.

Fig. 2 displays the graphical model representation of the observations shown in Fig. 1. We conceive a latent random variable Θ_j $(j = 1, 2)$ that governs random variables sharing the common distribution. In this paper, we focus on a practical data compression scheme leveraging the identical distribution shared by random variables with the same parent Θ_j, rather than consider relationships between these latent variables and infer the exchangeability of a new random variable for dynamic sampling, as discussed in the previous work [8].

Specifically, if we keep only a single sequence (distribution) from one of three random variables, i.e., X_1, X_2, and X_4, we can achieve data reduction with the **compression ratio** of 3, where the compression ratio is defined by the

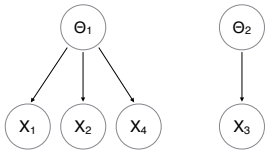


Figure 2: Graphical model representation of Fig. 1. Latent random variables Θ_1 and Θ_2 govern common distributions behind. These distributions are non-parametric, allowing any shapes of distributions.

ratio of the original size to the compressed size. Here X_1, X_2, and X_4 are *exchangeable* in the sense that we could represent any of them with each other. Therefore, the more similar random variables there are, the higher compression ratio we can achieve.

Kolmogorov-Smirnov Test

In order to design a working algorithm, we have to answer a question: how to measure similarity? Based on a given similarity measure, our compression procedure can determine whether X_i is similar to other random variables and then proceed to encode the block of data for compression.

Since Kolmogorov-Smirnov test (KS test) is the most popular test for the similarity of time series we know of [27, 21, 17], we have decided to use KS test as the first similarity measure. Technically, KS test is a non-parametric statistical hypothesis testing method that can test whether two underlying one-dimensional probability distributions of random variables differ or not [16, 9]. In addition to being widely used, KS test is also much easier to compute than related statistical tests such as Anderson-Darling test [9]. By choosing KS test, we aim for a relatively inexpensive compression method.

Since the KS test is non-parametric, it can compare two random variables from any arbitrary distributions without specifically assuming parametric distributions.[2] In particular, the maximum distributional distance (gap) 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. Fig. 3 shows an example of two empirical distributions where we can clearly see a distributional distance between them. The distance (1) is also called the **test statistic**, which is subsequently *standardized* with respect to n_i and n_j as follows:

$$D_{n_i,n_j}\sqrt{\frac{n_i n_j}{n_i + n_j}}. \tag{2}$$

This standardized distance (2) converges to the inverse of the Kolmogorov distribution. As the standardized distance (2) grows, the value of the complementary cumulative distribution function (ccdf) regarding the Kolmogorov distribution yields a smaller value, which is dubbed the **p-value** [24].

The p-value is interpreted as the probability of obtaining a result equal to or more extreme than what was actually observed, assuming that the **null hypothesis**, i.e., two

[2]We employ the two-sample KS test.

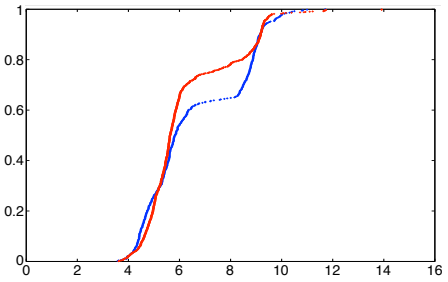


Figure 3: An example of two empirical distributions in blue and red colors. The gap (distributional distance) is visible in the middle. The maximum distance, defined by (1), ranges between zero and one.

random variables are from the same distribution, is true. Therefore, a small p-value indicates that 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.[3] If a p-value is less than or equal to a chosen α, we reject the null hypothesis, supporting its logical complement. However, this does not necessarily mean the logical complement of the null hypothesis is true. In other words, we cannot assure that two random variables are not from the same distribution, because there is also a small chance of the null hypothesis being true despite high improbability.

Therefore, with the chosen α, we can only say the **Type I error** rate is at most α, which is the incorrect rejection of a true null hypothesis, i.e., false positive. 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.

Sensitivity with Number of Samples

We can observe that 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, large n_i and n_j can yield the same standardized distance (2) with a small D_{n_i,n_j}. In other words, we become more confident about the identity of two underlying probability distributions with large samples. This in turn means more sensitivity in deriving the p-value with a given D_{n_i,n_j}.

Fig. 4 shows the plot of the p-value versus the test statistic (1) with various n's ($n_i = n_j = n$). Given a gap $D_{n,n}$, a larger n leads to a smaller p-value. Thus even a small gap 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. This sensitivity obviously affects compression performance, as discussed in Section 5.

4. DESIGN OF IDEALEM

IDEALEM is targeted for the compression of time series data in **floating-point values**. Therefore, IDEALEM currently handles data in IEEE 754 double precision floating-point format which is 8 bytes long. The design of IDEALEM can be explained with Fig. 2. The main idea is to store only Θ_j that is distinct from previous Θ_j's, according to the similarity measure discussed in Section 3.

[3]This value is also called the significance level.

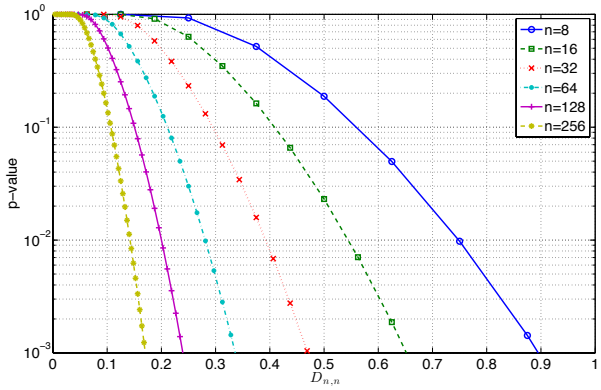


Figure 4: Effects of numbers of samples on the p-value and corresponding test statistics (distances) $D_{n,n}$, where the y-axis is drawn in log scale. Six different cases were synthetically generated, where each has n sample points for two random variables. As n grows, it becomes more difficult to exchange random variables due to lower p-values for a given distance.

4.1 Key Parameters

IDEALEM has three key parameters that affect its operation. First, **block length** n determines the number of samples in an individual sequence. An incoming time series is broken down into blocks with each of them having n elements. The block length has an effect on compression performance, due to the sensitivity of KS test shown in Fig. 4.

Second, **number of buffers** b controls how many Θ_j's are stored in memory for comparison, where each buffer holds one Θ_j. The number of buffers plays an important role in compression performance: more buffers in general promise higher compression ratios because there is a higher chance of finding a similar Θ_j stored in buffers when we encounter new X_i. However, increasing b also has drawbacks. We cannot simply store too many buffers at the same time in memory, especially if IDEALEM is to be used on resource-limited devices such as μPMU. Besides, a new data block is compared against each Θ_j to compute the p-value. Thus the more buffers we keep, the more KS tests are performed, which increases execution time.

