ISAAC: Intelligent Synchrophasor Data Real-Time Compression Framework for WAMS

Wenyu Ren*, Timothy Yardley* and Klara Nahrstedt*
* University of Illinois Urbana-Champaign, Urbana, Illinois, USA
Email: {wren3, yardley, klara}@illinois.edu

***Abstract*—Wide-Area Monitoring Systems (WAMS) have been widely accepted to provide real-time monitoring, protection, and control of power systems. The huge and rapidly increasing data volume in WAMS imposes a heavy burden on the communication and storage systems and could become the bottleneck for many real-time smart grid applications. This paper presents a framework for intelligent lossy compression in a real-time manner for synchrophasor data in WAMS. The proposed method is capable of achieving good compression ratios without introducing impractical delays and sacrificing much accuracy. Because disturbance data has much stricter delay and fidelity requirements, an early disturbance detection technique is introduced to identify disturbance data and handle it differently. The performance of the method is demonstrated by experiment results on real synchrophasor data collected from multiple substations.**

I. Introduction

Recently, Wide-Area Monitoring Systems (WAMS) are becoming more and more widely accepted because of their ability to monitor, protect and control power systems over large areas in real time. WAMS's capability to support real-time decision-making applications is based on the high reporting rates and precise time synchronization provided by the new data acquisition technology of phasor measurement. Allowed by the emerging and development of Phasor Measurement Units (PMUs), frequency, current, and voltage can be measured at a rate of 30 Hz or higher, much faster than in the conventional Supervisory Control and Data Acquisition (SCADA) systems, where samples are taken only every few seconds. The generated measurements are called synchrophasors, namely synchronized phasors, since they contain both magnitudes and phase angles, and are precisely time-synchronized by the Global Positioning System (GPS) technology. The synchrophasors generated by PMUs over wide-area power systems can serve as snapshots of the system status and can be further utilized for real-time wide-area monitoring, protection, and control. For instance, PMU data can aid or gradually replace the state estimation process which is a key function in supervisory control of power grids, since the accurate status information of the grid can be directly acquired from the real-time synchrophasors.

Since PMUs have very high sampling rates and usually multiple data channels, the volume of measurements collected is huge. 100 PMUs of 20 measurements, each running at 30 Hz, will generate over 50 GB of data per day [1]. 3500 data channels of 34 PMUs running at 100 Hz in Southwest China produce over 120 GB of data per day [2]. As the scale of power systems and WAMS grows, the number of PMUs is also growing rapidly to provide finer-grained global state of the more volatile power systems. For instance, the deployment of PMUs in North America has largely increased from only 200 research-grade PMUs in 2009 to almost 1700 production-grade PMUs in 2014 [3]. Besides the number of PMUs, the number of measurements at each PMU is also growing as more grid parameters get included for monitoring, from several synchrophasors to tens of them. Due to the higher sampling rate of modern PMUs, the increase in the number of PMUs and measurements per PMU, we can surely expect a multi-fold expansion in the already large volumes of synchrophasor data in WAMS.

The synchrophasor data generated by PMUs need to be transmitted in the underlying communication systems in real time and stored in control centers for archive purpose (historian). The huge and ever-increasing volume of synchrophasor data introduces tremendous storage and bandwidth capacity requirement for WAMS. Therefore, it is necessary to use data compression techniques to lighten the heavy burden on the storage and communication systems. Many works of power data compression focus on the compression for storage or offline bulk transmission [4]–[10]. However, we argue that online data compression for real-time transmission should be addressed as much, if not more, than offline compression. If not handled carefully, the huge data volume in the communication system could result in frequent and severe congestion. The WAMS applications could suffer a lot from the extremely long delays or high packet loss rates that follow the congestion.

The two main challenges for designing real-time compression frameworks in WAMS are as follows:

- *Delay*: The data should be compressed in a real-time manner. In other words, the delay matters. Most existing compression techniques use large sampling windows to achieve better compression performance, which is a luxury that real-time compression techniques cannot afford.
- *Disturbance*: The delay and accuracy requirements of data during disturbance[1] are different from those in normal status. Thus it is necessary to treat disturbance data and normal data differently in compression, which requires the early detection of disturbances to be incorporated into the compression framework.

[1]Disturbances in this paper means transients in measurements that may be caused by misoperations, faults, topology changes, load and source dynamics.

In this paper, we propose an Intelligent Synchrophasor dAta reAl-time Compression framework for WAMS named ISAAC. Combining the Principal Component Analysis (PCA) and Discrete Cosine Transform (DCT), ISAAC has the capability of largely improving the efficiency of communication and storage systems via data compression while maintaining strong data fidelity. A disturbance detector allows ISAAC to differentiate normal and disturbance data and process them in different ways. Two core techniques, which are the transformation matrix reuse in PCA and the self-adapt principal component number selection, allows ISAAC to achieve a good compression ratio (CR) without introducing an impractical delay.

The remainder of this paper is organized as follows: Section II reviews the related work. Section III provides some background and explains the two compression algorithms used by our approach. Section IV describes the design of ISAAC. Section V shows the performance evaluation of time, CR, and accuracy and Section VI concludes the paper.

