

A TIME SERIES DIMENSIONALITY REDUCTION METHOD FOR MAXIMUM DEVIATION REDUCTION AND SIMILARITY SEARCH

RUIDONG XUE

Doctor of Philosophy

ASTON UNIVERSITY

December 2021

© Ruidong Xue, 2021

Ruidong Xue asserts his moral right to be identified as the author of this thesis.

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright belongs to its author and that no quotation from the thesis and no information derived from it may be published without appropriate permission or acknowledgement.

Abstract

ASTON UNIVERSITY
A TIME SERIES DIMENSIONALITY REDUCTION METHOD FOR MAXIMUM
DEVIATION REDUCTION AND SIMILARITY SEARCH

Ruidong Xue

Doctor of Philosophy

December 2021

Similarity search over time series is essential in many applications. However, it may cause a “dimensionality curse” due to the high dimensionality of time series. Various dimensionality reduction methods have been developed. Some of them sacrifice max deviation to get a faster dimensionality reduction, such as equal-length segment dimensionality reduction methods: Piecewise Linear Approximation (*PLA*), Piecewise Aggregate Approximation (*PAA*), Chebyshev Polynomials (*CHEBY*), Piecewise Aggregate Approximation Lagrangian Multipliers (*PAALM*) and Symbolic Aggregate Approximation (*SAX*). Some sacrifice dimensionality reduction time for the smallest max deviation, such as adaptive-length segment dimensionality reduction methods: Adaptive Piecewise Linear Approximation (*APLA*) and Adaptive Piecewise Constant Approximation (*APCA*). *APLA* uses a guaranteed upper bound for the best max deviation with slow dimensionality reduction time.

We investigate the existing basic dimensionality reduction techniques for time series data. We point out the limitations of the existing dimensionality reduction techniques and evaluate the dimensionality reduction techniques on real-life datasets. We also conduct preliminary research and make some improvements to the existing dimensionality reduction techniques. Our experimental

results on several datasets compare and verify the efficiency and effectiveness of the existing techniques, including several dimensionality reduction techniques, one index-building method, and two k -Nearest Neighbours (k -NN) search methods.

We propose an adaptive-length segment dimensionality reduction method called Self Adaptive Piecewise Linear Approximation (*SAPLA*). It consists of 1) initialization, 2) split & merge iteration, and 3) segment endpoint movement iteration. Increment area, reconstruction area, conditional upper bound and several equations are applied to prune redundant computations. The experiments show this method can speed up about n (time series length) times than *APLA* when reducing the dimensionality of the original time series with minor max deviation loss. We propose a lower bound distance measure between two time series to guarantee no false dismissals and tightness for adaptive-length segment dimensionality reduction methods. When mapping time series into a tree index, we propose Distance Based Covering with Convex Hull Tree (DBCH-tree) and split node and pick branch by using the proposed lower bounding distance, not waste area. Our experiments show that DBCH-tree helps improve k -NN search over time series using dimensionality reduction methods.

Dedication

To my parents.

Acknowledgements

First, great gratitude is given to my supervisors, Dr. Hongxia (Helen) and Dr. Weiren Yu, for providing me with invaluable direction and strength throughout my research work. Working with them during my research adventures has been an absolute honour and an impressive experience. I am grateful for them to help me earn confidence in myself to find my way in the end. They have always trained me to persist in being the best version of myself. Nothing in this small space can express my gratitude and admiration for them.

I will always be grateful to researchers who have had a hand in my academic growth. Great gratitude is given to my final PhD examiners, Prof. Tony Bagnall and Dr. Felipe Campelo, for helping at my viva examination and providing fruitful comments on my research. Their guidance has supported me in achieving more than I ever could have expected for this thesis. This thesis would look very different without their guidance. I want to thank Prof. Aniko Ekart for her professional academic administration and irreplaceable assistance in making it possible for me to complete this thesis. I am also thankful to Dr. Maria Chli for her time and advice for this thesis. I would like to thank Prof. Jiajin Le for his support during my master's and PhD studies. He has led me to believe in my own abilities.