Third, **threshold** α explained in Section 3 is the threshold for similarity 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.

4.2 Encoded Stream Structure

Fig. 5 illustrates an example of encoded stream structure by IDEALEM, where there are three buffers. The first data sequence in an input stream is outputted *as is*, along

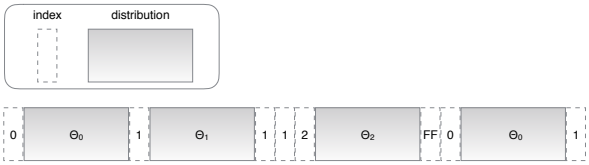


Figure 5: An example of encoded stream structure by IDEALEM, where $b = 3$. A dotted box represents an index in 1 byte; a solid box with gradation represents a distribution Θ_j whose size is $8n$ bytes. Seven sequences in total are shown here. Note that `0xFF` denotes a special marker for overwriting signal.

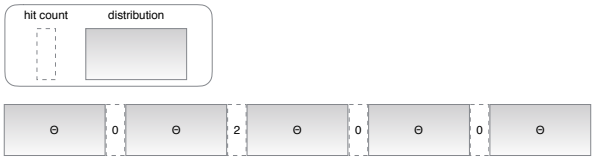


Figure 6: An example of encoded stream structure in the single buffer mode, which is the same scenario as in Fig. 5. A dotted box represents a hit count in 1 byte; a solid box with gradation represents a distribution Θ whose size is $8n$ bytes.

with the corresponding **index** which precedes the sequence.[4] Note that this data sequence is also stored in a buffer as the **distribution** Θ_0. Here, each buffer occupies $8n$ bytes.

Then, the second sequence is encountered and compared against the first sequence. In this example, it is not exchangeable, so the sequence 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 sequence is 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, where each index takes up 1 byte. The fourth sequence is exchangeable with Θ_1 as well.

The fifth sequence is not exchangeable with any of two stored distributions. So it is again written on the encoded stream as is with the corresponding index. And this sequence also occupies the last remaining buffer as the distribution Θ_2. The sixth sequence is compared with 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 currently discards the oldest buffer and replaces Θ_0 with this distribution, hence in first-in-first-out (FIFO) manner.

This overwriting should be signaled on the encoded stream so that the decoder can recognize it. To this end, IDEALEM uses a special marker `0xFF`, which automatically limits the number of buffers b to a maximum of 255. This marker is first outputted, and then the index and the sequence is written on the encoded stream. The seventh sequence is compared with from Θ_0 and finally exchangeable with Θ_1. (Comparison with Θ_2 is not necessary.) So only the index 1 is written on the encoded stream.

Single Buffer Case

When there is only a single buffer, i.e., $b = 1$, spending 1 byte on the index would waste a stream length, as an index would always have 0 for its value. Furthermore, frequent overwriting would lead to numerous `0xFF`'s each of which takes up 1 byte as well. Therefore, IDEALEM handles this single buffer case specially.

Fig. 6 shows an example of encoded stream structure in the single buffer mode, which is the same scenario presented in Fig. 5. However, the stream structure shown in Fig. 6 is different from the structure in Fig. 5: indices are now replaced by **hit counts** and the positions of the solid box and the dotted box are exchanged. Since there is only one buffer, a hit count records how many consecutive blocks are exchangeable with the previous distribution.[5]

In Fig. 6, the third and the fourth sequences are exchangeable with the previous distribution. Therefore, the hit count is 2 for these sequences. In contrast, the seventh sequence is no more exchangeable in this single buffer case. So this sequence is outputted as is.

Each hit count occupies 1 byte, allowing up to 255 repetitions to be represented by a single hit count.[6] However, while the hit count increases, the encoder cannot release it on the encoded stream, because the encoder has not encountered a dissimilar sequence yet. This phenomenon could happen especially when an input stream is monotonous, which indefinitely delays the decoding process in an online streaming environment. In order to prevent this, IDEALEM has a **maximum count** parameter that controls the latency of decoding. A smaller maximum count enables faster operation of IDEALEM in online streaming. On the other hand, it could also lengthen the encoded stream by using extra bytes for many repetitions.

4.3 Decoding

The encoded stream explained in Section 4.2 is in turn an input to the decoder of IDEALEM. The decoder reconstructs streaming data from learned probability distributions during the encoding process. This is accomplished with Θ_j's and corresponding indices j's in the encoded stream; and in the single buffer case, with Θ's and corresponding hit counts. In Fig. 5 and Fig. 6, data sequences that *initiate* new distributions are written on the stream as is. Therefore, even though we could generate new data sequences out of these distributions, it is better to retain original data sequences for these initiating sequences during reconstruction.

On the other hand, it is impossible to reconstruct the same data sequence as the original for an exchangeable case after the initiating sequence is reconstructed. Thus new data sequences should be generated from learned distributions for these cases, which means the orders of data sequences are no longer preserved. Since a stored distribution Θ_j or Θ is non-parametric, a random number generation from this distribution is equivalent to taking a random *uniform* sample from stored data samples. This sampling is done *without replacement* to avoid choosing any data sample more than once, which is fundamentally a random permutation. With the random permutation, we can also avoid any artificial

[4]Counting starts from 0.

[5]Thus, a hit count 0 denotes that there is no repetition after a specific data sequence.

[6]More than 255 repetitions can be represented with another hit count, and so on.

patterns generated during reconstruction.

4.4 Fundamental Limit on Achievable Compression Ratio

With the encoded stream structure discussed in Section 4.2, we inevitably set the maximum compression ratio we can achieve with a given block length n. We can show this theoretical upper bound by the following proposition.

PROPOSITION 1. *Given a block length n, the maximum achievable compression ratio of IDEALEM encoder with multiple buffers is $8 \cdot n$.*

PROOF. Without loss of generality, suppose we are compressing simple streaming data all of which can be represented with a single distribution Θ. In other words, there is a single source of distribution that governs the generation of data in this random process. Then ideally we can represent the entire data stream (except the beginning part that composes Θ) with many repetitions of the same index each of which takes up 1 byte.

Assuming there are i such repetitions, the original data size (in bytes) can be represented by $8n + 8ni$, where $8n$ is the size of the beginning part for Θ; $8ni$ is the size of the entire data stream excluding the beginning part. On the other hand, the compressed data size is represented by $1+8n+i$, where 1 and $8n$ are the sizes of the initial index and Θ; i is the size of repeating indices. If continuous streaming of data is assumed, the compression ratio is given by

$$\lim_{i\to\infty} \frac{8n + 8ni}{1 + 8n + i} = 8 \cdot n. \tag{3}$$

The compression ratio (3) is the maximum achievable compression ratio, because in a scenario where the exchangeability is not guaranteed, we would eventually have frequent overwriting of Θ, which no longer allows us to use the constant term $8n$ and $1 + 8n$ in (3).

