Optimal Bounded Error Piecewise Linear Approximation with Resolution Reduction for Sensor Data Compression

Jeng-Wei Lin[1], Yu-Hung Tsai[1], Ching-Che Huang[1], Chia-Wei Hsu[2], and Fang-Yie Leu[3(✉)]

[1] Department of Information Management, Tunghai University, Taichung 407224, Taiwan
{jwlin,g12490023,g12490013}@thu.edu.tw
[2] College of Engineering, Tunghai University, Taichung 407224, Taiwan
jcwhsu@thu.edu.tw
[3] Department of Computer Science, Tunghai University, Taichung 407224, Taiwan
leufy@thu.edu.tw

Abstract. As the fast improvement of semiconductor manufacturing, today, IoT devices can acquire data in very high resolution and frequency. In many cases, the granularity is finer than required. More energy is consumed for the sensor data transmission via networks and storage in data centers. Data compression is a common approach to reduce the data size. In the literature, many methods based on Bounded-Error Piecewise Linear Approximation (BEPLA) have been proposed that use multiple straight lines to approximate the original data and maintain a tolerable error. Swing-RR first introduced the Resolution Reduction strategy, abbreviated as RR, where all line segment endpoints must be encoded by small integers, rather than floating point real numbers. Swing-RR is simple and has O(n) time complexity when the date size is n. The compression ratio is significantly better than other BELPA methods. However, it is not optimal in terms of number of line segments. In this paper, an optimal BEPLA algorithm with RR is presented, denoted as OBEPLA-RR. The experiments on public real world time series datasets show that the numbers of line segments generated by OBEPLA-RR are fewer than those by Swing-RR, and the compress ratios are significantly better than Swing-RR and other BEPLA methods.

Keywords: Internet of Things · sensor data · time series · data compression · piecewise linear approximation · error bound

1 Introduction

Many IoT applications have been realized in our daily world, including smart health, smart home, smart transportation, smart logistic, smart city, smart manufacturing, and so on. Continuously monitoring the physical environments and cyberspace, and real-time and periodically analyzing these data by modern advanced artificial intelligence (AI) models give us an opportunity to better understand our world, and help us make better decisions.

D.-J. Deng and J.-C. Chen (Eds.): SGIoT 2024, LNICST 633, pp. 127–137, 2026.
https://doi.org/10.1007/978-3-031-93825-2_12

On the other hand, the challenges for management and analysis of the huge amount of IoT data also have emerged. Various data are sensed continuously by IoT devices for different purposes. They are transmitted from devices to edge nodes, and further to data centers via different network technologies, and stored temporarily or persistently in these systems. It costs a lot of energy to transmit these data via network and store them in disks. The data centers in the U.S. consumed about 1.8% of the total U.S. electricity consumption in 2014 [1]. In another report, the power demand of the data centers in the U.S. is expected to 35 GW by 2030, up from 17 GW in 2022 [2]. Data compression is widely adopted to reduce the disk size, and thus reduce the required disk space and network bandwidth [3–12]. Furthermore, device nodes and edge nodes usually have very limited local buffer. When data is compressed, these nodes can buffer more information.

In many applications, lossy data compression technologies are used to compress sensor data. Tiny differences between the original data and the reconstructed data are allowed. Hence, these technologies can achieve higher compression ratios than lossless data compression technologies. For example, JPEG is adopted for images [13] and H.263 is adopted for video [14].

For time series, Piecewise Linear Approximation (PLA) is widely adopted, where the original data stream is represented by multiple line segments, i.e., linear functions, as shown in Fig. 1. In Fig. 1(a), two (green) line segments are used to represent the original data stream (red dots). We note that the errors between some reconstructed values (green circles) and their original values at the same time ticks are smaller than or equal to a bound, here ε, and some are not. In Fig. 1(b) ~ (d), when all errors must be bounded, they are referred to as Bounded-Error Piecewise Linear Approximation (BEPLA). Although the concept of PLA is simple, it is effective for not only data size reduction, but also for data processing like similarity search [15], classification [16], and forecasting [17].