II. Related Work

Data compression techniques can be categorized into lossless compression and lossy compression in general. There are many works [4]–[6] focusing on lossless compression of power quality data, smart meter data, and PMU data. Our paper, on the contrary, focuses on lossy compression, since lossy compression has a potential to achieve a better CR compared with lossless compression and it is acceptable as long as parameters of compression algorithms are selected carefully to maintain information loss within required bound.

Among all the works using lossy compression techniques for PMU data, most of them [7]–[10] perform compression for storage or offline bulk transmission purpose, in which cases the delay of the compressed data is not a concern. Therefore, they are able to use a sampling window with the length of 10 seconds or longer to gather enough data and compress the entire data block to achieve better CRs. For sampling window based approaches, the earliest set of measurements in each window needs to wait for the entire window to be filled before it can be compressed and sent. Hence, a window size of 10 seconds means a delay of more than 10 seconds for the earliest set of measurements in each data block, which is far beyond the delay constraints of most of the real-time WAMS applications [11], [12]. As a result, compression techniques for PMU data storage or offline transmission are not directly applicable to real-time PMU data compression purpose.

A semantics-aware real-time data transmission reduction method is proposed in [13]. Grid applications consuming PMU data are modelled as continuous threshold queries and relevant data are delivered only when the threshold condition is broken. The drawback of this approach is not all applications can be modeled in their way and absence of detailed data during normal status will definitely impair the system's ability to conduct monitoring and detailed analysis of the data.

Another real-time data compression technique is presented in [2], combining exception compression with swing door trending compression. Each sequence of measurements of each PMU is compressed separately by only keeping the data with essential and effective information. The problem with this approach is that the correlation between multiple sequences of measurements in multiple PMUs is not utilized. And since this technique needs to be installed on each PMU, the deployment cost is high considering the large quantity of PMUs.

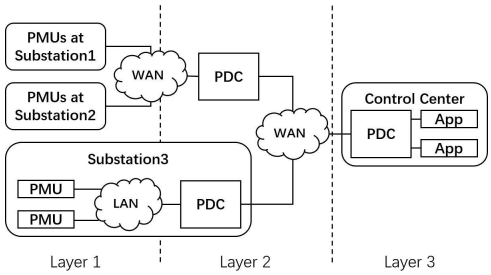


Fig. 1. WAMS architecture

III. Background

In this section, we first introduce the generic WAMS architecture and the phasor data concentration. Then we introduce two compression techniques we use.

A. WAMS Architecture

A generic architecture of WAMS consists of four main components: (1) PMU, (2) Phasor Data Concentrator (PDC), (3) WAMS Applications, (4) underlying Communication Network to connect the above three [14]. A simplified, multi-layered architecture of WAMS is shown in Figure 1[2]. In Layer 1, the PMUs are installed in power system substations to measure the connected bus bars or power lines. The synchrophasor measurements from the PMUs are then transmitted via Local Area Networks (LAN) or Wide Area Networks (WAN) to Layer 2, where they are concentrated and sorted by the PDCs. After that, the time-aligned measurements are forwarded via WAN to the control center in Layer 3 and usually further concentrated by the PDC there before finally consumed by the WAMS applications.

Our real-time compression technique, ISAAC, resides in the PDCs in Layer 2. We further divide the architecture into two scenarios: (1) LAN-WAN scenario, where the Layer-2 PDC locates in the substation and connects to the PMUs via LAN; (2) WAN-WAN scenario, where the Layer-2 PDC collects measurements from multiple substations and connects to the PMUs via WAN.

B. Phasor Data Concentration

As the key component of the WAMS architecture, PDC's core function is to combine synchrophasor measurements from more than one PMUs into a single time-synchronized data stream for further processing [16]. More specifically, it collects and sorts measurements from all connected PMUs according to their timestamps. Measurements with the same timestamp are encapsulated into one packet and forwarded to the control center. Since not all measurements arrive at the same time,

[2]There are scenarios in which some of the components are mobile [15]. In this work we only consider the most common static case.

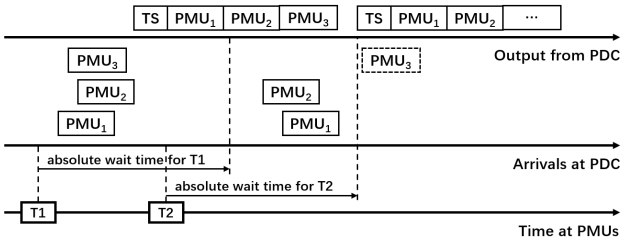


Fig. 2. Example of phasor data concentration with time alignment to absolute time [17]

PDC needs to wait and eventually timeout to mitigate the delay. The entire process is called time alignment and can be categorized into absolute-time-based and relative-time-based time alignment [16]. For real-time monitoring, protection and control applications, time alignment to absolute time reference is preferred since it can provide better delay guarantees. Therefore, we will only consider and introduce time alignment to absolute time reference here.

