Decomposition-Based Approximation of Time Series Data with Max-Error Guarantees

Boyu Ruan, Wen Hua(✉), Ruiyuan Zhang, and Xiaofang Zhou

The University of Queensland, Brisbane, QLD 4067, Australia
{b.ruan,w.hua,ruiyuan.zhang}@uq.edu.au, zxf@itee.uq.edu.au

Abstract. With the growing popularity of IoT nowadays, tremendous amount of time series data at high resolution is being generated, transmitted, stored, and processed by modern sensor networks in different application domains, which naturally incurs extensive storage and computation cost in practice. Data compression is the key to resolve such challenge, and various compression techniques, either lossless or lossy, have been proposed and widely adopted in industry and academia. Although existing approaches are generally successful, we observe a unique characteristic in certain time series data, i.e., significant periodicity and strong randomness, which leads to poor compression performance using existing methods and hence calls for a specifically designed compression mechanism that can utilise the periodic and stochastic patterns at the same time. To this end, we propose a decomposition-based compression algorithm which divides the original time series into several components reflecting periodicity and randomness respectively, and then approximates each component accordingly to guarantee overall compression ratio and maximum error. We conduct extensive evaluation on a real world dataset, and the experimental results verify the superiority of our proposals compared with current state-of-the-art methods.

Keywords: Time series compression · High periodicity · Strong randomness · Decomposition-based algorithm · Max-error guarantee

1 Introduction

With the growing popularity of IoT (Internet of Things), tremendous amount of time series data is being generated nowadays by modern sensor networks in various domains [1], such as power grids, manufacturing networks, and medical care systems. For example, due to the upgrading process from conventional power grid to "smart grid", all levels of components in a grid, ranging from power plants, substations, transformers, distributors to smart home appliances, are being monitored simultaneously. Hence, the central controller needs to receive and process massive high resolution time series data from the smart electricity meters and other sorts of measurement devices [2]. Apparently, this will cause problems in terms of data storage, transmission, and processing. First, the expense on data

Z. Huang et al. (Eds.): ADC 2017, LNCS 10538, pp. 71–82, 2017.
DOI: 10.1007/978-3-319-68155-9_6

storage is always a crucial financial concern for any organisation who owns millions of customers and measurement equipments. It is natural for the size of time series data to reach petabyte level, which requires enormous amount of storage space. Second, since massive data is being generated at real time, it will cause congestion during transmission.

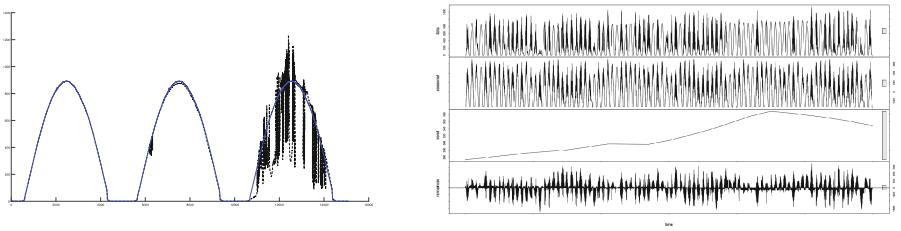
Data compression is the key to overcome the aforementioned problems. Existing data compression techniques can be roughly classified into two categories: lossless compression and lossy compression [3,4]. As its name implies, lossless compression can fully guarantee the compression accuracy, and the original data can be reconstructed flawlessly from the compressed data. Nevertheless, the compression ratio of lossless techniques is generally much smaller than that of the lossy techniques. As a result, challenges caused by the high volume of time series data cannot be easily resolved by lossless methods. Lossy compression techniques, on the contrary, achieve a higher compression ratio at a slight sacrifice of the quality of the compressed data, and hence have attracted extensive attention from both industry and academia during recent decades.

