Information Sciences 345 (2016) 1–8

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Segmenting time series with connected lines under maximum error bound

Huanyu Zhao [a,b,c], Zhaowei Dong [d], Tongliang Li [b,*], Xizhao Wang [f], Chaoyi Pang [b,c,e,**]

[a] SJZ JKSS Technology Co. Ltd, Shijiazhuang, China
[b] Institute of Applied Mathematics, Hebei Academy of Sciences, Shijiazhuang, China
[c] Center for Data Management & Intelligent Computing, Zhejiang University (NIT), China
[d] Radio and TV University, Shijiazhuang, China
[e] RMIT University, Melbourne, Australia
[f] Shenzhen University, Guangzhou, China

ARTICLE INFO

Article history:
Received 12 January 2015
Revised 2 September 2015
Accepted 14 September 2015
Available online 9 October 2015

Keywords:
Piecewise linear
Approximation/representation
Time series data
Haar wavelet
Data compression

ABSTRACT

The error-bounded Piecewise Linear Approximation (PLA) is to approximate the stream data by lines such that the approximation error at each point does not exceed a pre-defined error. In this paper, we focus on the version of PLA problem that generates connected lines in the segmentation for smooth approximation. We provide a new linear-time algorithm for the problem that outperform two of the existing methods with less number of connected segments. Our extensive experiments, on both real and synthetic data sets, indicate that our proposed algorithms are practically efficient.

1. Introduction

A time series is a sequence of data points where each data point is associated with a time stamp. As with most computer science problems, how to efficiently and effectively represent such data is challenging. Essentially, approximate representation is one of the most commonly used methods for data pre-processing and querying. There exist many interesting algorithms or strategies for data approximations, including Fourier Transforms [10], Discrete Wavelet Transform [8], Symbolic Mapping [11], Piecewise Linear Approximation (PLA) [2,5–7] and Piecewise Aggregate Approximation [4].

Recently, the research on maximum-error bound *Piecewise Linear Approximation* (L_∞-bound PLA) has gained some attention. This representation constructs a number of line segments to approximate the stream such that the approximation error on each corresponding point does not exceed a prescribed error bound (L_∞-norm). Xie et al. [12] give an optimal linear-time algorithm[1] that constructs minimum number of line segments in approximation. In their method, the minimum number of line segments is achieved through maximally extending each constructed segment. The general idea of DisConnAlg follows: in order to adjust a line segment to approximate the maximum number of stream points, the algorithm determines the range of all feasible line segments, which is incrementally maintained during the processing of consecutive sequence points. Whenever the current point

* Corresponding author.
** Corresponding author at: Institute of Applied Mathematics, Hebei Academy of Sciences, Shijiazhuang, China.
E-mail addresses: zhaohuanyu@163.com (H. Zhao), litongliang@tom.com (T. Li), chaoyip@netscape.net (C. Pang).

[1] OptimalPLR algorithm in [12], which is termed as DisConnAlg in this article.

http://dx.doi.org/10.1016/j.ins.2015.09.017

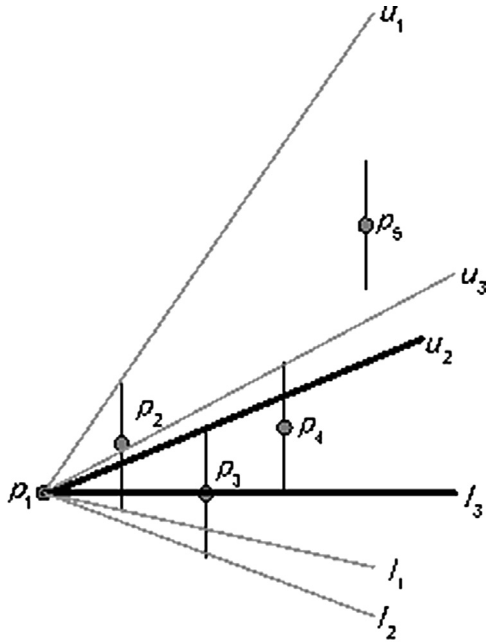

Fig. 1. The process of FSW.

cannot be approximated within the error bound, start a new segment from this point. Furthermore, DisConnAlg can be used to construct connected line segments when used on the restricted feasible space iteratively. That is, constructing the next segment from the feasible space of the last data point of the previous segment.[2] We denote this algorithm that generates connected segments as DConnAlg in this article.