A similar claim can be also made in the case where streaming data is composed of multiple *bounded* sources of distributions. Since this can be covered with Θ_j's and corresponding indices j's in the encoded stream, we can use a bounded number of constant terms for the compression ratio, which would again result in $8 \cdot n$. □

COROLLARY 2. *For the single buffer case, the maximum achievable compression ratio of IDEALEM encoder with a maximum count c is $8 \cdot cn$.*

PROOF. Again, suppose we are compressing simple streaming data all of which can be represented with a single distribution Θ. Then we can represent the entire data stream (except the beginning part that composes Θ) with many hit counts each of which can record up to c repetitions.

Assuming there are i such repetitions, the original data size can again be represented by $8n+8ni$. On the other hand, the compressed data size is represented by $8n+\lceil i/c \rceil$, where $8n$ is the size of Θ; $\lceil i/c \rceil$ is the size of hit counts. Assuming continuous streaming of data, the compression ratio is given by

$$\lim_{i\to\infty} \frac{8n + 8ni}{8n + \lceil i/c \rceil} = 8 \cdot cn. \tag{4}$$

The compression ratio (4) is the maximum compression ratio and can be achievable if and only if there is a bounded number of distribution changes (including no change) in streaming data. □

Proposition 1 indicates that a large n potentially increases compression ratio. However, a large n also increases the difficulty of passing the KS test due to the sensitivity discussed in Section 3. In practice, we cannot have ideal streaming data whose data sequences are nearly identical in the maximum distributional distance (1). Thus it is difficult to achieve the compression ratio of $8 \cdot n$ in real-world data, which is presented in Section 5.

Corollary 2 pushes this compression ratio even further: it can be as high as $2040 \cdot n$ with $c = 255$. This condition is obviously more difficult to achieve in reality, as we have to check the exchangeability with the previous distribution alone.

4.5 Text Input/Output Support

IDEALEM supports text as well as binary representation of floating-point data as input/output of the encoder and the decoder. Since IDEALEM is targeted for floating-point data compression, not for universal data as general compression schemes such as gzip handle [1], it is more efficient to represent floating-point data in binary representation: 8 bytes representation in binary typically needs more bytes in text representation.[7] ZFP also requires input/output of encoder and decoder to be entirely in binary representation.

On the other hand, the text representation has its own merits. First, it is common for an input stream to have text representation such as comma-separated values (CSV) format, which is a popular data exchange format across all platforms. Second, parsing of compressed data *without decoding* is made possible when the encoded stream explained in Section 4.2 is written in text representation, since the encoded stream structures shown in Fig. 5 and Fig. 6 are straightforward to interpret using text-processing software. To this end, IDEALEM supports CSV input for the encoder and CSV output for the decoder.[8] In addition, the encoded stream, which is the output of the encoder and also the input of the decoder, can be written in text representation.

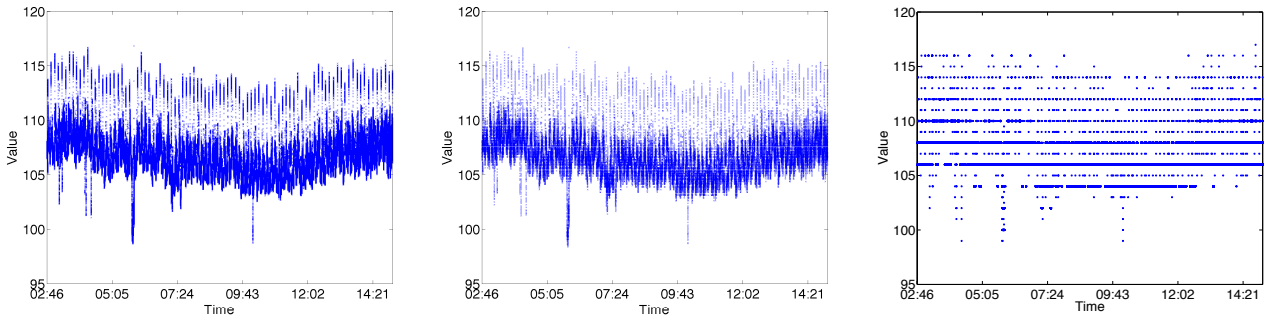
5. EVALUATION

The performance of IDEALEM can be assessed with various criteria. Some of common criteria are compression ratio and reconstruction quality. Since the reconstruction quality of IDEALEM cannot be directly assessed using a conventional measure such as MSE and SNR, as discussed in Section 2.1, we visually represent reconstruction results with various compression ratios, comparing them with original data. Here it is especially important not to lose *significant patterns* in the original data, which could be abnormal or singular such that they need attention of data analysts.

In terms of computational complexity, execution time and the memory usage of encoder and decoder are noteworthy, since IDEALEM is an online algorithm that handles streaming data. Furthermore, it can run on the sources of streaming data themselves, which could be resource-limited devices. We here employ a set of power grid monitoring data from μPMUs installed on-site at Lawrence Berkeley National Laboratory (LBNL) for experiments. The execution time

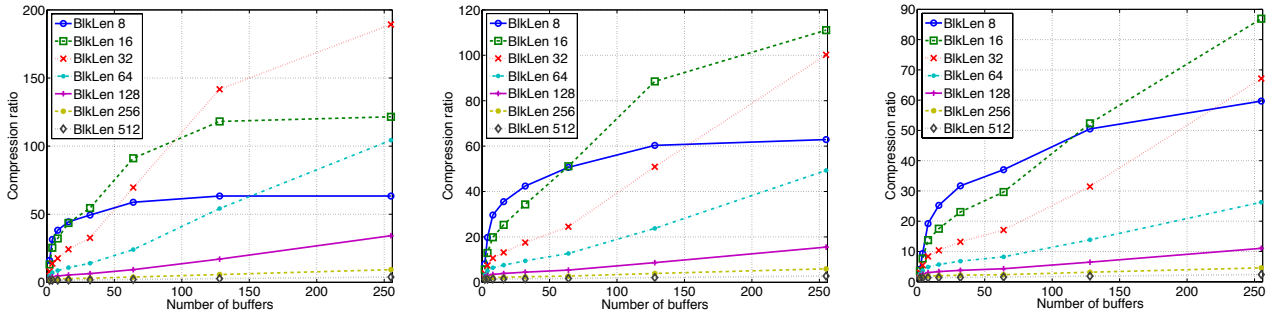
[7]For instance, representing a single floating-point value in CSV format would result in 14 bytes: 12 bytes for significant digits, 1 byte for the decimal-point character, and 1 byte for the newline.