So far, many model-based lossy compression techniques [5] have been proposed for various application domains where either the time series data is strictly stationary or it is compressed without any error guarantee. However, in practice, many time series data, such as traffic volume, illumination, solar energy generation, etc., demonstrate a common characteristic: they contain both *significant global periodicity* and *high local randomness*. As can be observed from Fig. 1(a) which illustrates solar energy generation for three consecutive days, although data of each day follows an approximately same distribution (periodicity), they differ from each other to various extent because of natural weather conditions (randomness). In this case, it is hardly possible for existing model-based compression methods to detect an appropriate model to approximate the original data due to frequent local fluctuations. Furthermore, the repeated patterns in each period can be utilised as well to improve compression ratio.

In this work, we propose a decomposition-based compression technique which divides the original time series into periodic component and random component (e.g., STL decomposition in Fig. 1(b)), and then approximates each component respectively using different methods to achieve the best performance. Our contributions can be summarised as below.

- We observe a common characteristic in certain time series data, i.e., high global periodicity and strong local randomness, where existing data compression techniques cannot be easily applied.
- We design a novel DBA (Decomposition-Based Approximation) algorithm to approximate such time series with max-error guarantee, based on which an optimised algorithm DBAfS (Decomposition-Based Approximation for Streams) is further proposed for online stream data compression.
- Experimental results on a real world dataset verify the superiority of our proposals compared with state-of-the-art compression techniques.

The rest of this paper is organised as follows: in Sect. 2, we briefly discuss related work for time series compression; then we formally define the research



(a) Solar energy generation. (b) STL decomposition.

Fig. 1. An example of periodic and stochastic solar energy generation data and its STL decomposition result. The four signals in Fig. 1(b) represent original data, *seasonal* part, *trend* part, and *remainder* part, respectively.

problem and introduce in detail our proposed algorithms in Sects. 3, 4 and 5 respectively. Our experimental results are described in Sect. 6, followed by a brief conclusion in Sect. 7.

2 Related Work

Time series data compression is a widely researched topic, and various compression mechanisms have been proposed in recent decades. With respect to the application scenario, existing techniques can be divided mainly into two categories: local data compression and streaming data compression [4]. Local data compression aims to reduce the storage cost on local databases and enable faster processing of online queries with less I/O cost. During compression, new patterns of the raw data need to be recognised to help further data analysis. Streaming data compression, on the other hand, targets on data transmission. It learns and adjusts models for newly observed data, and has the superiority of extending sensors' battery life and making better use of limited communication bandwidth. Both methods play an important role in the field of time series research, especially in data management and data mining [6].

In terms of compression strategy, many techniques have been applied for generating an abstract representation of time series, including fourier transform [7], wavelet transform [8,9], symbolic representation [10–12] and piecewise regression [5]. These techniques seek to find an approximate representation that is smaller in size than the original time series, without losing much information contained in the original data. In particular, fourier transform and wavelet transform have been used to extract features from time series for compression and enable efficient subsequence matching. These techniques cannot provide an error guarantee for the compressed data which, however, is crucial for many real world data analysis tasks. Symbolic time series representation is based on data discretization and can achieve high compression ratio. Whereas, applications of the compressed data are limited to statistics and machine learning algorithms for some selected purposes. Piecewise regression is the most widely adopted compression technique

which can approximate time series data with guaranteed quality at every data point. It divides a time series into fixed-length or variable-length intervals and represents each interval with different regression models. Currently, various models have been explored for piecewise regression, including constant models (e.g., PAA [13], PCA [14], APCA [15], PCH [16], etc.), linear models (e.g., PLA [17], PWLH [16], SWAB [18], SF [19], etc.), and nonlinear models (e.g., polynomial functions [20], CHEB [21], etc.), among which polynomial functions achieve the best balance between compression accuracy (under L_∞ norm) and efficiency.

As discussed in Sect. 1, existing compression techniques are largely inapplicable for time series data with both high periodicity and strong randomness. Therefore, we propose a decomposition-based compression algorithm with error guarantees under the L_∞ norm. The experimental results in Sect. 6 will demonstrate the limitations of existing compression methods and the superiority of our proposed algorithm in this case.