I want to thank Aston University for granting me the scholarship and financial support. I am also thankful to the research and administrative staff at Aston University for their professional advice and support during my PhD study to make this research possible. Getting to this moment would have been impossible without my friends' and colleagues' understanding, support and helpful discussions in and outside Aston University.

Finally, I am grateful for the endless support and love of my parents and other family members. Especially when I have been in a hole and finishing this thesis seemed like an impossible endeavour at times. They helped me believe in myself and realise that I can achieve anything I set my mind to. I could not have achieved as much as I have done in my life without my parents cheering me on.

Declaration

The thesis entitled ‘A Time Series Dimensionality Reduction Method for Maximum Deviation Reduction and Similarity Search’ submitted for the degree of PhD in Computer Science, has been composed by myself and has not been presented or accepted in any previous application for a degree. All sources of information have been acknowledged by means of references. Portions of the work presented in this thesis have been published at the following conference.

[69] R. Xue, W. Yu, and H. Wang. An indexable time series dimensionality reduction method for maximum deviation reduction and similarity search. In J. Stoyanovich, J. Teubner, P. Guagliardo, M. Nikolic, A. Pieris, J. Mühlig, F. Özcan, S. Schelter, H. V. Jagadish, and M. Zhang, editors, *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, pages 2:183–2:195. OpenProceedings.org, 2022. doi: 10.48786/edbt.2022.08. URL <https://dx.doi.org/10.48786/edbt.2022.08>.

Ruidong Xue

31 December 2021

Contents

1	Introduction	18
1.1	Background	18
1.2	Related Work	19
1.3	Motivation	21
1.4	Contributions	22
1.5	Limitations	23
1.6	Thesis Outline	24
2	Literature Review	25
2.1	Definition & Lemma	25
2.2	Relevant Dimensionality Reduction Methods [62]	28
2.2.1	Piecewise Aggregate Approximation (<i>PAA</i>) [39]	28
2.2.2	Adaptive Piecewise Constant Approximation (<i>APCA</i>) [40]	31
2.2.3	Piecewise Linear Approximation (<i>PLA</i>) [15]	34
2.2.4	Adaptive Piecewise-Linear Approximation (<i>APLA</i>) [48]	35
2.2.5	Chebyshev Polynomials Approximation (<i>CHEBY</i>) [10]	39
2.2.6	Piecewise Aggregate Approximation Lagrangian Multipliers (<i>PAALM</i>) [61]	43
2.2.7	Symbolic Aggregate Approximation (<i>SAX</i>) [45]	46
2.3	Lower Bounding Distance Measures for Dimensionality Reduction Methods	49
2.4	Index Structure & k -NN Search	53
2.4.1	R-tree Indexing & k -NN Search for <i>APCA</i>	53
2.4.2	R-tree Indexing & k -NN Search for <i>PLA</i>	57
2.4.3	R-tree Indexing & k -NN Search for <i>CHEBY</i>	60

2.5	Analysis of Dimensionality Reduction Methods & R-tree Index Structure & k -NN	
	Search Method	61
2.5.1	Analysis of <i>PAA</i> Dimensionality Reduction Method	61
2.5.2	Analysis of <i>APCA</i> Dimensionality Reduction Method	63
2.5.3	Analysis of <i>PLA</i> Dimensionality Reduction Method	64
2.5.4	Analysis of <i>APLA</i> Dimensionality Reduction Method	64
2.5.5	Analysis of <i>CHEBY</i> Dimensionality Reduction Method	68
2.5.6	Analysis of <i>SAX</i> Dimensionality Reduction Method	69
2.5.7	Analysis of Lower Bound Distance Measurements	69
2.5.8	Analysis of R-tree Index Structure	70
2.5.9	Analysis of k -NN Search Method	70
2.6	Conclusion	70
3	Dimensionality Reduction Method <i>SAPLA</i>	72
3.1	Overview	72
	3.1.1 Motivation	73
3.2	Related Work	73
3.3	Preliminaries	75
3.4	Self Adaptive Piecewise Linear Approximation (<i>SAPLA</i>)	75
	3.4.1 Proposed Equations and Area Computation	75
	3.4.2 Initialization	87
	3.4.3 Split & Merge Iteration	87
	3.4.4 Segment Endpoint Movement Iteration	93
	3.4.5 Time Complexity Analysis.	98
3.5	Experimental Evaluation	99
	3.5.1 Datasets in this thesis	99
	3.5.2 Comparison on Max Deviation (θ)	103
	3.5.3 Comparison on Dimensionality Reduction Time	104
4	Lower Bound Distance Measure & Index Structure for <i>SAPLA</i>	110
4.1	Overview	110
	4.1.1 Motivation	110
4.2	Lower Bounding Distance Measure $Dist_{PAR}$	111