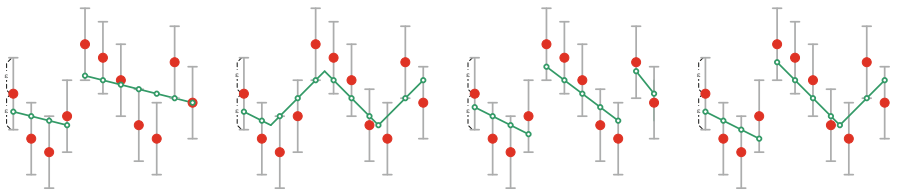
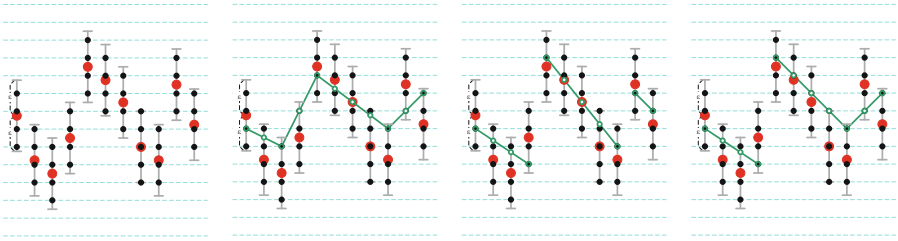


Fig. 1. Concept of Piecewise Linear Approximation (PLA) and Bounded-Error Piecewise Linear Approximation (BEPLA). The original data stream (red dots) are approximated by multiple greed line segments. Green circles (the intersections of green lines and time ticks) are the reconstructed values (color figure online).

In Fig. 1(b), four line segments are used to represent the original data. Since the stop point of a line segment is used as the start point of its successive segment, it is further referred to as connected BEPLA [3–5]. In addition to the start point of the first line segment, it records the stop point of each segment. On the other hand, in Fig. 1(c), line segments are not connected. It is referred to as disconnected BEPLA [6–8]. A new line segment always starts one time tick after its immediate previous segment. Since BEPLA

algorithms have freedom to select the start point of a new segment, fewer line segments are used. In Fig. 1(d), BEPLA algorithms use both connected and disconnected line segments [9, 10].

In [11], BEPLA with Resolution Reduction (RR) was introduced. As shown in Fig. 2, the start and stop points of line segments are selected from a small number of points in the bounded subrange. BEPLA algorithms can use less bits to encode each line segment. A simple BEPLA algorithm, referred to as Swing-RR, had been presented. Although Swing-RR uses more line segments to represent datasets chosen from UCR time series database [18] than state-of-the-art methods, the compression ratios are better.



(a) Resolution Reduction (RR) (b) Connected BEPLA with RR (c) Disconnected BEPLA with RR (d) Mixed BEPLA with RR

Fig. 2. Concept of BEPLA with Resolution Reduction (RR), where only black dots are chosen as start or stop points of line segments. However, the reconstructed data (green circles) are not limited to black dots (color figure online).

In this paper, we present an optimal BEPLA algorithm, referred to as OBEPLA-RR, when Resolution Reduction is taken into consideration. The experiment results on UCR time series database [18] show that OBEPLA-RR use fewer line segments than Swing-RR, and thus the compress rations are furthermore improved.

2 Related Works

2.1 Definition

PLA and BEPLA are wildly used to represent a time series. They have been studied in many different context. In general, PLA and BEPLA can be divided into two types: connected and disconnected, as shown in Fig. 1. Specifically, a time series $Y = \{ y_1, y_2, y_3, \ldots, y_n\}$ can be approximated by a set of m non-overlapped line segments $S = \{ s_1, s_2, s_3, \ldots, s_m\}$, where.

(1) $s_1.start.tick = 1$,
(2) $s_{i+1}.start = s_i.stop$ in connected PLA, i.e., s_i and s_{i+1} are connected, or $s_{i+1}.start.tick = s_i.stop.tick + 1$ in disconnected PLA, for $1 \leq i < m$, and
(3) $s_m.stop.tick = n$.