[8]It is also possible to have binary output for the decoder despite the CSV input for the encoder.



(a) original (b) compression ratio 7.5 (c) compression ratio 9.14

Figure 7: Scatter plots of power grid electricity data from µPMU (5,300,001 samples) and compression results using ZFP. ZFP was invoked with options (b) `-a 1` and (c) `-a 8`. Reconstruction quality is visibly poor in (c).



(a) $\alpha = 0.01$ (b) $\alpha = 0.05$ (c) $\alpha = 0.1$

Figure 8: Compression ratios of IDEALEM with µPMU data shown in Fig. 7a. Maximum compression ratios are (a) 189.29, (b) 111.08, and (c) 86.91, respectively.

and the memory usage become crucial factors if data compression schemes have to run on such devices.

5.1 ZFP Floating-Point Compression

We first discuss the performance of the best-known floating-point compression method **ZFP** [15, 3]. ZFP is a lossy compression method. In numerous performance tests [15, 3], ZFP was found to outperform all other compression methods, particularly for scientific simulation data in 3D arrays. On these 3D arrays, ZFP was shown to achieve compression ratios of 100 without noticeable information loss in visualization.

However, these outstanding results are largely attributable to strong correlation among three dimensions. It has been studied that full utilization of correlation yields better compression ratios than partial utilization does [14, 13]. On 1D arrays, though ZFP performs significantly better than other methods, its compression ratios are lower than on 3D arrays, as shown in Fig. 7. In this regard, our objective is to develop a compression method that is as effective on 1D arrays as ZFP on 3D arrays.

Fig. 7a shows floating-point time series data captured from a µPMU deployed at LBNL, which are measurements of power grid electricity [23, 26]. In Fig. 7b and Fig. 7c, ZFP compresses the original data with different accuracies.[9] With a coarser accuracy tolerance, more information loss is tolerated, which leads to a higher compression ratio. This trend is clearly illustrated in Fig. 7. If we further increase the accuracy tolerance, eventually all reconstructed data values are set to zero and the corresponding compression ratio reaches 21.3.

5.2 Compression Ratios of IDEALEM

In Section 4.1, we noted the three key parameters of IDEALEM that affect performance. Using the same data that we used for ZFP in Fig. 7, we present compression ratios of IDEALEM with respect to various key parameter combinations.[10] Fig. 8 shows results for three different thresholds α's, where compression ratios are shown with varying block length n and number of buffers b for each α.

The maximum compression ratio in Fig. 8 is nearly 200. In particular, the maximum compression ratios of Fig. 8a and Fig. 8b are close to the fundamental limits $8 \cdot n$ shown in Proposition 1. Overall, we see that the compression ratios are high with a smaller α, which permits more data blocks to be declared as *similar* by the KS test. This threshold α is directly related to the reconstruction quality as well. Another clear trend we see is that the compression ratios

[9]The fixed-accuracy mode (option `-a`) was used [3], which specifies the maximum absolute difference between an uncompressed value and a reconstructed value (tolerance parameter).

[10]We use input data in binary representation, which reduces the size of original data in text representation.

increase as b increase, and the maximum compression ratios are always achieved with $b = 255$.

However, the relationship between the compression ratio and the block length n is somewhat unclear. The maximum compression ratios are achieved with $n = 16$ in Fig. 8b and Fig. 8c; while the maximum compression ratio is achieved with $n = 32$ in Fig. 8a. Although the compression ratio should increase with n in principle, it becomes also difficult to pass the KS test as n grows, which can be identified with Fig. 4 where a large n decreases the p-value for a given distance. Therefore, a large n could rather degrade the compression ratio.

It also seems that results with large n's could show higher compression ratios if we allowed $b > 255$, as their curves keep increasing up to $b = 255$. However, the current implementation of IDEALEM spends 1 byte for an index and it would need 2 bytes for the index to represent the corresponding distribution after 255 buffers, which in turn leads to a drop of the maximum achievable compression ratio from $8 \cdot n$ to $4 \cdot n$.

In Fig. 8, it is obvious that more buffers mean higher compression ratios. However, more buffers entail more memory usage at the same time. As explained in Section 4, each buffer of IDEALEM occupies $8n$ bytes. Therefore, the primary memory usage of IDEALEM, for both encoder and decoder, can be calculated as $8bn$ bytes.[11] For instance, with $n = 32$ and $b = 255$, the memory usage of IDEALEM is merely 65.28 KB, which is an acceptable amount for resource-limited devices.

We are also interested in the reconstruction quality of IDEALEM with these compression ratios. For this, reconstructed data are shown in Fig. 9, where we select the parameter combination of the maximum compression ratio and another combination of a compression ratio around 10. In Fig. 9, we can see the reconstruction quality of IDEALEM is excellent even in the case of the maximum compression ratio shown in Fig. 9c, where all the notable shapes of Fig. 9a are retained.

5.3 Preserving Patterns in Data

In Fig. 9, we have seen that IDEALEM preserves all the notable shapes (peaks and valleys) in the original data, even when a compression ratio is high. However, streaming data may contain significant patterns which may last for a very brief duration. Fig. 10 exhibits how IDEALEM captures these significant patterns in the original data shown in Fig. 11a with respect to different parameter combinations. Fig. 11a is a longer record of time series data shown in Fig. 7a, which now shows significant patterns, i.e., sudden increases and decreases in value, in the beginning and the ending parts.

Since IDEALEM has three tunable parameters, we adjust one parameter at a time while two others are fixed. However, the significant patterns that only persist for very short periods in Fig. 11a cause difficulties for capturing them. As a result, IDEALEM generally loses some of the significant patterns as a compression ratio increases. It is also interesting to note that in some cases IDEALEM introduces artifacts in reconstructed data and produces duplicate patterns. The interplay among the three parameters is so complex that there is no general principle (other than the compression ratio) that can explain the behavior of IDEALEM to capture significant patterns or to introduce artifacts.

In fact, this is attributable to the insensitivity of the KS test to detecting differences in the tails of distributions [9], which correspond to sudden increases and decreases of values for a very brief duration. For instance, the reconstructed data in Fig. 10d ($n = 8$, $b = 255$, and $\alpha = 0.05$) do not capture a downward spike in the dotted box which exists in the original data in Fig. 11a. Fig. 11b shows the magnified plot of the original data and the reconstructed data that correspond to the location of the downward spike, where we can see that during reconstruction IDEALEM found a similar distribution previously stored in the buffer that could pass the KS test. Two empirical (cumulative) distributions from these data samples are shown in Fig. 11c, where the test statistic (distance) is 0.625, but the p-value calculated from the standardized distance (2) is $0.087 > \alpha$. Therefore, we can deduce that if we had used a higher α or a large n, we could have captured this downward spike.