4.3	Lower Bounding Lemma for $Dist_{PAR}$	112
4.4	Proof of $Dist_{PAR}$ Tightness	112
4.5	DBCH-tree	116
4.6	Node Splitting and Branch Picking	117
4.7	Experimental Evaluation	121
4.7.1	Comparison on Tightness of Lower Bound Distance	121
4.7.2	Comparison on Pruning Power (ρ)	126
4.7.3	Comparison on k -NN Time	133
5	Conclusion	141
5.1	Summaries	141
5.2	Limitations and Future Work	143
	Appendices	145
A	Detail Experiment Report for Each Dataset	146

List of Figures

2.1	The Euclidean distance can be visualized as the square root of the sum of the squared lengths of the black lines.	26
2.2	The maximum deviation can be visualized as the black line.	27
2.3	Categorization of dimensionality reduction methods including time reduction and frequency reduction. The grey colour rectangle ■ is what this thesis compares.	29
2.4	This figure is a visual illustration of the equal-length segment dimensionality reduction method <i>PAA</i>	30
2.5	This figure is a visual illustration of adaptive-length segment dimensionality reduction method <i>APCA</i>	32
2.6	The x -axis represents the time, and the y -axis represents the real value, the orange line represents represented line segments, the equation of every line segment is $y = a * t + b$	34
2.7	An illustration of data reduction method <i>APLA</i> , which is the adaptive-length segment approach.	37
2.8	The x -axis represents the time, and the y -axis represents the real value. The orange line represents the reconstruction of Chebyshev coefficients.	40
2.9	[61] One example of <i>PAALM</i>	45
2.10	[46] The <i>PAA</i> can be regarded as an attempt to model an original time series with a linear combination of box basis functions. The length of the original time series is 128. The segment number is eight.	46
2.11	[46] One example of <i>SAX</i>	47
2.12	The $Dist_{AE}$ measure can be visualized as the Euclidean distance between Q and the reconstructed time series of \hat{C}	51
2.13	\hat{Q} is obtained by projecting the endpoints of \hat{C} onto Q and calculating the mean values of the sections falling within the projected lines.	52
2.14	R-tree index structure.	54
2.15	An R-tree with 2-D rectangles.	55
2.16	An R-tree with 3-D points.	56
2.17	The white colour circle ○ is the original time series point. The length of time series is twenty. The reduced dimension is four. × is a reconstructed time series by <i>SAPLA</i> representation. (a) shows an example of one time series <i>MBR</i> . It is consist of 4 gray color rectangles ■s, denoted as $G = \{G_i[0], G_i[1], G_i[2], G_i[3]\}$, ($0 \leq i \leq 3$). (b) shows an example of distance between one time series and <i>MBR</i> , denoted as $Dist_{MBR}$	57
2.18	Three cases: line Segment A_1A_2 is completely in the third quadrant, partially in the first and third quadrants, or completely in the first quadrants. Case1.3 and Case3.3 miss special cases, as Figure 2.21 shows.	59

2.19 This figure is the first example of the *PAA* remainder assignment. 61

2.20 This figure is the second example of the *PAA* remainder assignment. 62

2.21 For case 1.3, when $|A_1A_2|$ is too short, the condition $u_{A1} \leq u_{B2} \leq u_{A2}$ is wrong. 66

2.22 For case 3.3, when $|A_1A_2|$ is too short, the condition $u_{A1} \leq u_{B1} \leq u_{A2}$ is wrong. 67

3.1 A visual comparison of time series dimensionality reduction methods. \circ is original time series. \times is reconstructed time series from representation. N is segment number. 74

3.2 An example of the *SAPLA* process. Grey circle \circ is the original time series point. $n = 20, N = 4$. The blue cross \times is reconstructed time series point from *SAPLA*. 76