The approximation $Y' = \{ y'_1, y'_2, y'_3, \ldots, y'_n \}$ represented by S can be derived as the following.

$$y'_j = s_k.\text{start.value} + (j - s_k.\text{start.tick}) * \frac{s_k.\text{stop.value} - s_k.\text{start.value}}{s_k.\text{stop.tick} - s_k.\text{start.tick}}, \quad (1)$$

where $s_k.\text{start.tick} \leq j \leq s_k.\text{stop.tick}$ and $1 \leq k \leq m$

To assess the quality of the approximation Y', the p-norm of errors between Y and Y', usually referred to as l_p-*error*, is commonly used.

$$l_p - error(Y, Y') = \sqrt[p]{\sum_{j=1}^{n} (y_j - y'_j)^p} \quad (2)$$

$$l_\infty - error(Y, Y') = \max_{1 \leq j \leq n} \left(\left| y_j - y'_j \right| \right) \quad (3)$$

Given a maximal tolerable error ε, when l_∞-*error* (Y, Y') $\leq \varepsilon$, S is a BEPLA of Y.

For connected PLA, S can be expressed as the sequence { s_1.start.value, (s_1.length, s_1.stop.value), (s_2.length, s_2.stop.value), …, (s_m.length, s_m.stop.value)}, where s_k.length = s_k.stop.tick – s_k.start.tick for $1 \leq k \leq m$. When p and q bits are used to encode value of segment endpoints and segment length, respectively, $p + (p + q) * m$ bits are used to encode S. Since y_i are usually real numbers, and represented by 32 bit floating point numbers, the compress ratio can be calculated by Eq. (4).

$$Compression\ Ratio_{connected}(S) = \frac{p + (p + q) * m}{32 * n} \quad (4)$$

For disconnected PLA, S can be expressed as the sequence { (s_1.start.value, s_1.length, s_1.stop.value), (s_2.start.value, s_2.length, s_2.stop.value), …, (s_m.start.value, s_m.length, s_m.stop.value)}. Thus, $(2 * p + q) * m$ bits are used to encode S. The compress ratio can be calculated by Eq. (5).

$$Compression\ Ratio_{disconnected}(S) = \frac{(2 * p + q) * m}{32 * n} \quad (5)$$

The authors note that S can be further compressed by lossless data compression technologies [12]. However, it is out of the scope of this paper.

2.2 BEPLA Algorithms

Many BEPLA algorithms have been proposed. For disconnected BEPLA, the first optimal algorithm was presented by O'Rourke in 1981 [3]. Given an error bound, the algorithm runs in a transformed space and uses as few line segments as possible to represent the original time series. In 2009, Elmeleegy et al. presented another optimal algorithm, referred to as SlideFilter, which runs in time space [4]. They also presented a non-optimal algorithm referred to as SwingFilter. Xie et al. improved SildeFilter and gave a linear time optimal algorithm, referred to as OptimalPLR [5]. Since the values of segment

endpoints are real numbers, represented by floating point numbers, and segment lengths are integers, the compression ratio is

$$Compression\ Ratio_{OptimalPLR}(S) = \frac{(2 * 32 + 32) * m}{32 * n} = \frac{3 * m}{n} \tag{6}$$

For connected BEPLA, Hakimi and Schmeichel [6] presented their optimal algorithm, referred to as Cont-PLA, in 1991. Since two consecutive line segments are usually intersected between time ticks, as shown in Fig. 1(b), the segment lengths are usually real numbers, and thus the compression ratio is

$$Compression\ Ratio_{Cont-PLA}(S) = \frac{32 + (32 + 32) * m}{32 * n} = \frac{2 * m}{n} \tag{7}$$

Due to the freedom in choosing start points of line segments, OptimalPLR usually can use fewer line segments to represent the original data. However, more bits are required to encode each line segment in disconnected BEPLA. The experiment results reported in [9] show that Cont-PLA outperforms OptimalPLR on more datasets. However, the advantage depends on both the dataset and error bound. There is no clear winner between OptimalPLR and Cont-PLA.