An example of phasor data concentration with time alignment to absolute time is shown in Figure 2. At a certain rate (e.g., 60Hz), measurements with timestamps are produced by synchronized PMUs. The countdown to the timeout starts at the time specified by each timestamp. As it is shown in the figure, there are two potential scenarios: (1) All measurements with timestamp T1 arrive before the timeout and the complete data set is forwarded before the timeout; (2) Not all measurement with timestamp T2 arrive before the timeout and the incomplete data set without measurement from PMU_3 is forwarded at the end of the timeout. Note that the processing time of PDC is omitted in this example for simplicity reason.

Phasor data concentration is only the core function of the PDC. Since the PDC is an edge device that phasor measurements reach before they are further forwarded to the control center, more and more functions are being placed at the PDC including data handling, processing, and storage [16]. And it is also a perfect location to deploy our intelligent synchrophasor data real-time compression framework.

C. Principal Component Analysis

As a commonly used linear dimensionality reduction technique, PCA [18] is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components (PCs).

Consider an $m \times n$ data matrix $\mathbf{X}$ containing n set of samples from m PMUs[3] expressed as

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}. \quad (1)$$

x_{ij} represents the jth measurement from the ith PMU. The PCA method starts by calculating the covariance matrix as $\mathbf{C} = \mathbf{X}\mathbf{X}^T \in \mathbb{R}^{m \times m}$. Since $\mathbf{C}$ is a square symmetric matrix, it can be orthogonally (orthonormally) diagonalized as

$$\mathbf{C} = \mathbf{E}\mathbf{D}\mathbf{E}^T, \quad (2)$$

where $\mathbf{E}$ is an $m \times m$ orthonormal matrix whose columns are the eigenvectors of $\mathbf{C}$, namely PCs, and $\mathbf{D}$ is an $m \times m$ diagonal matrix with the corresponding eigenvalues as the diagonal entries. The eigenvalues can also represent the variance explained by each PC and are sorted in descending order. PCA performs dimensionality reduction by preserving only a subset of PCs which explain most of the variance of the original data. Assume the first r out of the m PCs are selected. Let $\mathbf{E}(r)$ represent the left most r columns of $\mathbf{E}$. The transformation matrix is selected as $\mathbf{P} = \mathbf{E}(r)^T \in \mathbb{R}^{r \times m}$ and the dimension reduced matrix is expressed as $\mathbf{Y} = \mathbf{P}\mathbf{X} \in \mathbb{R}^{r \times n}$. Let λ_i represent the eigenvalue (variance), associated with the ith PC, the total variance explained by $\mathbf{Y}$ is defined as

$$\Gamma(r) = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{m} \lambda_i}. \quad (3)$$

Usually, we want to select r s.t. $\Gamma(r)$ is greater or equal to a variance threshold γ.

Although the core idea of dimensionality reduction by PCA is the same as above, there are various implementations of the method. In this paper, we use two implementations: The first one (we name it PCA-D) utilizes the *sklearn.decomposition.PCA* [19] implementation and cannot work with sparse matrix; the second one (we name it PCA-S) utilizes the *sklearn.decomposition.TruncatedSVD* [19] implementation and can preserve the sparsity of the matrix.

D. Discrete Cosine Transform

DCT [20] transforms a sequence of data points of length n to a domain of n cosine basis functions. The transformed data are the coefficients of the basis functions. Some of the coefficients have small magnitudes and thus can be discarded without sacrificing much accuracy. We use DCT implemented in *scipy.fftpack.dct* [21] to further compress each row of the dimension reduced matrix $\mathbf{Y}$ in the previous section. Similar to the previous section, let c_i represent the coefficient with the ith largest magnitude. If l out of all the n coefficients are kept, the cumulative energy kept can be expressed as:

$$E(l) = \frac{\sum_{i=1}^{l} c_i}{\sum_{i=1}^{n} c_i} \quad (4)$$

Similar to $\Gamma(r)$, $E(l)$ is used to select proper l value. In this work, we select the smallest l that satisfies $E(l) > 0.8$.

IV. Methodology

In this section, we present an overview of the design of ISAAC. The workflow of ISAAC is described in Figure 3. There are four main components utilized in ISAAC: (1) *Buffer* which buffers all the measurements have not been sent, (2) *Time Alignment Component* which time-aligns the most recent set of measurements with an absolute timeout, (3) *Disturbance Detector* which decides whether there are disturbances happening, (4) *Compressor* which compresses

[3]There are usually multiple measurements from each PMU. So the actual number of rows is larger than the number of PMUs. But we assume m rows here for simplicity.