3.3 Framework of *SAPLA*: 1) Initializing C into \hat{C} . 2) Split & merge iteration reduces β of \hat{C} . 3) Segment endpoint movement iteration reduces β of \hat{C} 77

3.4 Example of Increment Area $\varepsilon(\check{C}'_i, \check{C}^e_i)$ simplified as two green triangles \triangle . \diamond is intersection point τ . The black circle \circ is original point c_t in C . The grey dashed dot \check{c}_t is reconstructed point \check{c}_t in \check{C}^e_i . The black dot \bullet is reconstructed point \check{c}'_t in $\check{C}'_i, t \in [0, n - 1]$ 80

3.5 Example of proof $\beta_i \geq \theta_i$ in general cases, suppose $\theta_i = c_{t_i} - \check{c}''_{t_i}$. The black circle \circ is original point c_t in $C, t \in [0, n - 1]$. The grey dashed dot \check{c}_t is reconstructed point \check{c}_t in \check{C}^e_i . The green dot \bullet is reconstructed point \check{c}'_t in \check{C}'_i . The black dot \bullet is reconstructed point \check{c}''_t in $\check{C}''_i, c_m = \frac{c_{t_i} + \check{c}''_{t_i}}{2}$ 83

3.6 Example of Reconstruction Area $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ simplified as four green triangles \triangle . The black circle \circ is the original point c_t in C . The grey dashed dot \check{c}_t is the reconstructed point \check{c}_t in $\check{C}_i + \check{C}_{i+1}$. The black dot \bullet is the reconstructed point \check{c}'_t in \check{C}'_{i+1} 85

3.7 An example of finding a split point in \check{C}'_{i+1} . \bullet is the middle point of C_i . \bullet is the candidate split point in step 1), it is the middle point between endpoint and \bullet . 1) *SAPLA* gets split point \bullet with local max $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ and max magnitude (1). 2) *SAPLA* will check other candidate points \circ until has bigger $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ or all magnitudes of candidate split points are equal to \bullet magnitude. 91

3.8 Example of segment endpoint movement for \check{C}_i . \check{C}_i tries to move endpoints for β reduction. There are four cases. The grey dot \bullet is the endpoint in $\check{C}_{i-1}, \check{C}_{i+1}$. The black dot \bullet is the endpoint in \check{C}_i 94

3.9 Visual comparison of the original time series in the first ten homogeneous datasets. Each sub-figure shows one original time series. The black dot is one point in original time series. 101

3.10 Visual comparison of the original time series in the last ten homogeneous datasets. Each sub-figure shows one original time series. The black dot is one point in original time series. 102

3.11 Max deviation comparison between *SAPLA* and *APLA* on 20 homogeneous datasets. 107

3.12 The critical difference diagram of max deviation comparison on 20 homogeneous datasets. 107

4.1 A visual comparison of lower bounding distance measures for adaptive-length segment representation. The blue color circle \circ and the black color box \square are original time series points. \times and $+$ are reconstructed time series points from representation coefficients. Fig.4.1a is the Euclidean distance between two original time series. Fig.4.1b is two reconstructed time series from the *SAPLA* dimensionality reduction method ($N = 2$). Red line is the position to partition for Fig.4.1c. 113