3 Problem Definition

In this work, we investigate the problem of time series data compression. We first present the formal definitions of time series and time series compression.

Definition 1 ***(Time Series).*** *A time series D is defined as a sequence of data points (t_i, v_i), i.e., $D = \{(t_i, v_i)\} = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\}$, where t_i is the timestamp of v_i organised monotonically, i.e., $\forall j < k$, $t_j < t_k$. v_i represents the data value at time t_i, and it can be either one-dimensional or multi-dimensional.*

In practice, a time series dataset is sometimes extremely large from the following two perspectives: (1) there could be a huge amount of time series, and (2) each time series could be very long. In order to reduce its storage consumption and guarantee its usability in real world applications, a wisely designed compression strategy is indispensable. In particular, given a time series in the form of $\{(t_i, v_i)\}$, we focus on transforming it into another sequence of bits, such that the error for each v_i does not exceed a predefined threshold ϵ. The ultimate goal is to reduce the number of bits required for representing the original time series as much as possible. Note that we adopt the L_∞ metric for error constraint, which guarantees the accuracy of each data point (t_i, v_i). This is a widely used metric in recent time series compression techniques. Hence, the problem of time series compression with error bound is formally defined as follows.

Problem 1 ***(Time Series Data Compression with Error Bound).*** *Given a time series $D = \{(t_i, v_i^D)\}$ where $i \in [1, n]$, and an error threshold ϵ, the problem of time series compression is to find an approximated representation of D such that the reconstructed time series from the approximation, i.e., $D' = \{(t_i, v_i^{D'})\}$, satisfies the following condition:* $\max(e_i) = \max(v_i^{D'} - v_i^D) \leq \epsilon$.

Throughout the paper, we will use D' to denote both compressed time series and reconstructed time series whenever it is clear.

4 Decomposition-Based Compression Mechanism

We observe that most time series data in practice, such as illumination, traffic volume, solar energy generation, and energy consumption, etc., is a combination of necessity and contingency. In other words, they demonstrate both significant global periodicity and strong local randomness. Inspired by this unique property, we propose a decomposition-based compression technique that divides the original time series into several components corresponding to its periodicity and randomness respectively, and then compresses each component separately to guarantee the overall error bound and meanwhile achieve the largest compression ratio.

In particular, given the original time serious data $D = \{(t_i, v_i^D)\}$, we decompose D into three components: $S = \{(t_i, v_i^S)\}$, $T = \{(t_i, v_i^T)\}$, and $r = \{(t_i, v_i^r)\}$, where S, T and r represent the *seasonality*, *trend* and *remainder* parts of D respectively, such that:

$$D = S + T + r \quad i.e., \quad \forall i, v_i^D = v_i^S + v_i^T + v_i^r \tag{1}$$

Here, we divide the original time series with exiting STL (Seasonal Trend Decomposition using Loess) tools. Based on such a decomposition, we redefine the problem of time series compression as follows.

Problem 2 ***(Decomposition-Based Time Series Data Compression with Error Bound).*** *Given a time series $D = \{(t_i, v_i^D)\}$ that can be decomposed into $D = S + T + r$, and an error threshold ϵ, the problem of time series data compression is to find approximated representations of S, T and r such that the reconstructed time series D' and its components, i.e., $S' = \{(t_i, v_i^{S'})\}$, $T' = \{(t_i, v_i^{T'})\}$ and $r' = \{(t_i, v_i^{r'})\}$ respectively, satisfy the following conditions: $D' = \{(t_i, v_i^{D'})\} = S' + T' + r'$ and $\max(e_i) = \max(|v_i^{D'} - v_i^D|) \leq \epsilon$.*

4.1 Component Approximation

In this section, we will introduce our methods to compress each component, i.e., $S = \{(t_i, v_i^S)\}$, $T = \{(t_i, v_i^T)\}$ and $r = \{(t_i, v_i^r)\}$, of the original time series $D = (t_i, v_i^D)$ obtained by STL decomposition.