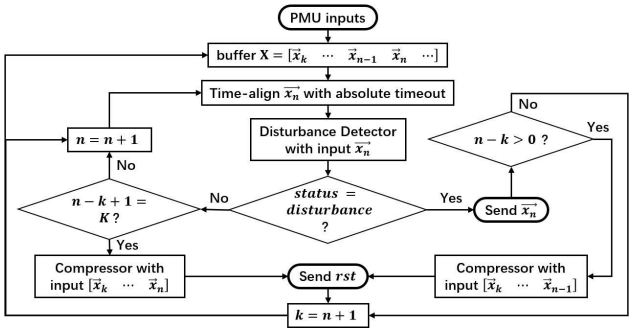


Fig. 3. Workflow of ISAAC

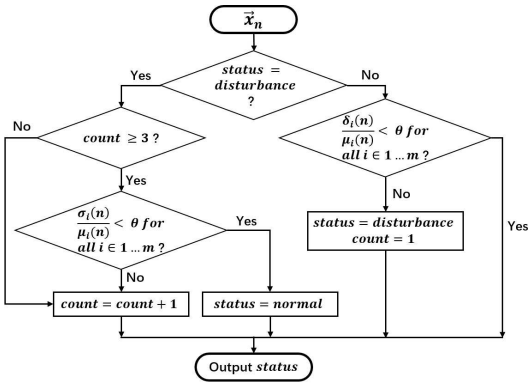


Fig. 4. Workflow of the disturbance detector

the input matrix based on PCA and DCT. Among the four components, the buffer and the time alignment component are the built-in functions of PDCs. The disturbance detector and the compressor, on the other hand, are the core parts of ISAAC and are the focus of this section.

In Figure 3, $\vec{x}_i$ is a column vector containing all the received measurements from all PMUs corresponding to time index i. The buffer serves as the sampling window and buffers all the received stream measurements with time index k or larger. $\vec{x}_k$ represents the earliest unsent data set and $\vec{x}_n$ represents the earliest unprocessed data set. Periodically, the time alignment component aligns the measurements in the buffer based on their timestamps and forwards $\vec{x}_n$ to the disturbance detector[4]. The disturbance detector processes $\vec{x}_n$ and outputs the estimated status (normal or disturbance) of the system. In normal status, ISAAC further compares the current window size from k to n with the maximum sampling window size K. If $n-k+1 = K$, all measurements buffered from k to n, i.e. matrix $[\vec{x}_k, \cdots, \vec{x}_n]$, are forwarded to the compressor and compressed. Otherwise, ISAAC waits for more measurements (longer sampling window) to compress. In disturbance status, $\vec{x}_n$ is directly sent without compression. If there are buffered measurements besides $\vec{x}_n$, i.e. matrix $[\vec{x}_k, \cdots, \vec{x}_{n-1}]$ is not empty, then they are forwarded to the compressor for compression. If the compressor is called, no matter in which status, the compressed results are sent. k is updated to $n+1$ as long as something is sent, which means the measurements in the buffer before but not including time index $n+1$ are cleared. And n is always increased by one after each period.

There are five kinds of measurements we consider in this work: *Frequency* (f), *Voltage Magnitude* (vm), *Voltage Angle* (va), *Current Magnitude* (im), and *Current Angle* (ia). The following two subsections describe the disturbance detector and the compressor in more detail.

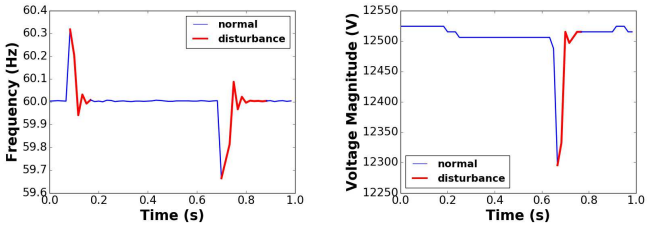
A. Disturbance Detector

The main purpose of the disturbance detector is to differentiate the normal status and the disturbance status of the system. Intuitively, a higher delay and lower accuracy are tolerable in normal status, whereas measurements should be collected as soon as possible during disturbances and higher fidelity is necessary to preserve the important information in the measurements. Since the requirements for delay and accuracy are different in the two status, it is worth differentiating them and handle them in different ways. Our disturbance detector implements a modified version of a relatively simple statistical change detection algorithm [9] for computation power and delay consideration.

According to PRC-002-2 by NERC [22], the recommended disturbance triggering criteria include: (1) frequency < 59.75 Hz or > 61 Hz, (2) rate of change of frequency < 0.03125 Hz/s or > 0.125 Hz/s, 3) Undervoltage trigger set no lower than 85% of the normal operating voltage for a duration of 5 seconds. However, as it is pointed out in [9], [23], these values are too conservative to detect all potential disturbances and preserve all critical information. Similar to [23], we select stricter triggering criteria as percentage deviation of $\theta_{vm} = 1\%$ for voltage and $\theta_f = 0.1\%$ for frequency. These values could be tuned if necessary without affecting the algorithm itself.

Since the disturbance triggering criteria are based on frequency and voltage magnitude only, the disturbance detector only processes those two kinds of measurements. The workflow is shown in Figure 4. First of all, the current status is checked and there could be two scenarios: (1) The system is in the normal status; (2) The system is in the disturbance status. In the first scenario, for each $i \in 1 \dots m$, the deviation $\delta_i(n) = |x_{in} - \mu_i(n)|$ is calculated, where $\mu_i(n) = \frac{1}{K}\sum_{j=n-K}^{n-1} x_{ij}$ is the current mean. If all the percentage deviations are smaller than the triggering threshold, namely $\delta_i(n)/\mu_i(n) < \theta$ for all $i \in 1 \dots m$, no disturbance is detected and the normal status remains. Otherwise, status is changed to disturbance and the disturbance count is set to 1. In the second scenario, we assume a disturbance will last for at least 3 periods. So if the disturbance count is smaller than 3, the disturbance status remains and the count is increased by 1. Otherwise, for each $i \in 1 \dots m$, a special standard deviation of the most recent 3 periods are calculated as $\sigma_i(n) = \sqrt{\frac{1}{3}\sum_{j=n-2}^{n}(x_{ij} - \mu_i(n))^2}$. If all the percentage standard deviations are smaller than the triggering threshold, namely $\sigma_i(n)/\mu_i(n) < \theta$ for all $i \in 1 \dots m$, the disturbance is considered over and the status is changed back to normal. Finally, the detector outputs the current status in all cases.