In fact, as an old research problem, there are many algorithms for computing either continuous or discontinuous PLAs under the L_∞ norm, including the original work [14,15] from Bellman and Gluss in the early 1960s. They indicated that this problem can be solved by using dynamic programming method. Other research results on this topic include [17–19]. Paper [17] provided algorithms that only work on special functions of “convex shape”. Paper [18] was about non-connected segmentation. Paper [19] proposed polynomial algorithms. Recently, Liu et al. proposed FSW algorithm [6] that uses the Feasible Space (FS) window method to construct segments from a fixed initial point. Qi et al. [9] extend FS to the polynomial functions in the processing of multidimensional data.

Let $u_i = line(p_1, p_{i+1} + \delta)$ be the line that passes points p_1 and $p_{i+1} + \delta$, and $l_i = line(p_1, p_{i+1} - \delta)$ be the line that passes points p_1 and $p_{i+1} - \delta$. As Fig. 1 shows, Liu’s method first constructs FS to be the area between the lines u_1 and l_1. The feasible space is then incrementally narrowed down to the intersection part of FS and area between of u_i and l_i for the newly arriving points $i + 1$. Continuing the process until the point when FS turns into empty where the next new segment is to be built from this very point iteratively. In the example of Fig. 1, $\{p_1, p_2, p_3, p_4\}$ is approximated by one segment whose FS is the area between u_2 and l_3.

Liu indicated that FSW algorithm outperforms the algorithms of [1,2,13,16] with less number of constructed segments. Liu’s method constructs the FS from the starting point without considering the use of error-tolerant rang $[p_1 - \delta, p_1 + \delta]$ as that of DisConnAlg and DConnAlg. Therefore, the segment constructed by FSW could contain less number of stream points than that of constructed by DisConnAlg or DConnAlg in general. As a result, Liu’s method could output many more segments than that of DisConnAlg or DConnAlg in general.

Our contributions in this article can be summarized as follows:

1. Design and implement ConnSegAlg algorithm. Through incorporating the “Forward-Checking” strategy used in DisConnAlg of [12] and using the “Backward-Checking” strategy, this algorithm has linear time complexity and constructs less number of segments than that of DConnAlg and FSW. Next, we indicate that the number of segments constructed from ConnSegAlg is bounded by $2k - 1$ where k is the optimal number of disconnected segments constructed by DisConnAlg. However, this bound does not hold for DConnAlg and FSW. We indicate that the number of segments constructed by DConnAlg and FSW can be above $2k - 1$ in some situations. Lastly, we show that the $2k - 1$ bound is tight. That is, there exists a stream such that the number of segments constructed from ConnSegAlg equals to $2k - 1$.

[2] Refer to Section 6.3.1 of [12] for details.

Table 1
Notations.

Symbol	Meaning
δ	A given error bound on each data point
$P = (p_1, \ldots, p_k)$	A time series
p_i or (t_i, p_i)	The ith data point in time series
$\underline{p_i}$ or $(t_i, p_i - \delta)$	Data point with deleted tolerant error at t_i
$\overline{p_i}$ or $(t_i, p_i + \delta)$	Data point with added tolerant error at t_i
$line(p_i, p_j)$	Line that passes point p_i and p_j
p_{start}	The starting point of a segment
p_{next}	The next coming data point of a segment
p_{s_i}	The end point of ith segment
t_{s_i}	The end timestamp of ith segment
u_i or $y = u_i(t)$	The extreme line of maximum slope in ith segment
l_i or $y = l_i(t)$	The extreme line of minimum slope in ith segment
$\overline{p_{s_i}}$	The intersection point with maximum extreme line at t_{s_i}
$\underline{p_{s_i}}$	The intersection point with minimum extreme line at t_{s_i}

2. Provide theoretical proofs and explanations for the above mentioned properties. We also indicate that "backward-checking", which satisfies the similar properties like "forward-checking", can be used to update extreme lines without requiring much modification.
3. Conduct extensive experiments on both synthetic and real life data to verify our theoretical conclusions. We compared ConnSegAlg with DConnAlg, DisConnAlg and FSW in terms of processing time and the number of segments constructed. The experimental results show that (1) the number of segments constructed by ConnSegAlg is generally less than that of DConnAlg and FSW; (2) the time efficiency of ConnSegAlg is comparable with that of DisConnAlg and DConnAlg; and (3) the proposed algorithm ConnSegAlg is practically effective and efficient for segmenting online time series.