4.2 Illustration of hierarchical DBCH-tree. 116

4.3 This example (10 time series) shows how homogeneous time series *MBRs* overlap each other. 118

4.4	Scatter plot between $Dist_{PAR}$ and $Dist_{LB}$ on 20 homogeneous datasets. The dimensionality reduction method is <i>SAPLA</i>	125
4.5	The critical difference diagram of tightness of lower bound comparison on 20 homogeneous datasets. The dimensionality reduction method is <i>SAPLA</i>	125
4.6	Scatter plot of pruning power between DBCH-tree and R-tree on 20 homogeneous datasets. The dimensionality reduction method is <i>SAPLA</i>	132
4.7	Pruning power comparison between DBCH-tree and R-tree for adaptive-length segment dimensionality reduction methods on 20 homogeneous datasets.	132
4.8	Scatter plot of k -NN time between DBCH-tree and R-tree on 20 homogeneous datasets. The dimensionality reduction method is <i>SAPLA</i>	139
4.9	k -NN time comparison between DBCH-tree and R-tree for adaptive-length segment dimensionality reduction methods on 20 homogeneous datasets.	140
A.1	Dataset <i>1HO, HandOutlines</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	147
A.2	Dataset <i>2HT, HouseTwenty</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	148
A.3	Dataset <i>3PAP, PigAirwayPressure</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	149
A.4	Dataset <i>4PAPS, PigArtPressure</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	150
A.5	Dataset <i>5CVP, PigCVP</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	151
A.6	Dataset <i>6IS, InlineSkate</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	152
A.7	Dataset <i>7EL, EthanolLevel</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	153
A.8	Dataset <i>8CT, CinCECGTorso</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	154
A.9	Dataset <i>9SHG, SemgHandGenderCh2</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	155

A.10 Dataset 10SHM, <i>SemgHandMovementCh2</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	156
A.11 Dataset 11SHS, <i>SemgHandSubjectCh2</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	157
A.12 Dataset 12ACS, <i>ACSF1</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	158
A.13 Dataset 13EHS, <i>EOGHorizontalSignal</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	159
A.14 Dataset 14EVS, <i>EOGverticalSignal</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	160
A.15 Dataset 15H, <i>Haptics</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	161
A.16 Dataset 16M, <i>Mallat</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	162
A.17 Dataset 17PM, <i>Phoneme</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	163
A.18 Dataset 18SLC, <i>StarLightCurves</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	164
A.19 Dataset 19MSR, <i>MixedShapesRegularTrain</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	165
A.20 Dataset 20MST, <i>MixedShapesSmallTrain</i> . $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M	166

List of Tables

1	Summary of Notation	17
2.1	Dimensionality Reduction Methods Comparison	28
2.2	Haar Wavelet Transform	33
2.3	[46] contains the breakpoints that divide a Gaussian distribution $([3, 10])$ of equiprobable regions.	47
2.4	A lookup table used by the $Dist_{SAX}$ function	50
2.5	Switching conditions for different cases	65
2.6	Improvement by our implementation for case 1 and 3	65
2.7	Lower Bounding Measures Comparison.	69
2.8	Comparison of Dimensionality Reduction Methods	71
3.1	20 Homogeneous Datasets [18]	100
3.2	Max deviation comparison on 20 homogeneous datasets. The baseline dimensionality reduction method is $APLA$	105
3.3	Max deviation comparison on 20 homogeneous datasets. Ranks of these dimensionality reduction methods in each dataset. The bottom line is the average rank of each method.	106
3.4	Dimensionality reduction time comparison on 20 homogeneous datasets. The baseline dimensionality reduction method is $APLA$. The time unit is second.	109
4.1	Average tightness of lower bound comparison on the twenty homogeneous datasets. The light blue rectangle  is the lower bound distance bigger than one, breaking the lower bounding lemma.	123
4.2	Tightness of lower bound comparison on the twenty homogeneous datasets. $Dist_{AE}$ does not follow the lower bounding lemma.	124
4.3	Pruning power comparison on the first ten homogeneous datasets. The white color rectangle  is original R-tree. The red color rectangle  is DBCH-tree.	128
4.4	Pruning power comparison on the last ten homogeneous datasets. The white color rectangle  is original R-tree. The red color rectangle  is DBCH-tree.	129
4.5	Pruning power comparison on the 20 homogeneous datasets.	130
4.6	k -NN search time comparison on the first ten homogeneous datasets. The  is original R-tree. The  is DBCH-tree.	136

4.7 k -NN search time comparison on the last ten homogeneous datasets. The \square is original R-tree. The \blacksquare is DBCH-tree. 137