Luo et al. proposed the mixed approach in 2015, i.e., to use both connected and disconnected line segments in a BEPLA [9]. Their optimal algorithm, referred to as Mixed-PLA, adopts dynamic programming technique. The experiment results showed that Mixed-PLA consistently outperforms OptimalPLR and Cont-PLA by roughly 15%. Zhao et al. improved Mixed-PLA in terms of computing time and memory costs in 2020 [10].

2.3 BEPLA with Resolution Reduction

Optimal BEPLA algorithms mentioned above, OptimalPLR [3–5], Cont-PLA [6], and Mixed-PLA [9, 10] that use disconnected, connected, and mixed line segments, respectively, try to reduce the number of line segments.

On the other hand, Lin et al. proposed to use fewer bits to encode line segments in 2019 [11]. When the values of line segment endpoints are restricted in a fixed number of levels, for example, 2^r levels, the values can be encoded by r-bit unsigned integer. Specifically, the global range of the sensor data is divided equally into 2^r blocks. The center of each block marks a level, and thus, there are 2^r levels totally. As shown in Fig. 2, each horizontal dashed line represents a level. It is referred to as Resolution Reduction, i.e., from 32-bit floating point real number to r-bit unsigned integer, and r is referred to as resolution.

Given an error bound ε (in percentage), for any time tick i, there are at least $\lfloor 2^{r+1}\varepsilon \rfloor$ levels in the allowed subrange between $y_i - \varepsilon$ and $y_i + \varepsilon$ when the block size $1/2^r$ is smaller than or equal to 2ε. The minimal resolution can be found by Eq. (8).

$$r_{min}(\varepsilon) = \lceil -\log_2(\varepsilon) - 1 \rceil \tag{8}$$

Black dots in Fig. 2 are the intersections of horizontal dashed lines (levels) and the allowed subrange of each time tick. Since the endpoints of line segments are chosen from black dots, their value can be encoded by r-bit unsigned integer.

Furthermore, segment lengths are restricted to 2^d for real-time applications and small memory size in the sensor nodes and edge nodes. As a result, segment lengths can be encoded by d-bit unsigned integer.

Taking Resolution Reduction into consideration, Swing-RR was introduced. Based on SwingFilter [4], Swing-RR is very simple and has $O(n)$ time complexity. However, it is not optimal in terms of line segments, and compression ratios.

The experiment results reported in [11] showed that although Swing-RR uses more line segments to approximate the original data stream than SwingFilter and state-of-the-art methods, it uses significantly less bits.

3 Methods

In this paper, we first consider connected BEPLA with Resolution Reduction. Given an error bound ε, resolution $r \geq r_{min}$ (ε), and maximal delay 2^d, BEPLA-RR problem is reduced to find a shortest path from a black dot at the first time tick 1 to a black dot at the last time tick n, referring to Fig. 2(a) and (b), where.

(1) line segments connected only black dots in different time ticks,
(2) for any time tick j in a line segment s_k, i.e., s_k.start.tick $< j <$ s_k.stop.tick, s_k intersects with the allowed subrange between $y_i - \varepsilon$ and $y_i + \varepsilon$, and
(3) all line segments use the same number of bits, i.e., $b + d$ bits.

Without violating condition (2), the two endpoints of a feasible line segment must be visible to each other. Figure 3(a) ~ (c) show the forward visibility of the start point of the green line segment, tick by tick, and Fig. 3(d) shows the backward visibility of the stop point.

Based on forward visibility, Swing-RR greedily finds a longest line segment from a given start point. However, it is not optimal, as shown in Fig. 4(a), where Swing-RR finds three longest line segments in the beginning. However, it finally finds a BEPLA with 5 line segments. Figure 4(b) shows an optimal BEPLA with 4 line segments.