The rest of the paper is organized as follows: Section 2 explains the idea, the possessed properties and the pseudo code of ConnSegAlg; Section 3 is the experimental results, including the performance comparisons among DisConnAlg, DConnAlg, ConnSegAlg and FSW; Section 4 concludes this paper.

2. Algorithm

In this section, we will introduce our algorithm, ConnSegAlg, to construct minimized number of connected segments. We will first give the outlines of ConnSegAlg. We then provide the theoretical proofs on the claimed properties for the algorithm. Lastly, we present the pseudo code of ConnSegAlg.

The general notations used in this paper are summarized in Table 1, where many of them are adopted from [12].

2.1. Methodology

ConnSegAlg integrates the procedures of DisConnAlg and DConnAlg. Its outline is summarized into the following steps.

Step 1. (Lines 1 and 2 of Algorithm 1). Construct the first segment with DisConnAlg and output this segment in the result. This segment is identical to the first segment constructed by DConnAlg. They have the same boundary (extreme lines) according to the mechanism of DisConnAlg and DConnAlg (refer to Fig. 2(I)). That is, $p_{s_1} = p'_{s_1}$ where p_{s_i} and p'_{s_i} are the end point of ith segment constructed from DisConnAlg and DConnAlg, respectively. Let u_i (u'_i) and l_i (l'_i) be the extreme lines of maximum and minimum slops in the ith segment constructed by DisConnAlg and DConnAlg, respectively. We also have $u_1 = u'_1$ and $l_1 = l'_1$ hold.

Step 2. Backward-checking strategy (Lines 4–9 of Algorithm 1). Construct the next segment with DisConnAlg and see if it can connect to the first one through backward-checking. Similar to the forward-checking that is used in DisConnAlg to intend to cover maximum number of incoming stream points in the process of constructing a segment, we can use backward-checking to decide if these two segments can be connected as indicated in Property 2.1. As described in the top figure of Fig. 2(II), this property intuitively means that the obtained two segments from DisConnAlg can be connected if the boundaries of the two segments have a common area (i.e., the intersection is not empty) at t_{s_1}.

If the backward-checking is successful, we output this segment in the result. Otherwise, the following step needs to be preformed.

Step 3. Length-checking strategy (Lines 11–20 of Algorithm 1). Construct the next two segments by DConnAlg. Let t_{s_i} and t'_{s_i} be the end timestamp of ith segment constructed from DisConnAlg and DConnAlg, respectively. To minimize the number of segments constructed, we need compare t'_{s_3} and t_{s_2}. If $t'_{s_3} \geq t_{s_2}$, we output the second and third segments constructed by DConnAlg in the result (Fig. 3(I)). Otherwise, we output a trivial segment from t_{s_1} to $t_{s_1} + 1$, and the second segment

Algorithm 1 Function ConnSegAlg(P, δ).

Input:
time sequence $P = (p_1, p_2, \ldots, p_n, \ldots)$, error bound δ
Output:
segmenting points$(s_{start}, s_1, s_2, \ldots)$
Description:
Use DisConnAlg to generate the first segment. Set $(t_{start}, ((l_1(t_{start}) + u_1(t_{start}))/2))$ as s_{start}
Initial the number of segments $n = 1$
While(not finished segmenting time series)
Use DisConnAlg to generate the next disconnected segment
if Backwardly check point $p_{s_{(n-1)}}$ according to Property 2.1 **then**
  $n = n + 1$
  Update the extreme lines for the nth and $(n-1)$th segments
  Set $(t_{(n-1)}, (l_{(n-1)}(t_{(n-1)}) + u_{(n-1)}(t_{(n-1)}))/2)$ as $s_{(n-1)}$
  Continue
else
  Use DConnAlg to construct two connected segments
  if $t'_{s_3} \geq t_{s_2}$ **then**
    $n = n + 2$
    Continue;
  else
    $n = n + 2$
    Add trivial segment
    Continue;
  end if
  Set $(t_{(n-1)}, (l_{(n-1)}(t_{(n-1)}) + u_{(n-1)}(t_{(n-1)}))/2)$ and $(t_{(n-2)}, (l_{(n-2)}(t_{(n-2)}) + u_{(n-2)}(t_{(n-2)}))/2)$ as $s_{(n-1)}$ and $s_{(n-2)}$, respectively
end if