4.8 k -NN time average rank comparison on the 20 homogeneous datasets. 138

Acronyms

APCA Adaptive Piecewise Constant Approximation. 2

APLA Adaptive Piecewise Linear Approximation. 2

CHEBY Chebyshev Polynomials. 2

GEMINI Generic Multimedia Indexing Method. 20

MBR Minimum Bounding Rectangle. 22

PAA Piecewise Aggregate Approximation. 2

PAALM Piecewise Aggregate Approximation Lagrangian Multipliers. 2

PLA Piecewise Linear Approximation. 2

SAX Symbolic Aggregate Approximation. 2

SAPLA Self Adaptive Piecewise Linear Approximation. 3

k-NN *k*-Nearest Neighbours. 3

DBCH-tree Distance Based Covering with Convex Hull Tree. 3

Notation

Notation	Meanings
n	The length of one time series
N	The number of segments

M	Baseline representation coefficient number
C	One original time series
\hat{C}	Representation of C
\check{C}	Reconstructed time series of \hat{C}
θ_i	Max deviation of \hat{c}_i
β_i	Segment upper bound
β	Sum upper bound
$\varepsilon(C, \check{C})$	$\sum_{t=0}^{n-1} c_t - \check{c}_t $
ω	Map $\langle key, value \rangle$
Q	Query time series
S	A spatio-temporal time series
l	The length of each segment
a, b	Two coefficients in the function of linear curve
r_i	Right endpoint of \hat{c}_i
$Dist_{euc}$	Euclidean Distance
$Dist_{index}$	Distance between representations
$cmax_i, cmin_i$	The maximum and minimum values of the i^{th} segment
MBR	Minimum Bounding Rectangle is the smallest rectangle that spatially contains the original time series points of the i^{th} segment
$MINDIST(Q, R)$	Minimum distance of $MBR R$ from query times Q
$MINDIST(Q, R, t)$	Minimum distance of $MBR R$ from query times Q at time t
$MINDIST(Q, G, t)$	Minimum distance of region G from query times Q at time t
$Dist_{PLA}$	The distance between two reduced PLA series

Table 1: Summary of Notation

Chapter 1

Introduction

1.1 Background

Similarity search [7] over time series is important in many applications [36], including the stock market [42], medical science [12], statistics [19], econometrics [56], quantitative finance [32], seismology [3], meteorology [28], geophysics [43], electroencephalography [70], control engineering [6], astronomy [25] and communications engineering [22]. Time series can be regarded as a high-dimensional data type, which suggests that time series could be indexed by multidimensional index structures such as the R-tree [31]. A time series $C = \{c_0, \dots, c_{n-1}\}$ with n data points can be considered as a point in an n -dimensional space and given a query time series $Q = \{q_0, \dots, q_{n-1}\}$ and a time series C from one dataset. The similarity between these two equal-length time series is typically measured with a distance measurement. The straightforward distance measurement is Euclidean distance measurement, shown in Figure 2.1. Euclidean distance [23] is applied for similarity measures between two equal-length time series. In this thesis, the most similar time series has the smallest Euclidean distance.

Application 1. (Similarity Search in Financial Domain) The stock market is a famously complex and jittery system. The price of company shares might stay the same, rise steadily, or suddenly crash. The time series of the stock prices may be collected for many years [67]. The mean value and variance of financial time series often change over time, which means financial time series may not have Gaussian distribution. Trends are driven by stochastic inflation. Financial time series show high autocorrelation and cross autocorrelation with other variables over time. People usually want to make a similarity query on large-scale financial time series data. Thus, we could use dimen-

sionality reduction techniques to reduce the original financial time series dimension and do a quick similarity search.

Application 2. (Similarity Search in Medical Signal Domain) It is essential to deal with time series for medical diagnosis problems to recognize existing or potential diseases [29]. For emergency medical service, time series features are a positive trend, special-day effects, seasonal cycles and autocorrelation. For case-based reasoning, time series is in the form of signals recorded for billions of seconds, such as electrocardiography (*ECG*) datasets. Feature extraction and weighting become complicated and depend on expert knowledge. Thus, we need dimensionality reduction methods to reduce large-scale medical time series dimensions, and we could use similar queries to diagnose new coming cases.

1.2 Related Work

Time series similarity search and dimensionality reduction techniques are widely studied and essential fields that provide the basis for many high-level data mining tasks. We could use similarity search or dimensionality reduction techniques to improve the runtime of classification, clustering, anomaly detection, motif discovery, semantic segmentation tasks etc. The time series dimensionality reduction technique is vital for decreasing the runtime of the similarity search.