5.4 Performance with Large Data

We here use a large data set from two different μPMUs (A6BUS1 and BANK514), which contains about half a month records of power grid monitoring data. Each μPMU monitors three-phase measurements (1, 2, 3) of voltages (L) and currents (C), where the measurements, also known as **phasors**, are composed of the **magnitude** (MAG) and the **phase angle** (ANG) of sine waves in electricity [23, 26].[12] Thus a data set from each μPMU comprises 12 time series, and each time series occupies nearly 1 GB in binary representation.

Although direct comparison is rather unconvincing, we present the results of the compression ratio and the execution time of IDEALEM along with the results of gzip [1] and ZFP [3], for reference. In particular, **gzip** is a popular lossless coding scheme that is based on the combination of **LZ77** and **Huffman coding** [28, 20], which can handle general data types. On the other hand, ZFP is specifically designed to handle the floating-point data type and currently the most advanced lossy coding scheme [15], as discussed in Section 5.1.

5.4.1 Compression Ratio Comparison

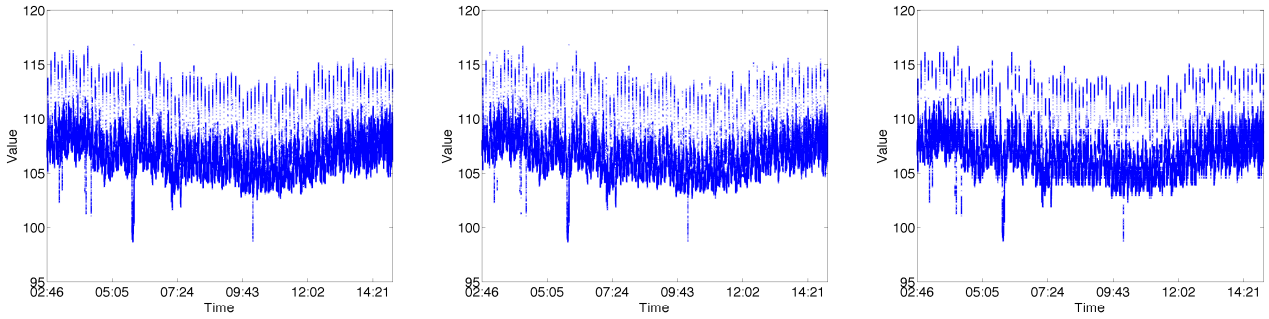
Table 1 shows the compression ratios of three compression schemes for μPMU data. As gzip provides nine different compression levels for a trade-off between the compression ratio and the execution time,[13] we selected three different levels among them. It is interesting to note that the best compression ratio is achieved with the compression level `-9` for magnitudes; but with the compression level `-5` for phase angles, which means a higher compression level does not always promise a better compression ratio.

We also show the compression ratio of ZFP in Table 1, where we selected four different tolerance parameters in the fixed-accuracy mode (option `-a`) [3]. In particular, we limited the tolerance parameter to 8, because we thought reconstruction quality was unacceptable beyond this point, as

[11] Currently, the encoder of IDEALEM stores a sorted version of a data sequence for each stored distribution, so as to accelerate the computation of KS test. Thus the memory usage is doubled ($16bn$ bytes) for the case of the encoder.

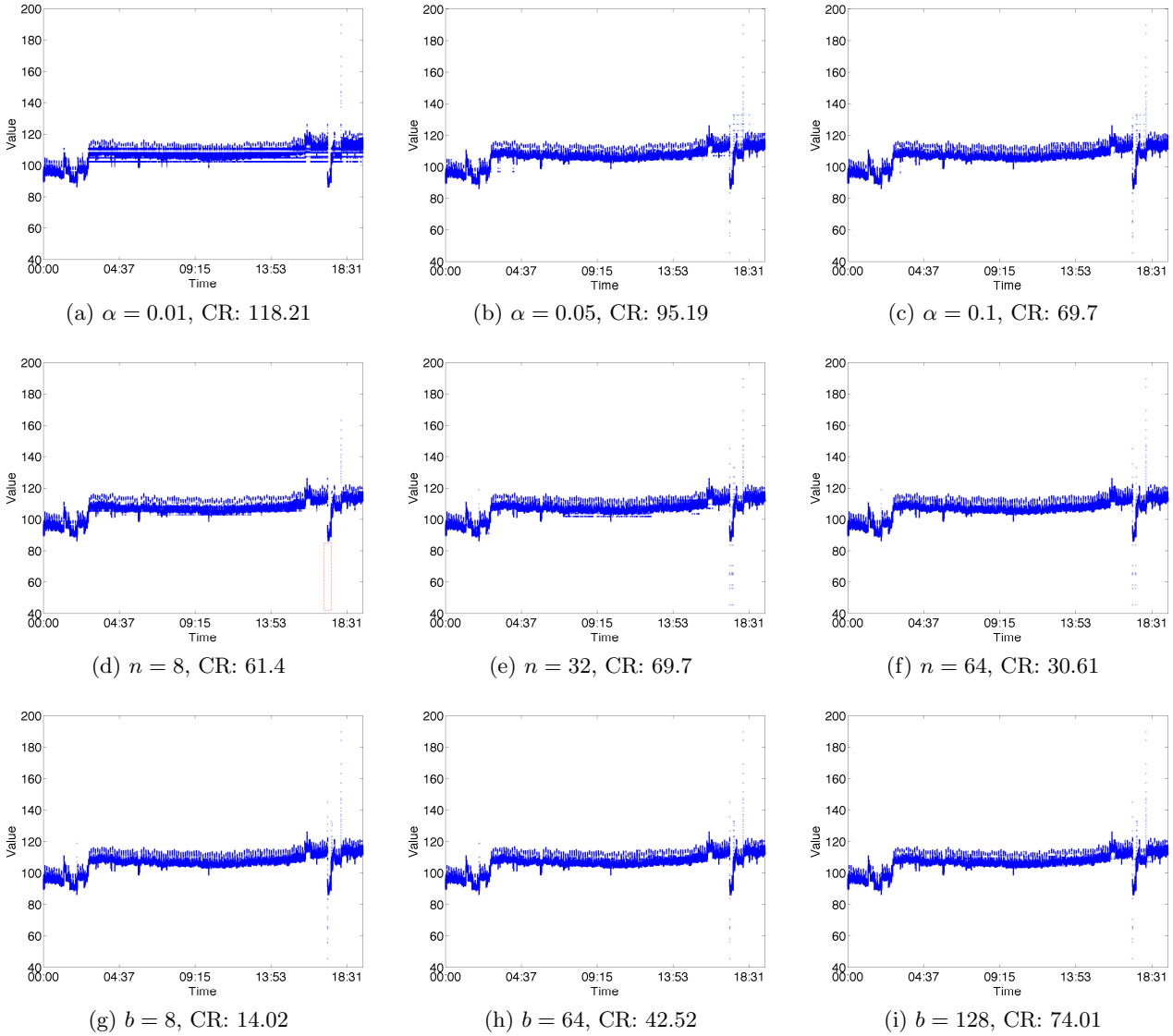
[12] For instance, 'A6BUS1C1MAG' denotes the magnitude of current on phase 1 measured by a μPMU named A6BUS1.