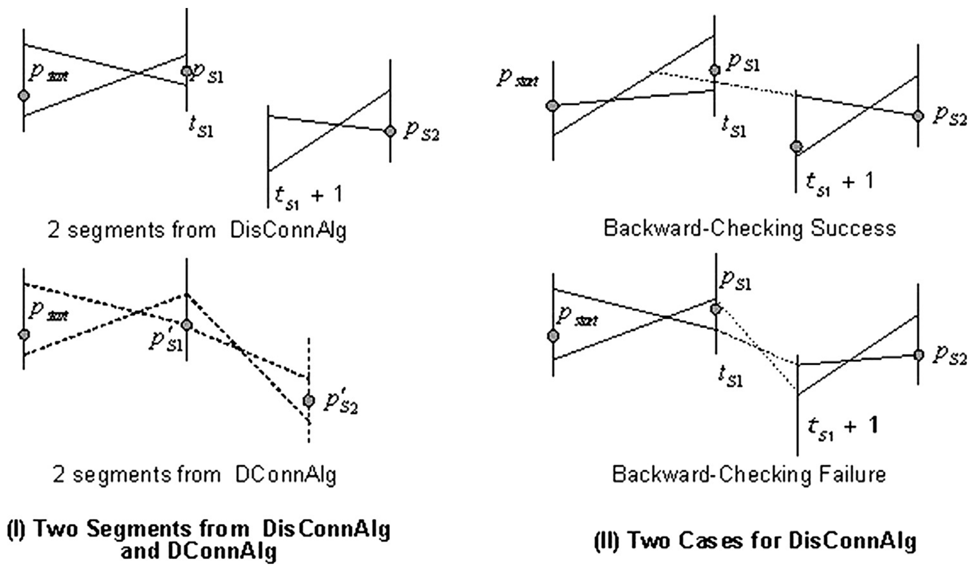

Fig. 2. The basic algorithm.

constructed from DisConnAlg in the result (Fig. 3(II)). This is because that we could always add a trivial segment from t_{s_1} to $t_{s_1} + 1$ to make the first two segments constructed from DisConnAlg connected.

Step 4. Repeat Step 2 for the last outputted segment until all the data points are processed.

Based on the above discussion, we give the pseudo code of ConnSegAlg in Algorithm 1.

2.2. Properties

Structurally, ConnSegAlg is very similar to DisConnAlg and DConnAlg except the backward-checking the length-checking strategies which use a bounded number of unit time. Therefore, the time complexity of our algorithm is $O(n)$.

The following property is used for backward-checking. Its validity is directly drawn from the definitions of extreme lines of [12].

Property 2.1. Let u_i and l_i be the extreme lines of maximum and minimum slop in the ith segment constructed by DisConnAlg, respectively. Then the ith segment and the $i + 1$th segment of DisConnAlg can be connected if and only if the two intervals of