The first use case of dimensionality reduction for the original time series is reducing the whole storage on the hard disk. Because there is a tremendous volume of the original time series data generated each second, storing each data point on the hard disk is challenging. The second use case is to improve the read performance. Some time series could be recorded for several years. Reading the entire time series from a large volume dataset each time could be less efficient. The third use case is to prune useless information. Some parts of the original time series would frequently appear inside the same time series. The fourth use case is for an efficient similarity search in time series datasets. A Similarity search for the original time series should scan many high dimension time series, making it low efficient in large databases, especially for nearest neighbour queries.

This thesis focus on equal-length time series data, not the data stream. Each time series has the same length, denoted as n . The distance measurement for the original equal-length time series is Euclidean distance measurement. Time series can be regarded as a high-dimensional data type, suggesting that the similarity query over the original time series would cause a “dimensionality

curse” [11]. A general framework called Generic Multimedia Indexing Method (*GEMINI*) [23] uses the dimensionality reduction technique to transfer the original time series into a lower-dimensional space (reduced space) by representation coefficients. *GEMINI* stores the representation coefficients in the reduced space with a multidimensional index. *GEMINI* uses a lower bound distance function of the Euclidean distance to improve the efficiency of the similarity search by introducing no-false-dismissals.

We apply the branch-and-bound methodology in the k nearest neighbour search. As a similarity search, the k nearest neighbour search identifies the top k nearest neighbours to the query, called the k -NN search. For a given lower bound measurement and a given index structure, the efficiency of the k -NN search depends on the quality of the dimensionality reduction technique. Max deviation [20, 15, 9, 48, 59] is used to measure the reduction performance of each dimensionality reduction method. The smaller the max deviation, the more pruning opportunity in the k -NN search.

There are two domains for time series dimensionality reduction techniques: time domain and frequency domain. We represent time series through the x -axis (timestamp) or y -axis (point value) for dimensionality reduction. For the y -axis dimensionality reduction, one classic method is Symbolic Aggregate Approximation (*SAX*) [45], which is already a well-solved problem. For x -axis dimensionality reduction, the common choice is equal-length segment dimensionality reduction, such as Piecewise Linear Approximation (*PLA*) [15], Piecewise Aggregate Approximation (*PAA*) [39], Chebyshev Polynomials (*CHEBY*) [9], Piecewise Aggregate Approximation Lagrangian Multipliers (*PAALM*) [61]. Adaptive-length segment dimensionality reduction creates segments of variable length to capture changing trends in the original time series. Adaptive Piecewise Constant Approximation (*APCA*) [40] uses the mean value of the adaptive-length segment to represent low activity area by a single segment and high activity area by several segments. Adaptive Piecewise-Linear Approximation (*APLA*) [48] combines the virtues of *PLA* [15] and *APCA* [40]. *APLA* has $O(Nn^2)$ (n is time series length, N is segment number) time complexity and gets dynamic partition by choosing segments with minimum segment max deviation.

We focus on the x -axis dimensionality reduction in this thesis because the number of time stamps is regarded as the number of dimensions, and investigating x -axis compression is already broad enough with clear potential gains. The y -axis dimensionality reduction method *SAX* is compared with other x -axis dimensionality reduction methods in this thesis. We compared *SAX* on dimensionality reduction time, the pruning power and k -NN search time. *SAX* is a symbolic version of the *PAA*. Symbolic dimensionality reduction of *SAX* does not have max deviation and does not

consider the segment trend. Thus, the reconstruction of *SAX* has lower reconstruction accuracy than the *PAA* (symbol \rightarrow number). We do not compare *SAX*'s max deviation in this thesis. We know the extension of *SAX* could perform a good similarity search on time series. However, we want to improve the similarity search performance by reducing the max deviation of the dimensionality reduction method in this thesis. Improvements in *x*-axis compression already provide good gains. Focus on *x*-axis methods is already a sufficiently broad topic. Proposing an extended *SAX* method is our future work.

1.3 Motivation

We focus on speeding up the adaptive-length dimensionality reduction method in this thesis. This section describes the reason for proposing a fast adaptive-length segment dimensionality reduction method with a small maximum deviation close to optimal, called Self Adaptive Piecewise-Linear Approximation (*SAPLA*), the improved lower bound distance measure denoted as $Dist_{PAR}$ and the improved node splitting and branch picking algorithms in the improved index structure.