[13] The reconstruction quality does not play a role here, since gzip adopts the lossless compression.



(a) original (b) compression ratio 9.35 (c) compression ratio 189.29

Figure 9: Scatter plots of μPMU data (5,300,001 samples) in Fig. 7a and reconstructed data from IDEALEM compression. Results from IDEALEM were obtained with (b) $n = 8$, $b = 4$, and $\alpha = 0.1$; (c) $n = 32$, $b = 255$, and $\alpha = 0.01$.

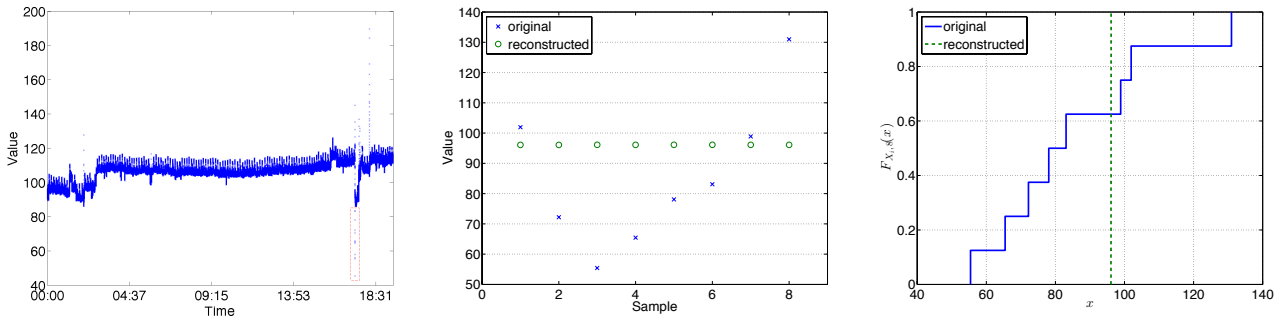


(a) $\alpha = 0.01$, CR: 118.21 (b) $\alpha = 0.05$, CR: 95.19 (c) $\alpha = 0.1$, CR: 69.7

(d) $n = 8$, CR: 61.4 (e) $n = 32$, CR: 69.7 (f) $n = 64$, CR: 30.61

(g) $b = 8$, CR: 14.02 (h) $b = 64$, CR: 42.52 (i) $b = 128$, CR: 74.01

Figure 10: Scatter plots of reconstructed data from IDEALEM compression of Fig. 11, showing the compression ratio (CR) when one parameter is adjusted, while two others are fixed: (a)-(c) $n = 16$ and $b = 255$; (d)-(f) $b = 255$ and $\alpha = 0.05$; (g)-(i) $n = 16$ and $\alpha = 0.05$.



(a) original (b) magnified (c) empirical distributions

Figure 11: Scatter plot of μPMU data (8,423,880 samples) that is a longer record of Fig. 7a and the magnified plot (8 samples) of a downward spike in the dotted box, shown together with the reconstructed result in Fig. 10d. Empirical distributions of original and reconstructed data samples are also shown.

Table 1: Compression Ratios

	(a) gzip			(b) ZFP				(c) IDEALEM				
Data	`-1`	`-5`	`-9`	`-a 1`	`-a 2`	`-a 4`	`-a 8`	**#1**	**#2**	**#3**	**#4**	**#5**
A6BUS1C1MAG	2.21	2.31	2.45	7.89	8.59	9.26	9.99	11.41	48.40	125.1	127.0	242.3
A6BUS1C2MAG	2.22	2.32	2.45	7.76	8.47	9.13	9.85	11.09	49.21	125.2	126.8	242.8
A6BUS1C3MAG	2.21	2.32	2.44	7.75	8.46	9.13	9.85	11.05	45.64	124.1	126.4	241.9
A6BUS1L1MAG	2.99	3.14	3.22	5.43	5.78	6.09	6.40	4.34	16.47	76.05	103.6	120.0
A6BUS1L2MAG	2.99	3.14	3.21	5.42	5.78	6.09	6.40	4.38	16.78	77.66	104.6	122.0
A6BUS1L3MAG	3.00	3.15	3.22	5.44	5.79	6.09	6.40	4.26	16.12	75.03	103.3	118.6
BANK514C1MAG	2.18	2.29	2.39	5.61	6.16	6.79	7.50	22.53	53.34	124.1	127.3	248.4
BANK514C2MAG	2.17	2.28	2.39	5.61	6.15	6.79	7.50	17.10	53.49	125.2	127.3	250.0
BANK514C3MAG	2.18	2.29	2.39	5.58	6.12	6.76	7.46	23.32	54.89	125.7	127.6	250.4
BANK514L1MAG	3.01	3.22	3.30	6.74	7.11	7.53	8.00	6.60	27.88	103.1	114.3	156.5
BANK514L2MAG	3.00	3.22	3.30	6.74	7.11	7.53	8.00	6.81	28.44	104.0	113.4	155.2
BANK514L3MAG	3.00	3.22	3.30	6.74	7.11	7.53	8.00	6.81	29.97	106.0	115.4	163.2
A6BUS1C1ANG	2.35	2.42	2.39	7.04	7.65	8.20	8.76	2.08	2.11	1.44	1.48	2.04
A6BUS1C2ANG	2.36	2.42	2.40	7.08	7.66	8.20	8.77	1.96	1.99	1.38	1.42	1.99
A6BUS1C3ANG	2.35	2.42	2.39	7.06	7.66	8.20	8.77	2.03	2.06	1.42	1.46	2.04
A6BUS1L1ANG	2.44	2.45	2.40	7.25	7.72	8.21	8.78	1.00	1.01	1.07	1.11	1.68
A6BUS1L2ANG	2.44	2.45	2.40	7.24	7.71	8.21	8.77	1.00	1.01	1.07	1.11	1.68
A6BUS1L3ANG	2.44	2.45	2.40	7.25	7.72	8.22	8.78	1.00	1.01	1.07	1.11	1.68
BANK514C1ANG	2.32	2.39	2.37	6.97	7.59	8.17	8.76	2.41	2.45	1.63	1.68	2.45
BANK514C2ANG	2.32	2.39	2.37	6.91	7.54	8.14	8.75	2.38	2.42	1.62	1.67	2.60
BANK514C3ANG	2.31	2.37	2.36	6.89	7.53	8.14	8.75	2.62	2.67	1.79	1.90	2.83
BANK514L1ANG	2.44	2.45	2.41	7.24	7.72	8.21	8.78	1.03	1.04	1.07	1.12	1.69
BANK514L2ANG	2.44	2.45	2.41	7.24	7.72	8.21	8.78	1.03	1.04	1.07	1.12	1.69
BANK514L3ANG	2.44	2.45	2.41	7.25	7.72	8.21	8.78	1.03	1.04	1.08	1.12	1.69

shown in Fig. 7. The compression ratios of ZFP shown in Table 1 are clearly higher than the results of gzip; but they do not exceed 10.