[4]There could be missing data points in $\vec{x}_n$ and in the input matrix of the compressor. We fill in each missing data point by its current mean and record it by a boolean. We omit this process in the workflow for simplicity.



(a) Frequency (b) Voltage

Fig. 5. Example of disturbance detection

Examples of using the disturbance detector for both frequency and voltage are shown in Figure 5. Blue lines represent normal status and red lines represent disturbance status. We can see that for both measurements, the disturbances can be detected quickly and the normal status is retained soon after the disturbances end. Note that the status here describes the entire system, therefore it is shared among all the measurements. So as long as one measurement (f or vm) in one of the PMUs contains disturbances, all the five kinds of measurements from all PMUs for this PDC are considered in disturbance status.

B. Compressor

The purpose of the compressor is to use a combination of PCA and DCT to compress the input matrix in an intelligent way in order to reduce the data volume to send while keeping the sampling window small and maintaining a certain accuracy for the reconstructed data. It mainly uses two techniques to achieve that: the transformation matrix reuse in PCA and the self-adapt PC number selection. The measurements of each kind are processed separately and in parallel. Hence, there are actually five instances of the compressor running simultaneously, each for one of the five kinds of measurements.

The workflow of the compressor is shown in Figure 6. Assume the input is an $m \times n$ matrix $\mathbf{X}$. The compressor first checks whether n is equal to the maximum window size K. If $n = K$, it means the system is in normal status and the matrix is of standard size $m \times K$. The transformation matrix reuse module is then called. If $n < K$, it means the system is in disturbance status and the self-adapt PC number selection module is called. These two modules are explained as follows:

- *Transformation Matrix Reuse*: The intuition behind this module is that temporally closed standard matrices could share very similar PCs, namely transformation matrix $\mathbf{P}$. We use $\mathbf{P}_{pre}$ to represent the latest transformation matrix calculated and sent to the receiving side in previous iterations[5]. Whenever a new data block is ready to be compressed, instead of recalculating and resending each time a new transformation matrix, $\mathbf{P}_{pre}$ is tested for reuse. To be more specific, the module compresses $\mathbf{X}$ using $\mathbf{P}_{pre}$ followed by DCT. Then it reconstructs $\widetilde{\mathbf{X}}$ by inverse DCT and inverse PCA based on $\mathbf{P}_{pre}$. The reconstruction accuracy is evaluated by the maximum relative error defined as[6]:

$$\Delta(\mathbf{X}, \widetilde{\mathbf{X}}) = \max_{\substack{i=1\dots m \\ j=1\dots n}} \frac{|x_{ij} - \tilde{x}_{ij}|}{\xi_i} \tag{5}$$

where ξ_i equals the current mean defined in IV-A for f, vm, im and equals to 360 for va and ia. If the maximum relative error is less or equal to the specified tolerance represented as τ[7], $\mathbf{P}_{pre}$ can be reused and the compressor only outputs the compressed data. Otherwise, $\mathbf{P}_{pre}$ cannot be reused and the self-adapt PC number selection module is called.

- *Self-adapt PC Number Selection*:
While using PCA to perform dimensionality reduction, one needs to decide the number of PCs to keep, represented as r. The trade-off here is that decreasing r will decrease the size of the compressed data but increase the error of reconstructed data. The purpose of the self-adapt PC number selection module is to select the proper r to minimize the compressed data size while keeping the reconstructed error in a certain threshold τ. Equation 3 and a variance threshold γ is used to select r. The module works in an iterative way. The iteration starts after matrices $\mathbf{D}$ and $\mathbf{P}$ are calculated according to equation 2 and γ is initialized as γ_0. In each iteration, the smallest r that satisfies $\Gamma(r) \geq \gamma$ is selected, where $\Gamma(r)$ is defined in equation 3. The transformation matrix is then selected as $\mathbf{P} = \mathbf{E}(r)^T$. The compressed matrix $\mathbf{Z}$ is calculated by projecting $\mathbf{X}$ by $\mathbf{P}$ followed by DCT. If $size(\mathbf{P}) + size(\mathbf{Z}) < size(\mathbf{X})$, data size is reduced after compression and the iterations continues. Otherwise, no size reduction is gained after compression and the original matrix $\mathbf{X}$ is assigned to the result. After the size check, $\widetilde{\mathbf{X}}$ is reconstructed by inverse DCT and inverse PCA. The maximum relative error is calculated and compared with the tolerance. If $\Delta(\mathbf{X}, \widetilde{\mathbf{X}}) \leq \tau$, the current r satisfies the required reconstruction accuracy and the compressed matrix $\mathbf{Z}$ and transformation matrix $\mathbf{P}$ are assigned to the result. Otherwise, the variance threshold γ is increased and a new iteration begins. Note that the $\mathbf{P}$ in result are recorded by $\mathbf{P}_{pre}$ only when the compression succeeds and X is of standard size, namely $n = K$.