When conducting STL decomposition, the length of a single period needs to be determined. In practice, most time series have a natural period p (e.g., one day for illumination, solar energy generation, etc.) which, however, might be problematic if adopted directly for compression. The trend part for each natural period could be dramatically different, which will then lead to low compression ratio. But if a longer period is adopted, as in Fig. 1(b), the trend part will become smoother and easier to compress. Therefore, we define a manual period $d \cdot p$ where d is a positive integer, and use $d \cdot p$ to conduct STL decomposition. In this case, each natural period in a manual period follows similar patterns, whereas there are still slight differences (e.g., peak values) among these periods. To avoid the cost of storing the entire manual period, we only preserve its mean

value as a reference. Specifically, we replace the periodic part S with its mean value $\bar{S} = \{(t_i, v_i^{\bar{S}})\}$, and add the difference between S and $\bar{S}$ to the remainder r such that the new remainder will be $r_p = r + S - \bar{S}$. More formally, let $\{S_1, S_2, \ldots, S_d\}$ be a manual period where S_j, $j \in [1, d]$ is the j-th natural period, namely $S_j = \{(t_{p\cdot(j-1)+1}, v^S_{p\cdot(j-1)+1}), \ldots, (t_{p\cdot j}, v^S_{p\cdot j})\}$. We calculate the mean value of $\{S_1, S_2, \ldots, S_d\}$ based on Eq. 2, where $i\%p$ represents the modulo operation, i.e., $i\%p = i - \lfloor \frac{i}{p} \rfloor \cdot p$.

$$\forall i \in [1, d \cdot p], v_i^{\bar{S}} = \frac{\sum_{j=1}^{d} v^S_{p\cdot(j-1)+i\%p}}{d} \tag{2}$$

We then store only one period of $\bar{S}$ as the representative of the periodic part S. Since we have added the difference between S and $\bar{S}$ into the remainder part r_p, there is no information loss when compressing the periodic part $\bar{S}$ in this way. We will show the impact of d on the compression performance in Sect. 6.

We adopt piecewise linear approximation to compress the trend part T because it demonstrates an obvious linear property, as in Fig. 1(b). The information we need to store is just the coefficients of line segmentations in T'. For the remainder part r_p, we divide its value range $[\min(v^{r_p}), \max(v^{r_p})]$ into M intervals and construct a histogram on these M intervals. We then approximate each interval as well as it constituting values $v_i^{r_p}$ with the median of that interval. Finally, we adopt *Huffman coding* to encode these median values and transform the original remainder part r_p into a sequence of Huffman codes r'_p. *Run-length coding* can also be applied to further reduce the size of r'_p. The whole DBA approximation algorithm is presented in Algorithm 1.

4.2 Error Analysis

The error between D' and D equals to the sum of errors when approximating S, T and r respectively. Since we transfer the difference between S' and S into r_p, the error under L_∞ norm can be calculated as below.

$$e_i = |v_i^{D'} - v_i^D| = |(v_i^{T'} - v_i^T) + (v_i^{r'_p} - v_i^{r_p})| \tag{3}$$

We can approximate T using any piecewise linear approximation algorithm, such as PLA, PWLH, SWAB, SF, etc., that guarantees L_∞ norm accuracy, namely

$$\max(e_i^T) = \max(|v_i^{T'} - v_i^T|) \leq \epsilon_1 \tag{4}$$

For the remainder part r_p, since we divide its value range into M intervals and use the median values for Huffman coding, the maximum error of r_p is

$$\max(e_i^{r_p}) = \max(|v_i^{r'_p} - v_i^{r_p}|) \leq \frac{\max(v^{r_p}) - \min(v^{r_p})}{2M} \tag{5}$$

We can choose a proper M such that $M \geq \frac{\max(v^{r_p}) - \min(v^{r_p})}{2\epsilon_2}$ to guarantee $\max(e_i^{r_p}) \leq \epsilon_2$. Combining Inequations 4 and 5, we can conclude that

$$\max(e_i) \leq \max(e_i^T) + \max(e_i^{r_p}) \leq \epsilon_1 + \epsilon_2 \tag{6}$$

By simply setting $\epsilon_1 + \epsilon_2 \leq \epsilon$, the error bound $\max(e_i) \leq \epsilon$ is satisfied.