Table 1 also shows the compression ratio of IDEALEM, where we selected five parameter combinations: (#1) $n = 8$ and $b = 4$, (#2) $n = 8$ and $b = 16$, (#3) $n = 16$ and $b = 64$, (#4) $n = 16$ and $b = 128$, and (#5) $n = 32$ and $b = 255$, all with $\alpha = 0.01$.

We can see that compression ratios reach close to the maximum achievable compression ratios for magnitude measurements (MAG) when the numbers of buffers are large enough. However, for phase angle measurements (ANG), IDEALEM does not show competitive results. This is because the phase angle data are mainly composed of smoothly increasing parts, as shown in Fig. 12. Data of this kind are not easily compressible with the current implementation of IDEALEM that uses the KS test and the limited number of buffers.

5.4.2 *Execution Time Comparison*

We also measured the execution time of each compression scheme using the same parameters as in Section 5.4.1. Experiments were conducted on a server equipped with Intel Xeon X3450 (2.66 GHz) CPU, 8 GB RAM, and 256 GB SSD, which runs Ubuntu 10.04.4 LTS (Linux kernel 2.6.32-57-server). Average execution time of both encoder and decoder was measured for each scheme. Since the three-phase measurements all showed similar results as the compression

Table 2: Execution Time (s) for Three-Phase Group

		(a) gzip			(b) ZFP				(c) IDEALEM				
	Data	`-1`	`-5`	`-9`	`-a 1`	`-a 2`	`-a 4`	`-a 8`	**#1**	**#2**	**#3**	**#4**	**#5**
Encoding	A6BUS1CMAG	27.75	52.93	1001	12.03	11.42	10.68	10.73	14.37	20.12	36.75	69.64	80.31
	A6BUS1LMAG	22.59	40.01	591.6	13.40	13.57	12.71	12.54	17.99	22.97	36.73	64.66	93.81
	BANK514CMAG	28.49	53.12	855.5	13.52	13.17	12.76	11.98	14.18	19.78	35.48	56.09	79.12
	BANK514LMAG	23.42	40.58	615.5	11.91	11.82	11.44	11.42	16.40	21.74	38.69	67.47	93.94
	A6BUS1CANG	25.15	46.70	889.0	11.84	11.69	11.32	10.85	19.72	34.44	75.30	130.6	181.9
	A6BUS1LANG	27.28	48.91	1045	12.40	11.81	11.53	10.96	26.78	47.40	81.42	145.3	170.7
	BANK514CANG	29.25	47.95	906.2	12.32	11.16	10.79	11.00	20.13	32.15	70.51	123.0	149.0
	BANK514LANG	26.84	48.83	1038	12.02	11.21	11.41	11.10	27.03	46.85	81.07	148.4	169.7
Decoding	A6BUS1CMAG	19.25	19.27	19.77	12.78	13.95	17.50	12.52	25.09	22.32	18.91	20.23	17.13
	A6BUS1LMAG	19.85	19.73	22.10	14.98	15.58	17.76	13.91	21.58	23.41	19.63	19.33	19.16
	BANK514CMAG	22.12	21.85	19.79	15.95	17.29	15.91	13.59	22.60	20.50	19.56	18.74	17.22
	BANK514LMAG	24.11	19.55	18.42	14.04	16.44	13.54	12.64	23.09	22.83	19.51	19.58	18.09
	A6BUS1CANG	16.23	18.79	21.84	13.83	13.91	15.88	11.53	21.21	18.42	16.19	17.35	15.76
	A6BUS1LANG	19.01	18.90	22.93	13.50	13.42	16.36	11.95	18.45	17.58	16.65	16.87	18.45
	BANK514CANG	20.45	22.29	18.73	13.98	13.91	13.29	11.70	21.95	21.32	18.93	19.26	18.25
	BANK514LANG	20.58	20.80	17.67	13.76	13.81	11.69	11.08	16.68	18.36	15.81	15.22	17.52

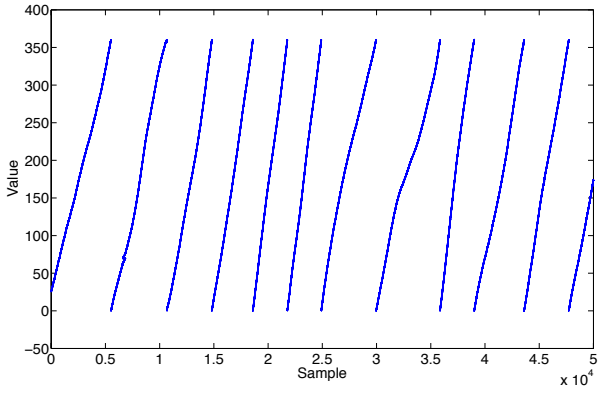


Figure 12: Scatter plot of first 50,000 samples of A6BUS1C1ANG. Other phase angle measurements show similar shapes.

ratios in Table 1 showed similar results for all three phases, we group them together and present *average* time measurements. For instance, 'C1,' 'C2,' and 'C3' are grouped into 'C.'

Table 2 shows the execution time of gzip, ZFP, and IDEALEM, measured in seconds. Here the encoding time is shown at the top and the decoding time is shown at the bottom. In Table 2, the encoding of gzip with the compression level `-9` takes very long time compared with other two levels, whereas the decoding time is similar for all three levels. This is because gzip is based on the dictionary-based coding and a higher compression level entails longer dictionary searching.

A similar tendency for IDEALEM can be identified in Table 2, where large numbers of buffers for IDEALEM demand longer execution time. But the decoding time does not depend on the number of buffers, as in the case of gzip. The encoding time of IDEALEM is faster than gzip in the worst case. It should be noted that for IDEALEM, the results of phase angles (ANG) show longer execution time than those of magnitudes (MAG). This is directly related to the fact that the current implementation of IDEALEM cannot efficiently compress the data of phase angle measurements: IDEALEM searches its buffers to find a similar distribution every time it encounters a new data sequence, but its attempts generally fail after a full search. On the other hand, encoding the magnitude measurements is faster than encoding the phase angle measurements especially with large numbers of buffers, since IDEALEM easily finds a similar distribution stored in the buffers and this search finishes quickly.