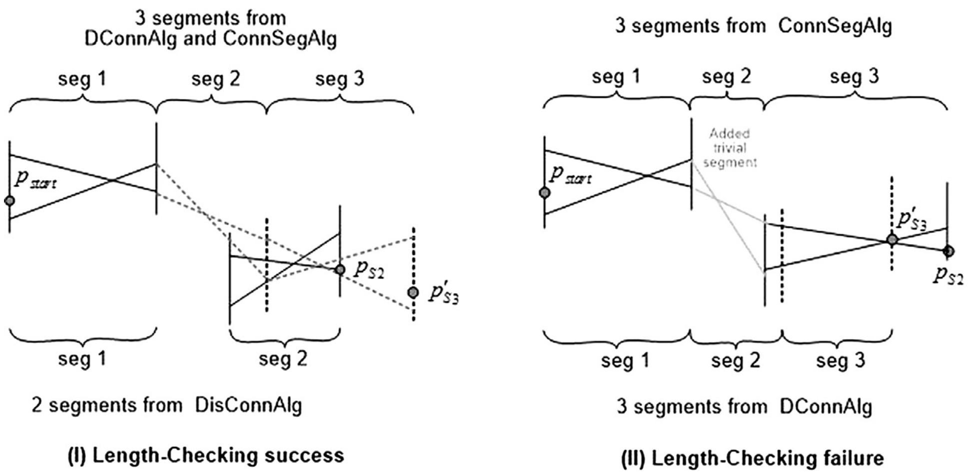

Fig. 3. The length-checking strategy.

$(l_i(t_{s_i}), u_i(t_{s_i}))$ and $(u_{i+1}(t_{s_i}), l_{i+1}(t_{s_i}))$ are intersected. That is,

$$(l_i(t_{s_i}), u_i(t_{s_i})) \cap (u_{i+1}(t_{s_i}), l_{i+1}(t_{s_i})) \neq \emptyset. \tag{2.1}$$

Next, we measure the number of constructed segments from ConnSegAlg in terms of the number from DisConnAlg.

Theorem 2.1. *Given a time series and an error bound δ, let the number of constructed segments from DisConnAlg be k. Then, for the given time series and an error bound, the number of constructed segments from ConnSegAlg h satisfies*

$$k \leq h \leq 2k - 1. \tag{2.2}$$

Proof. Clearly, Formula 2.2 holds for $k = 1$. Assume Formula 2.2 holds for $k = m$. That is, $m \leq h' \leq 2m - 1$

In the case of $k = m + 1$, $m + 1 \leq h$ holds as DisConnAlg is optimal that constructs the least number of segments. On the other hand, $h \leq h' + 2$ holds as we can always add a trivial segment to make two segments being connected via the trivial segment (refer to the bottom figure in Fig. 3(II)). According the inductive hypothesis,

$$h \leq h' + 2 \leq 2m - 1 + 2 \leq 2(m + 1) - 1 \leq 2k - 1.$$

In conclusion, $k \leq h \leq 2k - 1$ holds. Thus the Formula 2.2 is proven inductively. □

Theorem 2.1 ensures the output quality of ConnSegAlg which does not met by DConnAlg. In Section 3, we also indicated that the upper bound $2k - 1$ is strict.

3. Experiments

To verify the characteristics of our algorithm, we provide extensive experimental comparisons against DisConnAlg, DConnAlg, ConnSegAlg and FSW on a wide range of synthetic and real data sets.

For the real data sets, we adopt the 43 data sets from the UCR time series archive [3]. These data sets have been widely used for evaluating time series algorithms.

We construct two synthetic data sets specially designed for the situations where all backward checking are successful and failed, respectively. In the first data set, we generate a 10^5 sized disconnected time series (called DisConnect-Series) by the linear function $y = x - 10 * (i - 1) - 1$ with 10 points, where $0 < i \leq 10^4$ and $\forall i, x \in [10 * i - 9, 10 * i]$. In the second data set, we generate a $1.9{*}10^5$ time series called Connect-Series by two consecutive linear function $y = x - 18 * (i - 1) - 1$ and $y = -x + 18 * i + 1$ with 9 points on each function in a iteration, where $0 < i \leq 10^4$, $\forall i$ $x \in [18 * i - 17, 18 * i - 9]$ for the former function and $x \in [18 * i - 8, 18 * i]$ for the former one.

The original source codes are obtained from the authors of [9,12] . All algorithms, ConnSegAlg, DisConnAlg, DConnAlg and FSW are implemented in Eclipse with C++. The test experiments are conducted on a PC with CPU of Intel Core 3.20GHz and 16G memory. We have tested all the data sets on the error bound of 2.5 %, 5 % and 10 % of the value range. Due to the similarities, we only show the results on the situation of 2.5 %. All the tested results are listed in Tables 2 and 3.