In this paper, an optimal algorithm, referred to as OBEPLA-RR, is presented based on backward feasibility scan and the relaxation strategy used in common shortest path algorithms, such as Bellman-Ford algorithm and Dijkstra's algorithm [19]. Specifically, for a black point v, the cost of the shortest path from any black point at time tick 1 to v can be calculated from the following recursive equation.

$$\begin{cases} cost[v] = 1 & v.\text{tick} = 1 \\ cost[v] = \min_{w \in P(v)\{cost[w]+1\}} & v.\text{tick} = 1 \end{cases}, \tag{9}$$

where $P(v)$ is the set of black points backward visible from v.

Whenever OBEPLA-RR gets a new value y_i at time tick i, it calculates the allowed subrange between $y_i - \varepsilon$ and $y_i + \varepsilon$, and find all black dots in the subrange according to the resolution r. Then, for each black dot v, it invokes back_swing_rr (v) to calculate the cost of the shortest path from any black dot at time tick 1, and then record the start point of the last segment. We note that the cost of all black dots at time tick 1 is initialized to 0. When all data are processed, OBEPLA-RR identifies the black dot at the last time tick, $i = n$, with minimal cost, $best_i$, and the line segment from $best_i$.from to $best_i$. Recursively, OBEPLA-RR finds a set of line segments, which forms an optimal BEPLA.

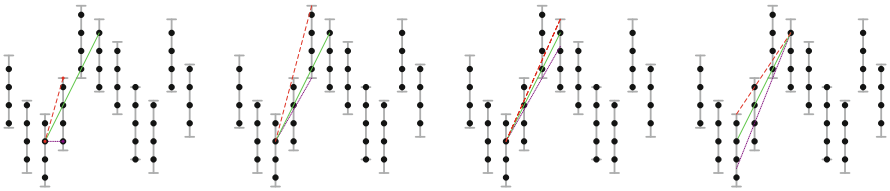


Fig. 3. The two endpoints of a feasible line segment must be visible to each other (color figure online).

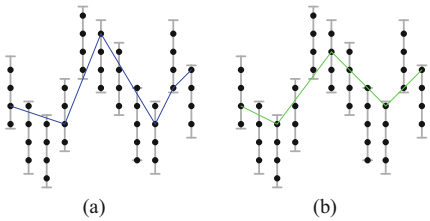


Fig. 4. Greedy Swing-RR doesn't always find optimal BEPLA.

Function back_swing_rr (v) scans all black dots visible to v, one tick backward by one tick, until no more ticks to scan or the line segment length reaches the limit. Similarly to SwingFilter and SwingRR, it maintains two auxiliary lines to form a fan area visible to v. (u, v) is upper bounder of the visible fan, and (l, v) is lower bounder of the visible fan. When there are black dots at tick pt between the two bounders, (l, v) and (u, v), it tries to relax $v.cost$ by Eq. (8), and the same time, it also records the start point of the line segment at $v.from$ (Fig. 5).

Since there are $\lfloor 2^{r+1}\varepsilon \rfloor$ or so black dots in any time tick, and back_swing_rr () backward scans at most 2^d time ticks, the complexity of back_swing_rr () is $O(\lfloor 2^{r+1}\varepsilon \rfloor * 2^d) = O(2^{r+d}\varepsilon)$. Thus, the complexity of OBEPLA-RR is $O(2^{r+d}\varepsilon n)$.