Results of ZFP shown in Table 2 are the fastest among three schemes. Note that ZFP is fast for both encoding and decoding. And encoding is in general faster than decoding. Furthermore, the results of ZFP do not vary across four different parameters, because ZFP is not based on dictionary or buffer searching.

6. CONCLUSIONS

In this work, we report our design and implementation of a novel data reduction technique named IDEALEM based on statistical similarity. The key feature that distinguishes this method is that it permits data blocks to be compared without regard to the relative positions of the values in incoming data. This method breaks an incoming data stream into blocks of the fixed size and represents *similar* blocks with a one that appears earlier in the data. Instead of measuring the similarity of two blocks based on traditional measures such as the Euclidian distance, we use a statistical tool known as Kolmogorov-Smirnov test (KS test). Through a careful design of the compression algorithm, we are able to keep distinctive features in a dataset, while significantly reducing the size needed to keep *common* and *uninteresting* data records. On a set of data from power grid, IDEALEM can compress many of the variables by more than 100-fold, while capturing important features in the data such as voltage sags and current spikes at the same time. This clearly demonstrates the usefulness of our new compression method.

An immediate plan for additional work on IDEALEM is to quantify how the distinctive features are kept. We want to exercise the property that allows analysis without decompression to perform some advanced feature detection techniques. As one might expect, our tests also show that

IDEALEM is not effective in some cases. In the test data set, the variables that smoothly vary over time are hard for IDEALEM to compress. It would also be interesting to modify the compression scheme to adopt features of known compression methods to improve the effectiveness of IDEALEM on such smoothly varying data records. Clearly KS test is only one possible similarity measure IDEALEM could use, and we can explore alternative similarity measures.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge helpful discussions with Emma Stewart, Sean Peisert, Chuck McParland, Reinhard Gentz, Mahdi Jamei, Ciaran Roberts, and Sebastian Ainslie. This work was supported by the Office of Advanced Scientific Computing Research, 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).

8. REFERENCES

[1] The gzip home page. http://www.gzip.org.

[2] IDEALEM. http://datagrid.lbl.gov/idealem.

[3] zfp & fpzip: floating point compression. http://computation.llnl.gov/projects/floating-point-compression.

[4] M. Burtscher and P. Ratanaworabhan. FPC: a high-speed compressor for double-precision floating-point data. *IEEE Trans. Comput.*, 58(1):18–31, January 2009.

[5] M. D. Cao, T. I. Dix, L. Allison, and C. Mears. A simple statistical algorithm for biological sequence compression. In *Proc. Data Compression Conf. (DCC '07)*, pages 43–52, 2007.

[6] X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *Proc. Int'l Conf. Comput. Mol. Biol. (RECOMB '00)*, page 107, 2000.

[7] X. Chen, M. Li, B. Ma, and J. Tromp. DNACompress: fast and effective DNA sequence compression. *Bioinform.*, 18(12):1696–1698, December 2002.

[8] J. Choi, K. Hu, and A. Sim. Relational dynamic Bayesian networks with locally exchangeable measures. Technical Report LBNL-6341E, Lawrence Berkeley National Laboratory, July 2013.

[9] S. Engmann and D. Cousineau. Comparing distributions: the two-sample Anderson-Darling test as an alternative to the Kolmogorov-Smirnoff test. *J. Appl. Quant. Methods*, 6(3):1–17, September 2011.

[10] J. Iverson, C. Kamath, and G. Karypis. Fast and effective lossy compression algorithms for scientific datasets. In *Proc. Int'l Conf. Parallel Process. (Euro-Par '12)*, pages 843–856, 2012.

[11] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Compressing the incompressible with ISABELA: in-situ reduction of spatio-temporal data. In *Proc. Int'l Conf. Parallel Process. (Euro-Par '11)*, pages 366–379, 2011.

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

[13] D. Lee, J. Choi, and H. Shin. A scalable and flexible repository for big sensor data. *IEEE Sensors J.*, 15(12):7284–7294, December 2015.

[14] D. Lee, J. Ryu, and H. Shin. Scalable management of storage for massive quality-adjustable sensor data. *Computing*, 97(8):769–793, August 2015.

[15] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Trans. Vis. Comput. Graphics*, 20(12):2674–2683, December 2014.

[16] F. J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *J. Am. Stat. Assoc.*, 46(253):68–78, March 1951.

[17] C. Quinsac, A. Basarab, J.-M. Girault, and D. Kouamé. Compressed sensing of ultrasound images: sampling of spatial and frequency domains. In *Proc. Int'l Workshop Signal Process. Syst. (SiPS '10)*, pages 231–236, 2010.

[18] I. E. Richardson. *The H.264 Advanced Video Compression Standard*. John Wiley and Sons, second edition, 2010.

[19] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. Approximate storage in solid-state memories. In *Proc. Int'l Symp. Microarchitecture (MICRO '46)*, pages 25–36, 2013.

[20] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, fourth edition, 2012.

[21] J. Seabra and J. Sanches. Modeling log-compressed ultrasound images for radio frequency signal recovery. In *Proc. Int'l Conf. Eng. Med. Biol. Soc. (EMBC '08)*, pages 426–429, 2008.

[22] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki. Practical data compression in wireless sensor networks: a survey. *J. Netw. Comput. Appl.*, 35(1):37–59, January 2012.

[23] E. M. Stewart, S. Kiliccote, C. McParland, C. Roberts, R. Arghandeh, and A. von Meier. Using micro-synchrophasor data for advanced distribution grid planning and operations analysis. Technical Report LBNL-6866E, Lawrence Berkeley National Laboratory, July 2014.

[24] R. L. Wasserstein and N. A. Lazar. The ASA's statement on p-values: context, process, and purpose. *Am. Stat.*, 2016. doi: 10.1080/00031305.2016.1154108.

[25] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, June 1984.

[26] M. H. Wen, R. Arghandeh, A. von Meier, K. Poolla, and V. O. Li. Phase identification in distribution networks with micro-synchrophasors. In *Proc. Power & Energy Soc. Gen. Meet. (PES-GM '15)*, pages 1–5, 2015.

[27] J. Yu, S. Ongarello, R. Fiedler, X. Chen, G. Toffolo, C. Cobelli, and Z. Trajanoski. Ovarian cancer identification based on dimensionality reduction for high-throughput mass spectrometry data. *Bioinform.*, 21(10):2200–2209, May 2005.

[28] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, 23(3):337–343, May 1977.