Regarding to the number of constructed segments, the results listed in Table 2 confirm the bound on the number of constructed segments from our algorithm. That is, the number of constructed segments from ConnSegAlg is within $[k, 2k{-}1]$ on all tested data sets and mostly much less than $2k - 1$. However, DConnAlg and FSW do not satisfy this bound and the number of constructed segments from DConnAlg can be more than 10% of ours sometimes. Furthermore, the number of constructed segments from ConnSegAlg is less than that from FSW in general.

Table 2
Tests on the real data sets.

Data set	Length	Number of segment				Processing time (ms)			
		DConn	DisConn/2k−1	Conn	FSW	DConn	DisConn	Conn	FSW
50words	123,305	3253	1625/3249	2828	3541	3698	3283	3157	30
Adiac	69,207	2281	1532/3063	2277	2293	2101	1777	1492	20
Beef	14,131	91	61/121	90	95	455	382	379	1
Coffee	8037	673	476/951	646	739	227	184	159	1
OliveOil	17,131	581	391/781	538	617	518	441	341	10
CBF	116,100	69,652	40,183/80,365	69273	76,893	1684	946	**1033**	50
ECG200	9700	2308	1376/2751	2195	2761	241	179	168	1
FaceAll	223,081	44,681	29,994/59,987	42,541	49,636	5597	4204	3826	60
FaceFour	30,888	5916	3632/7263	5670	7230	780	608	557	10
FISH	81,201	1119	854/1707	1084	1189	2445	2225	1639	20
Gun-Point	22,650	1270	983/1965	1260	1370	665	550	410	10
Lighting2	38,918	1793	866/1731	1483	2219	1123	945	936	10
Lighting7	23,360	2119	1175/2349	1973	2615	650	544	540	10
OSULeaf	103,576	5289	3795/7589	5052	5598	2982	2521	1955	20
SwedishLeaf	80,626	6434	4415/8829	6200	6941	2282	1885	1542	10
synthetic-c-ontrol	18,301	11,695	6707/13,413	11626	12792	245	135	**142**	10
Trace	27,600	1080	627/1253	992	1290	852	700	536	1
Two-Patter-ns	516,000	226,095	129,045/258,089	224,964	246,252	10,149	6868	**7335**	170
yoga	1,281,000	49,531	36,078/72,155	47,730	52,887	37,420	31,901	24,761	300
wafer	943,092	64540	38309/76617	63726	66900	27685	23107	20736	230
ChlorineCo	641,281	124,413	62,393/124,785	113,284	142,499	16,236	13,353	**13,491**	175
Cricket-X	117,391	6722	3672/7343	6005	8049	3364	2809	2755	30
Cricket-Y	117,391	7659	3549/7097	5765	7659	3396	2868	2698	30
Cricket-Z	117,391	6738	3669/7337	5983	8065	3358	2802	2696	30
DiatomSize	105,877	2523	2045/4089	2500	2675	3147	2770	2133	30
ECGFiveDay-s	117,958	10,595	7419/14,837	10,417	11,542	3318	2669	2322	30
Haptics	336,645	5650	3028/6055	5188	6150	9960	8563	7486	70
InlineSkate	1035650	7794	4240/8479	6616	10,230	30672	26,546	23,750	220
MedicalIma-ges	76,000	7450	4633/9265	7068	8224	2155	1734	1575	20
MoteStrain	106,420	12,230	7486/14971	11,945	13,347	2926	2387	2207	30
SonyAIBOR-e	42,671	15,926	10,583/21,165	15,604	18,148	876	555	541	10
SonyAIBOR-eII	62,899	27,051	17,088/34,175	26,592	30,477	1215	747	**748**	20
Symbols	397,006	10,568	7174/14,347	10,245	11,105	11,832	10,216	8314	90
TwoLeadEC-G	94,537	14,126	9510/19,019	13,859	15,576	2441	1955	1806	30
WordsSyno-nyms	172,,898	6099	4124/8247	5883	6361	5137	4387	4004	40
CinC-ECG-t-orso	2,263,201	21181	12001/24001	17,509	24,343	67,106	58,157	51,027	514
FacesUCR	270,601	54,370	37,101/74,201	52,250	60,427	6794	5019	4688	72
ItalyPower	25,726	11,610	7376/14751	11,494	12,769	479	285	**288**	10
MALLAT	2,403,626	77,637	51,763/103,525	72,900	81,401	70,407	62,472	47,341	540
StarLightCu-rves	8,441,901	93,008	58,599/117,197	85,024	98,145	257,568	237,367	191,292	1905
uWaveGest–X	1131913	39341	25709/51417	37005	41651	33540	28175	24049	266
uWaveGest–Y	1131913	40381	25396/50791	37543	42859	33893	28349	24895	266
uWaveGest–Z	1131913	41463	27169/54337	39159	43914	33746	28176	23752	266

Table 3
Tests on two synthetic data sets.