Algorithm 1: OBEPLA-RR, given error bound ε, resolution r, and maximal delay $delay = 2^d$

```
1:  i =1; y_i = get_value()                    // i records the tick, and get the first value
2:  best_i = any black dot at tick 1           // best_i records the black dot whose cost is minimal at tick i
3:  foreach black dot v at tick i whose value is between y_i − ε and y_i + ε
4:      v.cost = 0; v.from = NIL
5:  end
6:  while more values do
7:      i ++; y_i = get_value()                // get next value
8:      best_i.cost = ∞
9:      foreach black dot v at tick i whose value is between y_i − ε and y_i + ε
10:         back_swing_rr( v )                 // find v.cost and v.from where the last line segment starts
11:         if (v.cost < best_i.cost) then best_i = v
12:     end foreach
13: end while
14: trace_back ( best_i )                      // find the segment (best_i.from, best_i) recursively and form a BEPLA
15:
16: Function back_swing_rr( v )
17: Begin
18:     v.cost = ∞                             // v.cost is initialized to an impossible big number
19:     pt = v.tick − 1; length = 1            // pt is the tick for backward scan
20:     while pt ≥ 1 and length ≤ delay do
21:         When necessarily, swing up     the lower bounder (l, v) of the visible fan from v
22:         When necessarily, swing down the upper bounder (u, v) of the visible fan from v
23:         if u is below (l, v) then break    // no more backward
24:         foreach black dot w at tick pt between the two lines (l, v) and (u, v)
25:             if w.cost + 1 < v.cost then
26:                 v.cost = w.cost + 1        // relaxation
27:                 v.from = w
28:             end if
29:         end foreach
30:         pt --; length ++                   // backward one more tick
31:     end while
32: End
```

Fig. 5. Shows an example how OBEPLA and back_swing_rr () run.

4 Experiments

Similarly to the aforementioned studies, in this study, we use the UCR time series database [18] to evaluate the performance of OBEPLA-RR, and compare it with state-of-the-art methods, including Swing-RR [11] for connected BEPLA, OptimalPLR [5] for disconnected BEPLA, and Mixed-PLA [10] for mixed BEPLA (Fig. 6).

Table 1 shows the preliminary experiment results on eight datasets. The error bound is 5%. For Swing-RR and OBEPLA-RR, minimal resolution, $r_{\min}$ (5%) = 4, is used. Thus, the block size is $1/2^4 = 6.25\%$, which is smaller than 2*5% = 10%. This ensures that at least one encoded level, i.e., black dot, in any allowed subrange. As well, an octet is used to encode the segment length, i.e., $d = 8$, and maximal delay is 256 time ticks.

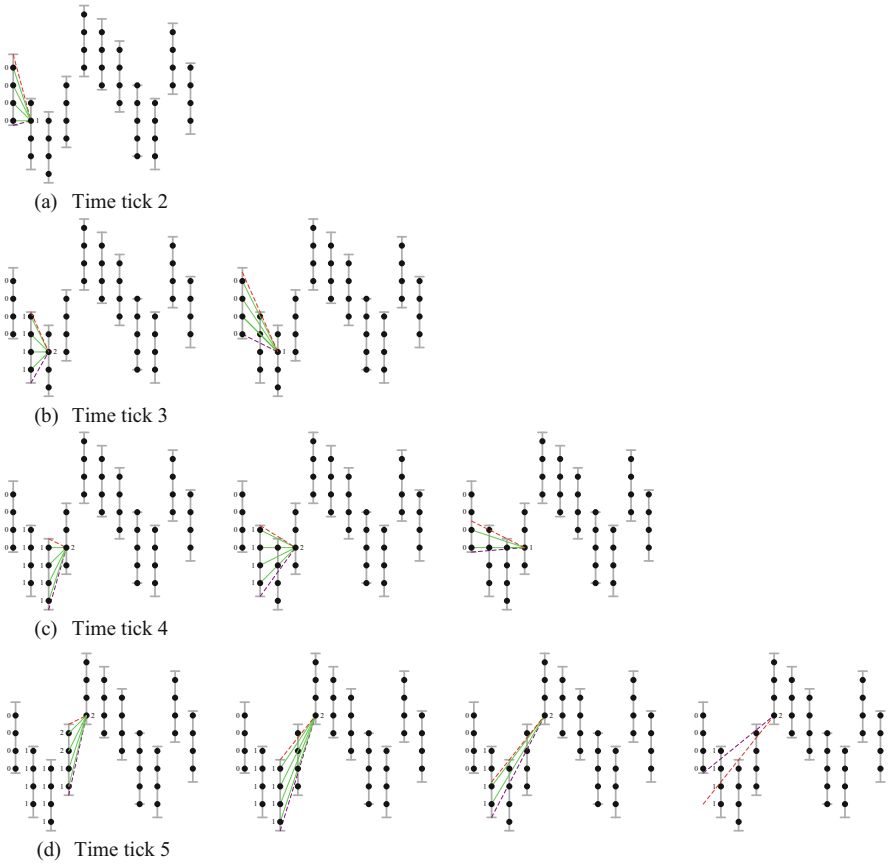


Fig. 6. Illustration of the execution of OBEPLA-RR and the function back_swing_rr ()