Algorithm 1. Decomposition-Based Approximation (DBA)

Input:
D: original time series
ϵ_1, ϵ_2: error bounds for trend and remainder
p, d: natural and manual periods

Output:
D': compression result of D

1: Divide D into S, T and r using STL decomposition
2: $S' \leftarrow \bar{S}$ calculated by Equation 2
3: $T' \leftarrow \textsc{LinearAppro}(T, \epsilon_1)$
4: $r_p \leftarrow r + S - S'$
5: $M \leftarrow \frac{\max(v^{r_p}) - \min(v^{r_p})}{2\epsilon_2}$
6: Divide the value range of v^{r_p} into M intervals and encode the median m_k of each interval using Huffman coding, $k \in [1, M]$
7: $v^{r_p'} \leftarrow m_k$ of corresponding interval
8: $D' \leftarrow S' + T' + r_p'$

5 Online Compression Mechanism

In order to conduct STL decomposition, the original time series needs to be provided beforehand. Therefore, it is impossible to purely decompose a single data point (t_i, v_i) into the form $v_i = v_i^S + v_i^T + v_i^r$. In other words, the compression technique described in Sect. 4 can only be applied for offline cases. In this section, we introduce our optimisation to make Algorithm 1 capable of compressing stream data. We observe that the periodic part of original time series roughly follows certain distributions. Hence we can divide a time series $D = \{t_i, v_i^D\}$ into two components N_s and r_s, where $N_s = \{(t_i, v_i^{N_s})\}$ fits a pre-learned distribution and $r_s = \{(t_i, v_i^{r_s})\}$ is the remainder, such that:

$$D = N_s + r_s \quad i.e., \quad \forall i, v_i^D = v_i^{N_s} + v_i^{r_s} \tag{7}$$

In this way, $v_i^{r_s}$ can be calculated at real time and the algorithm can be applied for online stream data compression. Based on such a decomposition, we define the problem of stream data compression as follows.

Problem 3 ***(Decomposition-Based Stream Data Compression with Error Bound).*** *Given a time series $D = \{(t_i, v_i^D)\}$ that can be decomposed into $D = N_s + r_s$, and an error threshold ϵ, the problem of stream data compression is to find an approximated representation $r_s' = \{(t_i, v_i^{r_s'})\}$, such that $D' = \{(t_i, v_i^{D'})\} = N_s + r_s'$ and $\max(e_i) = \max(|v_i^{D'} - v_i^D|) \leq \epsilon$.*

5.1 Component Approximation

Recall that we divide D in a simpler form $D = N_s + r_s$ rather than STL decomposition. We first learn the distribution of the periodic part from historical data.

Algorithm 2. Decomposition-Based Approximation for Stream (DBAfS)

Input:

D: original time series

ϵ: error bound p, d: natural and manual periods

Output:

D': compression result of D

1: Learn the proper N_s from $d \cdot p$ period of D
2: $r_s \leftarrow D - N_s$
3: $M \leftarrow \frac{\max(v^{r_s}) - \min(v^{r_s})}{2\epsilon}$
4: Divide the value range of v^{r_s} into M intervals and encode the median m_k of each interval using Huffman coding, $k \in [1, M]$
5: $v^{r_s'} \leftarrow m_k$ of corresponding interval
6: $D' \leftarrow N_s + r_s'$