Data set	Number of segment	
	DisConnAlg	ConnSegAlg
Connected-series	10,000	19,999
DisConnected-series	38,000	38,000

Although the time complexities of three algorithms are $O(n)$, their actual running time are different. As showed in Table 2, the time cost of ConnSegAlg is less than that of DConnAlg on all the tested situations and outperforms DisConnAlg on most situations except the 6 bold cases denoted. The time cost of FSW is much less than that of ConnSegAlg even through both algorithms have line time complexity. Furthermore, also indicated in Table 2, our proposed algorithm scales well on bigger data size as it based on DisConnAlg and DConnAlg. That is, the consuming time will grow in a similar scale of the increased data size.

The test results on synthetic data sets are listed in Table 3. ConnSegAlg constructs an optimal result and outputs the minimum number of segments if all backward-checking are successful as the number of constructed segments from ConnSegAlg equals to that of DisConnAlg. In contrast, the number of constructed segments from ConnSegAlg reaches the upper bound $2k-1$ where k is the number of constructed segments by DisConnAlg. These results also indicate that the upper bound $2k-1$ is tight. Through

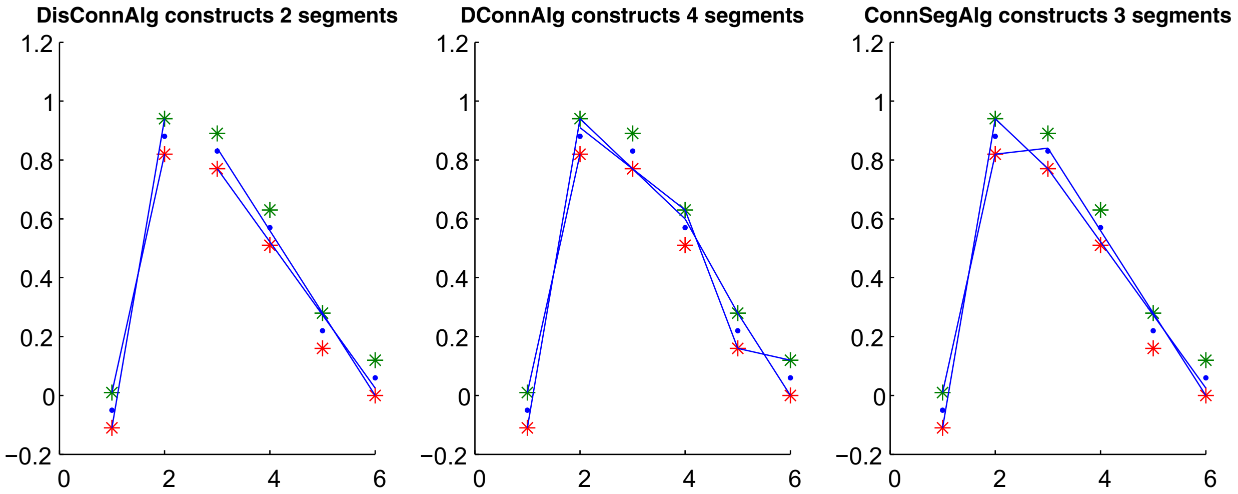

Fig. 4. The result of Example 1.

referring the test results of Table 2 and the following example, it can be seen that the upper bound $2k - 1$ does not hold for DConnAlg yet.

The following example indicates that DConnAlg can generate more segments than that of ConnSegAlg in the situation of “backward-checking” failure and “Length-Checking” adopted (refer to Fig. 4).

Example 1. Give a time series

$$P = (-0.05, 0.88, 0.83, 0.57, 0.22, 0.06)$$

and error bound $\delta = 0.06$. The number of segments constructed from DisConnAlg, DConnAlg and ConnSegAlg are 2, 4 and 3, respectively.