V. Performance Evaluation

In this section, we present the performance evaluation of ISAAC in terms of time, CR, and accuracy. The experiments run on a dataset consisting of field synchrophasor data collected from a microgrid at Illinois Institute of Technology (IIT) [24]. The dataset contains 18 hours of data (including disturbances) collected from 11 PMUs running at 60 Hz from 6 PM, 1/28/2014 to noon, 1/29/2014. The dataset includes 11 sequences of frequency measurements, 94 sequences of voltage synchrophasors, and 119 sequences of current synchrophasors.

The parameter values used in ISAAC are shown in Table I. Note that $\tau_f < \theta_f$ and $\tau_{vm} < \theta_{vm}$, so that reconstruction

[5] $\mathbf{P}_{pre}$ is only updated when a new $\mathbf{P}$ is calculated and actually sent.

[6] This is slightly different from the standard definition of maximum relative error

[7] τ has different values for different measurement kinds and τ should always be smaller than θ to prevent reconstruction error from triggering disturbances.

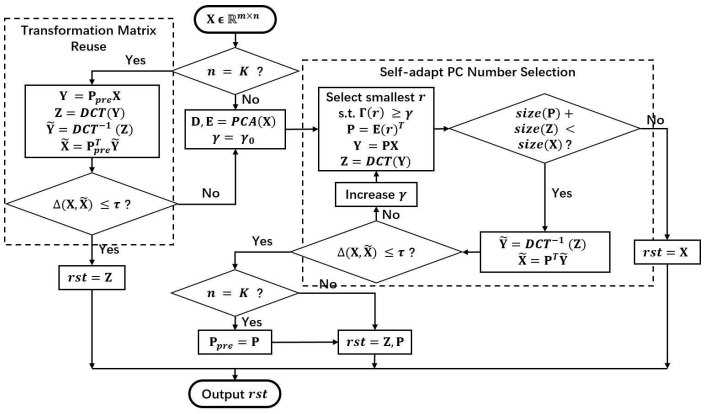


Fig. 6. Workflow of the compressor

error won't trigger disturbances. With the maximum sampling window size $K = 10$[8] and arrival rate of 60 samples/second, the sampling window of 10 samples has a length of 167 ms. All the parameter values are not fixed and can be tuned by users to satisfy their own needs. Of the two WAMS architecture scenarios mentioned in III-A, we choose the WAN-WAN scenario for experiments since it is likely to have a longer communication delay and serves as a worse case for validation. We assume all the 11 PMUs are connected to one PDC via WAN and the PDC is connected with a control center via WAN. According to [25], the communication delay between a PMU and a PDC connected by WAN follows a bimodal distribution containing two normal distributions. We assume the communication delays between each pair of PMU and PDC and between PDC and control center are all i.i.d. random variables following the same bimodal distribution with $p = 0.5$, $\mu_1 = 10$ ms, $\sigma_1 = 1$ ms, $\mu_2 = 16$ ms, $\sigma_2 = 3$ ms, where p is the mixing factor. The absolute timeout value of the PDC is set to 25 ms.

TABLE I
DEFAULT PARAMETER VALUES OF ISAAC

Parameter	Value	Parameter	Value	Parameter	Value
K	10	τ_f	0.025%	τ_{im}	4%
θ_f	0.1%	τ_{vm}	0.2%	τ_{ia}	0.5%
θ_{vm}	1%	τ_{va}	0.5%	γ_0	0.5

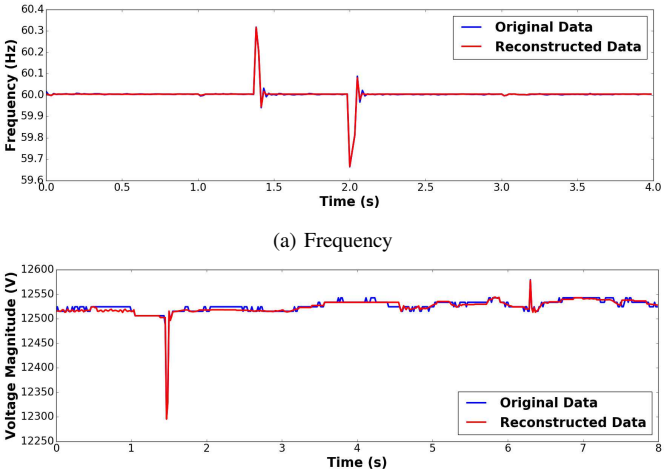
The compression ratio is calculated by the raw data size divided by the compressed data size. And the accuracy of reconstructed data is measured in terms of the maximum percentage error (MPE) and the normalized root-mean-square error (NRMSE). Assume the original data matrix is $\mathbf{X} \in \mathbb{R}^{m \times t}$ and the reconstructed matrix is $\widetilde{\mathbf{X}} \in \mathbb{R}^{m \times t}$. MPE and NRMSE can be calculated as

$$MPE = \max_{\substack{i=1\ldots m \\ j=1\ldots t}} \frac{|x_{ij} - \tilde{x}_{ij}|}{\mu_i} \times 100\% \quad (6)$$

$$NRMSE = \frac{1}{m} \sum_{i=1}^{m} \sqrt{\frac{\sum_{j=1}^{t} (x_{ij} - \widetilde{x}_{ij})^2}{t}} \Big/ \mu_i \ , \quad (7)$$

where $\mu_i = \sum_{j=1}^{t} x_{ij}/t$ for f, vm, im, and $\mu_i = 360$ for va and ia.