In the case of solar energy generation, a Gaussian distribution $N_s = (\mu_s, \sigma_s)$ can be observed. But the problem is how many historical data should be utilised to learn the parameters of the distribution. On one hand, the distribution should be generalised enough to cover as much data as possible. On the other hand, it needs to be locally typical so that not too much difference is transferred into the remainder part. To this end, we adopt data within a manual period $d \cdot p$ to learn the distribution. Then we approximate r_s in the same way as r_p described in Sect. 4. Since $v_i^{r_s}$ can be calculated directly by $v_i^{r_s} = v_i^D - v_i^{N_s}$ at every timestamp t_i, this technique can be executed every time when a new data point arrives in the stream. The whole approximation algorithm is presented in Algorithm 2. After compression, we only need to store the coefficients of N_s and the Huffman coding results of r_s'. As before, run-length coding can be applied to further reduce the size of r_s'.

5.2 Error Analysis

The error between D' and D originates only from the error of approximating r_s. Hence, the error under L_∞ norm can be calculated as follows.

$$e_i = |v_i^{D'} - v_i^D| = |v_i^{r_s'} - v_i^{r_s}| \tag{8}$$

As discussed in Sect. 4, the maximum error of r_s is

$$\max(e_i) = \max(|v_i^{r_s'} - v_i^{r_s}|) \leq \frac{\max(v^{r_s}) - \min(v^{r_s})}{2M}. \tag{9}$$

We can also choose a proper M such that $M \geq \frac{\max(v^{r_s}) - \min(v^{r_s})}{2\epsilon}$ to make sure the error bound $max(e_i) \leq \epsilon$ is satisfied.

6 Experiments and Results

To evaluate the performance of our proposals, we conducted extensive experiments on a real world data set from the smart grid. The data set consists of

information about solar energy generation for one year with each day containing 5,040 data collected at a 10-second resolution. All the experiments were conducted on a PC with Intel Core i7 processor 3.4 GHz and 16 GB main memory.

We compared DBA and DBAfS algorithms with current state-of-the-art lossy compression methods, and examined the impact of parameter setting on the compression performance which were measured in terms of both Compression Ratio (CR, Eq. 10) and accuracy.

$$CR = \frac{|D|}{|D'|} \tag{10}$$

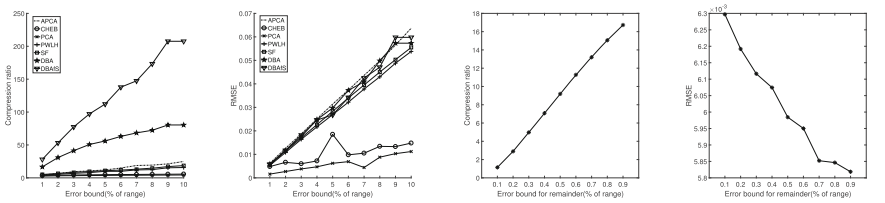
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i^D - v_i^{D'})^2} \tag{11}$$

In Eq. 10, $|D|$ and $|D'|$ represents the storage cost of original data D and compressed data D', respectively. We calculated accuracy with the Root Mean Square Error (RMSE, Eq. 11), a widely adopted metric for evaluating the quality of approximation. We noticed that the range of original data might vary a lot throughout the entire time series. Therefore, we used relative error (in percentage) in the experiments to represent RMSE and error bound ϵ. For the remaining part, we still use ϵ to denote relative error bound whenever it is clear.