The compression ratios of Mixed-PLA on all eight datasets is consistently better than OptimalPLA. When Resolution Reduction is considered, and when minimal resolution is used, there are only one or two black dots in the allowed subrange. The choice of line segment endpoints is significantly restricted. Swing-RR and OBEPLA-RR use more line segments than OptimalPLR. However, they both use less bits than OptimalPLR, and even Mixed-PLA.

It is worthy to note that when minimal resolution is used, $\lfloor 2^{r+1}\varepsilon \rfloor$ is very small, and only a very small number of black dots at a time tick, the complexity of OBEPLA-RR is reduced to $O(\ 2^d n)$.

Table 1. Eight datasets from UCR time series database are approximated by multiple line segments, i.e., piecewise linear functions, under maximal 5% error bound.

Dataset	Length	Number of Segments and Compression Ratio						
		Mixed-PLA[1] (Mixed) (Optimal)	OptimalPLR[2] (Disconnected) (Optimal)		Swing-RR[3] (Connected) (Non-Optimal)		OBEPLA-RR[4] (Connected) (Optimal)	
Cricket_X	117,000	2.2%	2,064	5.29%	6,046	1.94%	4,252	1.36%
Cricket_Y	117,000	2.0%	2,015	5.17%	6,282	2.01%	4,274	1.37%
Cricket_Z	117,000	2.1%	2,053	5.26%	5,959	1.91%	4,242	1.36%
FaceFour	30800	7.1%	1,876	18.27%	2,963	3.61%	2,090	2.55%
Lighting2	38220	1.0%	339	2.66%	837	0.82%	562	0.55%
Lighting7	22330	2.8%	357	4.80%	929	1.56%	575	0.97%
MoteStrain	105168	5.7%	4,011	11.44%	8,518	3.04%	6,422	2.29%
wafer	152000	4.3%	3,902	7.70%	7,760	1.91%	5,423	1.34%

[1] The results of Mixed-PLA are retrieved from the work of Zhao et al. [10].
[2] The results of OptimalPLR are retrieved from the work of Xie et al. [5].
[3] Swing-RR [11] in this experiment uses 4 and 8 bits to encode codepoint and length of segments, respectively, and generates connected segments.
[4] OBEPLA-RR in this experiment uses 4 and 8 bits to encode codepoint and length of segments, respectively, and generates connected segments.

5 Conclusions and Future Works

The huge data sensed from IoT devices in many different applications today consume a large percentage of energy. Lossy data compression methods based on Bounded-Error Piecewise Linear Approximation (BEPLA) are commonly used to compress these data, in sensor nodes, edge nodes, and data centers, to reduce the data size to benefit the network transmission between these systems and disk storage usage, while the errors between the original data and the reconstructed data is under a given error bound.

In the literature, many optimal BEPLA algorithms have been proposed to reduce time series [3–11]. Most of them try to use as few line segments as possible to approximate the original data. Swing-RR [11] introduced the concept of Resolution Reduction, where endpoints of line segments are restricted to a small number of data points. Although Swing-RR is not optimal in terms of the number of line segments, and uses more line segments to approximate the original data, it uses significantly fewer bits to encode these line segments, and achieves better data compression ratios.

In this study, we present OBEPLA-RR, an optimal algorithm for BEPLA when Resolution Reduction is considered. The preliminary experiment results show that OBEPLA-RR significantly outperforms Swing-RR, OptimalPLR, and Mixed-PLA on eight real word datasets selected from UCR time series databases.