[8]A larger K will result in a higher compression ratio but also a higher delay. Here we choose $K = 10$ for a trade-off.



(a) Frequency

(b) Voltage

Fig. 7. Original and reconstructed data

There are two implementations of ISAAC, namely PCA-D/DCT and PCA-S/DCT. The CR, MPE, NRMSE of each type of measurement for the two implementations are shown in Table II. It can be seen that good CRs can be achieved while maintaining satisfactory reconstruction accuracy. The only exception is the current magnitude data, where it is hard to compress without introducing relatively large MPE. This is due to the extremely noisy current magnitude data we observe. In this case, it might not worth compressing current magnitude data to avoid sacrificing data fidelity.

TABLE II
CR, MPE, AND NRMSE OF TWO IMPLEMENTATIONS

Measurement	PCA-D/DCT			PCA-S/DCT		
	CR	MPE (%)	NRMSE	CR	MPE (%)	NRMSE
f	9.11	0.025	4.12E-5	5.21	0.025	3.01E-5
vm	20.36	0.202	3.36E-4	19.86	0.202	3.31E-4
va	20.58	0.500	1.20E-3	21.45	0.500	1.05E-3
im	2.72	6.937	5.02E-3	1.31	6.937	5.05E-3
ia	3.52	0.500	5.96E-4	1.47	0.500	6.87E-4

Figure 7 illustrates a comparison between original data (blue lines) and reconstructed data (red lines) for two sample frequency and voltage measurement sequences, based on the PCA-D/DCT implementation of ISAAC. As we can see from the figure, most noises are discarded while critical changes and disturbances are preserved. We also evaluate the performance of DCT alone on data compressed by PCA already. After compressing the data by PCA-D, DCT can further reduce the data size by 34.2% for f and 14.2% for vm.

Based on the accuracy requirements of the WAMS applications, τ can be changed to achieve different accuracy levels. With various τ_f, the CR, MPE, NRMSE of frequency measurements based on the PCA-D/DCT implementation of ISAAC is shown in Table III[9]. We can see that the CR and reconstruction error both decrease as τ decreases. Therefore, usually the largest τ that satisfies the accuracy requirements should be selected to maximize the CR.

To demonstrate ISAAC's ability to satisfy the real-time requirements of WAMS applications, we also evaluate and

[9]These results are based on one hour of data in the IIT dataset.

TABLE III
CR, MPE, AND NRMSE OF FREQUENCY WITH VARIOUS τ_f

τ_f	CR	MPE (%)	NRMSE
0.001	18.31	0.1	1.52E-4
0.0005	14.27	0.05	6.42E-5
0.00025	8.37	0.025	4.20E-5
0.0001	3.44	0.01	2.37E-5
0.00005	1.79	0.005	1.32E-5

processing time of different components and the end-to-end data delay for both normal and disturbance status. The end-to-end delay is a sum of the communication delay between PMU and PDC, PDC processing delay, communication delay between PDC and control center, and the data reconstruction time. We run ISAAC on a Mac with an Intel Core i7 1.7GHz CPU and 8GB memory. The average processing time of the disturbance detector, compressor and data reconstruction (done at the control center) for each period is shown in Table IV. It can be seen that the processing time is quite negligible and should be even shorter on real PDCs which are much more powerful devices than our Mac. The average and the maximum end-to-end delays of normal and disturbance status are shown in Table V. We can see that the end-to-end delays in disturbance status are significantly smaller than delays in normal status, which is desirable by most of the WAMS applications. And a maximum delay of 201 ms satisfies the delay requirements of real-time WAMS applications [26], [27].

TABLE IV
AVG. PROCESSING TIME (MILLISECOND)

disturbance detector	0.12
compressor	4.75
reconstruction	0.46

TABLE V
END-TO-END DATA DELAY (MILLISECOND)

Status	Avg	Max
normal	107.1	201.4
disturbance	32.5	49.6

VI. CONCLUSION

In this paper, we present ISAAC, an intelligent synchrophasor data real-time compression framework for WAMS to be deployed in Layer 2 PDCs. Based a combination of PCA and DCT techniques, ISAAC is able to mitigate the burden on communication and storage systems laid by the huge synchrophasor data volume while satisfying the requirements of real-time WAMS applications. A disturbance detector is utilized to identify disturbance data and satisfy its stricter delay and accuracy requirements. The use of two techniques named transformation matrix reuse in PCA and self-adapt PC number selection enables ISAAC to achieve good compression ratios while maintaining satisfying delay and accuracy for the reconstructed data. The performance of ISAAC is validated by experiments based on real synchrophasor data.