6.1 Effect of Error Bound ϵ

As discussed in Sect. 2, piecewise regression is currently the most prevalent lossy compression technique. Hence, we selected several representative piece wise regression methods (i.e., PCA and APCA for constant models, PWLH and SF for linear models, and CHEB for nonlinear models) and compared with our DBA and DBAfS algorithms. Figures 2(a) and (b) demonstrate their respective compression ratio and RMSE, when error bound ϵ ranges from 1% to 10%. We can see that both compression ratio and RMSE of all methods rise with the increasing of ϵ. This is natural and also consistent with the results obtained from [5]. Moreover, DBA and DBAfS achieve the largest compression ratio at the cost of nearly the same RMSE with most counterparts, and such an improvement further enlarges when ϵ increases. In particular, when ϵ is set as 10%, the compression ratio of DBAfS is at least 10 times larger than all the existing piecewise regression methods. Comparing DBA and DBAfS, we observe that DBA is not as powerful as DBAfS with respect to compression ratio, especially when ϵ is large. This is because except for periodic part, DBA needs to compress both trend and remainder parts while DBAfS only handles remainder part. When ϵ is small, the storage cost of the compressed data is dominated by the remainder part, which results in similar compression ratio between DBA and DBAfS. However, when ϵ grows, remainder becomes less dominant and thus the difference between DBA and DBAfS gradually emerges.

To further explore the insights of DBA algorithm, we evaluated the impact of ϵ_1 and ϵ_2 on its compression performance. In this case, we set $\epsilon = 1\%$ and vary



(a) CRs (b) RMSEs (c) CR of DBA (d) RMSE of DBA

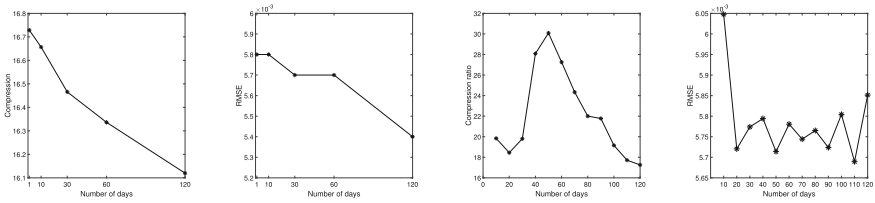
Fig. 2. Effect of error bound ϵ. Figures 2(a) and (b) compare the performance of our techniques with other state-of-the-art compression methods. Figures 2(c) and (d) show the impact of how we distribute ϵ_1 and ϵ on the performance of DBA.

ϵ_2 (i.e., the error bound for remainder part) from 0.1% to 0.9% (resp. ϵ_1 ranges from 0.9% to 0.1%). Figures 2(c) and (d) present the results. It can be observed that larger ϵ_2 leads to higher compression ratio and smaller RMSE. This also verifies the remainder part dominates the size and quality of compressed data in the solar generation data when total error bound ϵ is small, i.e. 1%.

6.2 Effect of Period Length *d*

As mentioned in Sect. 4, most periodic time series have a natural period p, and we can define a manual period $d \cdot p$, which is d times larger than the natural period, to balance between the compression of trend part and remainder part. In the case of solar energy generation, its natural period is one day. We evaluated the impact of d on the compression performance of DBA algorithm. Figures 3(a) and (b) present DBA's compression ratio and RMSE respectively when ϵ is set as 1%. We can see that both compression ratio and RMSE decrease with the increasing of d. It has been verified in previous experiments that in solar energy data, the storage size of compressed data is dominated by remainder part when error bound ϵ is small. The increase of d means more difference is transferred from periodic part to remainder part, which leads to larger deviation between $max(v^{r_p})$ and $min(v^{r_p})$, and hence larger M according to Eq. 5. Consequently, the compression of remainder part takes longer codes and thus more storage cost. This in turn results in smaller compression ratio when d enlarges.

For DBAfS, parameter d is also adopted to trade-off the generalisability and typicality of learned standard distribution of the original data. Therefore, we studied the impact of d on the compression performance of DBAfS, as depicted in Figs. 3(c) and (d) where $\epsilon = 1\%$. When d increases from 0 to 50 days, the learned distribution can capture more periods of original data. However, when d further enlarges, the learned distribution is too generalised and no longer typical for a particular part of the data. This means too much difference is transferred into the remainder part, which will lead to the decrease of compression ratio, as analysed above. Hence, the best setting of parameter d in the DBAfS algorithm for our solar generation data is selected as 50 days.