4. Conclusions and future works

In this paper, we propose a new linear time algorithm to generate connected lines with guaranteed error bound for PLA. Comparing with the existing two approaches, our proposed algorithm guarantees to generate less number of segments and is practically efficient. Extensive experiments on both real and synthetic data sets indicate that our algorithm has better performance than the existing algorithms. Our future work would consider how to design an optimal algorithm that constructs minimum number of connected segments for PLA.

Acknowledgments

This work was partially supported by the Hebei “One Hundred Plan” Project (no. E2012100006), Hebei Academy of Sciences Project (no. 15606) and Natural Science Foundation of China (grant no. 61572022). The authors would like to thank the authors of [9,12] for their generous giving us their source code for the comparison tests in this paper.

References

[1] U. Appel, A.V. Brandt, Adaptive sequential segmentation of piecewise stationary time series., Inf. Sci. 29 (1) (1983) 27–56.
[2] E. Keogh, S. Chu, D. Hart, M. Pazzani, Segmenting time series: A survey and novel approach., Data Mining in Time Series Databases, World Scientific, 2003, pp. 1–22.
[3] E. Keogh, X. Xi, L. Wei, C. Ratanamahatana, The UCR Time Series Classification/Clustering Homepage, 2006.
[4] E.J. Keogh, M.J. Pazzani, Scaling up dynamic time warping to massive datasets, Principles of Data Mining and Knowledge Discovery, Lecture Notes in Computer Science, 17041999, pp. 1–11.
[5] A. Koski, M. Juhola, M. Meriste, Syntactic recognition of ECG signals by attributed finite automata, Pattern Recognit. 28 (12) (1995) 1927–1940.
[6] X. Liu, Z. Lin, H. Wang, Novel online methods for time series segmentation, IEEE Trans. Knowl. Data Eng. 20 (12) (2008) 1616–1626.
[7] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, Streaming time series summarization using user-defined amnesic functions, IEEE Trans. Knowl. Data Eng. 20 (7) (2008) 992–1006.
[8] C. Pang, Q. Zhang, X. Zhou, D.P. Hansen, S. Wang, A.J. Maeder, Computing unrestricted synopses under maximum error bound, Algorithmica 65 (1) (2013) 1–42.
[9] J. Qi, R. Zhang, K. Ramamohanarao, H. Wang, Z. Wen, D. Wu, Indexable online time series segmentation with error bound guarantee, World Wide Web (2013) 1–43.
[10] D. Rafiei, A. Mendelzon, Efficient retrieval of similar time sequences using dft, in: Proceedings of the 5th International Conference on Foundations of Data Organization and Algorithms (FODO), Kobe, 1998, pp. 249–257.
[11] C. Perng, H. Wang, S.R. Zhang, D.S. Parker, Landmarks: a new model for similarity-based pattern querying in time series databases, in: Proceedings of the 16th International Conference on Data Engineering(ICDE), 2000, pp. 33–42.

[12] Q. Xie, C. Pang, X. Zhou, X. Zhang, K. Deng, Maximum error-bounded piecewise linear representation for online stream approximation, VLDB J. (2014) 1–23.
[13] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, W. Truppel, Online amnesic approximation of streaming time series, in: Proceeding of the 20th International Conference on Data Engineering, 2004, pp. 339–349.
[14] R. Bellman, On the approximation of curves by line segments using dynamic programming, Commun. ACM 284 (6) (1961).
[15] B. Gluss, Further remarks on line segment curve-fitting using dynamic programming, Commun. ACM 5 (8) (1962) 441–443.
[16] E. Keogh, S. Chu, D. Hart, M. Pazzani, An online algorithm for segmenting time series, in: Proceedings of the 2001 IEEE International Conference on Data Mining, 2001, pp. 289–296.
[17] G.M. Phillips, Algorithms for piecewise straight line approximations, Comput. J 11 (2) (1968) 211–212.
[18] D.G. Wilson, Piecewise linear approximations of fewest line segments, in: Proceedings of the 1972 Spring Joint Computer Conference on AFIPS, 40, 1972, pp. 187–198.
[19] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Can. Cartogr. 10 (2) (1973) 121–122.