REFERENCES

[1] M. Patel, S. Aivaliotis, E. Ellen *et al.*, "Real-time application of synchrophasors for improving reliability," *NERC Report, Oct*, 2010.

[2] F. Zhang, L. Cheng, X. Li, Y. Sun, W. Gao, and W. Zhao, "Application of a real-time data compression and adapted protocol technique for wams," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 653–662, 2015.

[3] (2014, October) Synchrophasor technology fact sheet. [Online]. Available: https://www.naspi.org/File.aspx?fileID=1326

[4] R. Klump, P. Agarwal, J. E. Tate, and H. Khurana, "Lossless compression of synchronized phasor measurements," in *Power and Energy Society General Meeting, 2010 IEEE*. IEEE, 2010, pp. 1–7.

[5] M. Ringwelski, C. Renner, A. Reinhardt, A. Weigel, and V. Turau, "The hitchhiker's guide to choosing the compression algorithm for your smart meter data," in *Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International*. IEEE, 2012, pp. 935–940.

[6] J. Kraus, P. Štěpán, and L. Kukačka, "Optimal data compression techniques for smart grid and power quality trend data," in *Harmonics and Quality of Power (ICHQP), 2012 IEEE 15th International Conference on*. IEEE, 2012, pp. 707–712.

[7] J. Ning, J. Wang, W. Gao, and C. Liu, "A wavelet-based data compression technique for smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 212–218, 2011.

[8] M. Wang, J. H. Chow, P. Gao, X. T. Jiang, Y. Xia, S. G. Ghiocel, B. Fardanesh, G. Stefopolous, Y. Kokai, N. Saito *et al.*, "A low-rank matrix approach for the analysis of large amounts of power system synchrophasor data," in *System Sciences (HICSS), 2015 48th Hawaii International Conference on*. IEEE, 2015, pp. 2637–2644.

[9] P. H. Gadde, M. Biswal, S. Brahma, and H. Cao, "Efficient compression of pmu data in wams," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2406–2413, Sept 2016.

[10] J. C. S. de Souza, T. M. L. Assis, and B. C. Pal, "Data compression in smart distribution systems via singular value decomposition," *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 275–284, 2017.

[11] M. Chenine, K. Zhu, and L. Nordstrom, "Survey on priorities and communication requirements for pmu-based applications in the nordic region," in *PowerTech, 2009 IEEE Bucharest*. IEEE, 2009, pp. 1–8.

[12] D. E. Bakken, A. Bose, C. H. Hauser, D. E. Whitehead, and G. C. Zweigle, "Smart generation and transmission with coherent, real-time data," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 928–951, 2011.

[13] K. Khandeparkar, K. Ramamritham, R. Gupta, A. Kulkarni, G. Gajjar, and S. Soman, "Timely query processing in smart electric grids: Algorithms and performance," in *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*. ACM, 2015, pp. 161–170.

[14] C. Martinez, M. Parashar, J. Dyer, and J. Coroas, "Phasor data requirements for real time wide-area monitoring, control and protection applications," *EIPP White Paper*, vol. 26, p. 8, 2005.

[15] H. Jin, S. Uludag, K.-S. Lui, and K. Nahrstedt, "Secure data collection in constrained tree-based smart grid environments," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*. IEEE, 2014, pp. 308–313.

[16] "Ieee guide for phasor data concentrator requirements for power system protection, control, andmonitoring," *IEEE Std C37.244-2013*, pp. 1–65, May 2013.

[17] A. Moga and T. Locher, "Scalable and reliable monitoring for power systems," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov 2015, pp. 259–264.

[18] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[19] sklearn.decomposition module. [Online]. Available: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

[20] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[21] scipy community. [Online]. Available: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.fftpack.dct.html

[22] N. A. E. R. C. (NERC), "Prc-002-2 disturbance monitoring and reporting requirements," April 2017.

[23] M. Biswal, S. M. Brahma, and H. Cao, "Supervisory protection and automated event diagnosis using pmu data," *IEEE Transactions on Power Delivery*, vol. 31, no. 4, pp. 1855–1863, Aug 2016.

[24] R. W. G. C. for Electricity Innovation. Microgrid project at iit. [Online]. Available: http://iitmicrogrid.net/microgrid.aspx

[25] K. Zhu, M. Chenine, L. Nordström, S. Holmström, and G. Ericsson, "An empirical study of synchrophasor communication delay in a utility tcp/ip network," *International Journal of Emerging Electric Power Systems*, vol. 14, no. 4, pp. 341–350, 2013.

[26] M. Chenine, K. Zhu, and L. Nordstrom, "Survey on priorities and communication requirements for pmu-based applications in the nordic region," in *PowerTech, 2009 IEEE Bucharest*. IEEE, 2009, pp. 1–8.

[27] D. E. Bakken, A. Bose, C. H. Hauser, D. E. Whitehead, and G. C. Zweigle, "Smart generation and transmission with coherent, real-time data," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 928–951, 2011.