Currently, OBEPLA-RR finds connected BEPLA. We will further study disconnected and mixed BEPLA when Resolution Reduction is considered. Recent researches indicate that the error bound may vary for different applications [12]. Thus, many BEPLA with different error bounds are required. In some cases, the error bound may adapt with time [20]. We will further study these issues.

Acknowledgments. .

References

1. Shehabi, A., et al.: United States Data Center Energy Usage Report; LBNL-1005775; Lawrence Berkeley National Laboratory: Berkeley. CA, USA (2016)
2. Newmark: 2023 U.S. Data Center Market Overview & Market Clusters. https://www.nmrk.com/insights/market-report/2023-u-s-data-center-market-overview-market-clusters
3. O'Rourke, J.: An on-line algorithm for fitting straight lines between data ranges. Commun. ACM **24**, 574–578 (1981)
4. Elmeleegy, H., Elmagarmid, A.K., Cecchet, E., Aref, W.G., Zwaenepoel, W.: Online piecewise linear approximation of numerical streams with precision guarantees. Proc. VLDB Endow. **2**, 145–156 (2009)
5. Xie, Q., Pang, C., Zhou, X., Zhang, X., Deng, K.: Maximum error-bounded piecewise linear representation for online stream approximation. VLDB J. **23**, 915–937 (2014)
6. Hakimi, S.L., Schmeichel, E.F.: Fitting polygonal functions to a set of points in the plane. CVGIP: Graphical Models and Image Processing **53**, 132–136 (1991)
7. Wang, D., Huang, N., Chao, H., Lee, R.: Plane sweep algorithms for the polynomial approximation problems with applications. Proc. International Symposium on Algorithms and Computation, pp. 515–522 (1993)
8. Goodrich, M.: Efficient piecewise-linear function approximation using the uniform metric. Proc. Annual Symposium on Computational Geometry, 322–331 (1994)
9. Luo, G., et al.: Piecewise linear approximation of streaming time series data with max-error guarantees. Proc. IEEE 31st International Conference on Data Engineering, pp. 173–184. Seoul, Korea (2015)
10. Zhao, H., Li, T., Chen, G., Dong, Z., Bo, M., Pang, C.: An online PLA algorithm with maximum error bound for generating optimal mixed-segments. Int. J. Mach. Learn. Cybern. **11**, 1483–1499 (2020)
11. Lin, J.-W., Liao, S.-W., Leu, F.-Y.: Sensor data compression using bounded error piecewise linear approximation with resolution reduction. Energies **12**, 2523 (2019)
12. Barbarioli, B., Mersy, G., Sintos, S., Krishnan, S.: Hierarchical residual encoding for multiresolution time series compression. Proc. ACM Manage. Data **1**, 99 (2023)
13. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. Cons. Electr. **38**, xviii–xxxiv (1992)
14. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H. 264/AVC video coding standard. IEEE Trans. Circ. Sys. Video Technol. **13**, 560–576 (2003)
15. Chen, Q., Chen, L., Lian, X., Liu, Y., Yu, J.X.: Indexable PLA for efficient similarity search. Proc. International Conference on Very Large Data Bases, pp. 435–446 (2007)
16. Chen, Y., et al.: Classification of short single-lead electrocardiograms (ECGs) for atrial fibrillation detection using piecewise linear spline and XGBoost. Physiol. Meas. **39**, 104006 (2018)
17. Chen, Y., Hao, Y.: A novel framework for stock trading signals forecasting. Soft. Comput. **24**, 12111–12130 (2020)
18. Chen, Y., et al.: The UCR Time Series Classification Archive. http://www.cs.ucr.edu/~eamonn/time_series_data/
19. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C.: Introduction to Algorithms, 4th edition. The MIT Press (2022)
20. Zhou, Z., Baratchi, M., Si, G., Hoos, H.H., Huang, G.: Adaptive error bounded piecewise linear approximation for time-series representation. Eng. Appl. Artif. Intell. **126**, 106892 (2023)