(a) CR of DBA (b) RMSE of DBA (c) CR of DBAfS (d) RMSE of DBAfS

Fig. 3. Effect of period length d. Figures 3(a) and (b) demonstrate the impact of d on the compression performance of DBA. Figures 3(c) and (d) show the impact of d on the compression performance of DBAfS.

7 Conclusion

In this work, we observe a common characteristic in certain time series data, i.e., significant periodicity and strong randomness, which makes existing compression techniques, either lossless or lossy, largely inefficient in terms of compression ratio. Therefore, we propose two novel decomposition-based compression algorithms, namely DBA and DBAfS, to approximate such time series with max-error guarantees. Our experimental results on a real world data set demonstrate the superiority of our proposals, compared with current state-of-the-art lossy compression methods. As future work, we plan to apply DBA and DBAfS algorithms to more time series data, such as illumination, traffic volume, etc., to further examine their applicability and compression performance.

Acknowledgment. This research is partially supported by the Australian Research Council (Grant No.DP170101172) and the Queensland Government (Grant No.AQRF12516).

References

1. Ratanamahatana, C.A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining time series data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 1049–1077. Springer, Boston (2009)
2. Eichinger, F., Efros, P., Karnouskos, S., Böhm, K.: A time-series compression technique and its application to the smart grid. VLDB J. **24**(2), 193–218 (2015)
3. Tak-chung, F.: A review on time series data mining. Eng. Appl. Artif. Intell. **24**(1), 164–181 (2011)
4. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surveys (CSUR) **45**(1), 12 (2012)
5. Hung, N.Q.V., Jeung, H., Aberer, K.: An evaluation of model-based approaches to sensor data compression. IEEE Trans. Knowl. Data Eng. **25**(11), 2434–2447 (2013)
6. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Mining Knowl. Discov. **7**(4), 349–371 (2003)

7. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Databases, vol. 23. ACM, New York (1994)
8. Chan, K.P., Fu, A.W.C.: Efficient time series matching by wavelets. In: 1999 Proceedings of 15th International Conference on Data Engineering, pp. 126–133. IEEE (1999)
9. Shahabi, C., Tian, X., Zhao, W.: Tsa-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In: Proceedings of 12th International Conference on Scientific and Statistical Database Management, pp. 55–68. IEEE (2000)
10. Lin, J., Keogh, E., Li, W., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining Knowl. Discov. **15**(2), 107 (2007)
11. Shieh, J., Keogh, E.: i sax: indexing and mining terabyte sized time series. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 623–631. ACM (2008)
12. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. In: VLDB (2000)
13. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. **3**(3), 263–286 (2001)
14. Lazaridis, I., Mehrotra, S.: Capturing sensor-generated time series with quality guarantees. In: Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), pp. 429–440 (2003)
15. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Locally adaptive dimensionality reduction for indexing large time series databases. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, SIGMOD 2001, pp. 151–162. ACM, New York, NY, USA (2001)
16. Buragohain, C., Shrivastava, N., Suri, S.: Space efficient streaming algorithms for the maximum error histogram. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 1026–1035 (2007)
17. Chen, Q., Chen, L., Lian, X., Liu, Y., Yu., J.X.: Indexable pla for efficient similarity search. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 435–446. VLDB Endowment (2007)
18. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 289–296 (2001)
19. Elmeleegy, H., Elmagarmid, A.K., Cecchet, E., Aref, W.G., Zwaenepoel, W.: Online piece-wise linear approximation of numerical streams with precision guarantees. Proc. VLDB Endow. **2**(1), 145–156 (2009)
20. Ni, J., Ravishankar, C.V.: Indexing spatio-temporal trajectories with efficient polynomial approximations. IEEE Trans. Knowl. Data Eng. **19**(5), 663–678 (2007)
21. Cai, Y., Ng, R.: Indexing spatio-temporal trajectories with chebyshev polynomials. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004, pp. 599–610. ACM New York, USA (2004)