- Equal-length segment dimensionality reduction method has unstable performance on time series with a drastic changing trend. For example, *PAA* shows different performances in different homogeneous datasets.

Adaptive-length dimensionality reduction method tries to find the approximate segments of the original time series, but it is time-consuming, especially for regularly changed time series. *APLA* tries to minimize the segment max deviation but cannot guarantee getting the minimum sum max deviation. *APLA* uses $\hat{c}_i = \{a_i, b_i, r_i\}$ to represent the i^{th} segment. a_i and b_i are two coefficients in a linear function $\check{c}_{t+r_{i-1}+1} = a_i * t + b_i$, $0 \leq t < l_i$, ($l_i = r_i - r_{i-1}$ is the length of the i^{th} segment). For example, there are two adjacent segment representations \hat{c}_i and \hat{c}_{i+1} . \hat{c}_i already has the minimum max deviation. It is possible that a movement of \hat{c}_i right endpoint would reduce the sum max deviation. Instead of computing the segment max deviation with $O(l_i)$ time complexity, the segment max deviation upper bound is proposed in *SAPLA* in this thesis. *SAPLA* also considers merging adjacent segments to reduce the sum max deviation. Because *APLA* has the guaranteed error bounds (scan each point to get max deviation) in the reduction process, *APLA* has $O(Nn^2)$ time complexity (n is the original time series length and N denotes the segment number after the reducing process). Our experiment shows that the dimensionality reduction time of *APLA* is much slower than other methods.

- A lower bound of actual Euclidean distance between time series helps guarantee no false dismissals while doing a similarity search. *APCA* [40] proposes two lower bounding distance measures that make adaptive-length dimensionality reduction methods indexable. One keeps a lower bounding lemma, called $Dist_{LB}$, and another has a tight Euclidean distance approximation but a non-lower bounding called $Dist_{AE}$. We propose $Dist_{PAR}$, which has a guaranteed lower bounding lemma and tightness.
- *GEMINI* [23] structure maps the reduced data into a multi-dimensional index. In a branch-and-bound [16] search (e.g. K nearest neighbour search denoted as k -NN), R-tree [31] is a multi-dimensional index. The node splitting and branch picking algorithms in R-tree [31] are not efficient for homogeneous time series datasets which are from the same data sources (e.g. temperature sensors). Node splitting algorithm attempts to find a small-area split, and the branch picking algorithm also tries to pick a branch with a minimum area increase. Minimum Bounding Rectangle (*MBR*) of homogeneous time series in R-tree could cause an overlap problem.

1.4 Contributions

For whole sequence similarity matching, R-tree index, DBCH-tree index and k -NN algorithms are implemented. Under the *GEMINI* structure, this thesis proposes one adaptive-length segment dimensionality reduction method *SAPLA*, one lower bounding measure $Dist_{PAR}$ for adaptive-length segment dimensionality reduction methods, and a DBCH-tree structure that uses distance based node splitting and branch picking algorithms for adaptive-length segment dimensionality reduction methods. Our implementation language is C++ and can be publicly available on our website [1].

We evaluated several classic dimensionality reduction methods, one R-tree index structure and two k -NN similarity search methods in Chapter 2. These dimensionality reduction methods include equal-length segment dimensionality reduction methods and adaptive-length segment dimensionality reduction methods. Our experimental results provide a comprehensive comparison under the same evaluation condition.

We propose a fast adaptive-length segment dimensionality reduction method with a small maximum deviation close to optimal, called Self Adaptive Piecewise-Linear Approximation (*SAPLA*) overcomes the time complexity problems of *APLA*. (Chapter 3).

We propose an adaptive-length lower bounding distance measure $Dist_{PAR}$ for use with *SAPLA*

representation that guarantees a lower bound and the bound is tighter than the commonly used alternative. (Chapter 4)

We propose a DBCH-tree index structure and the distance based node-splitting and branch picking algorithm for use with *SAPLA*. This DBCH-tree solves the overlap problem in time series Minimum Bounding Rectangle (*MBR*) [40] and could efficiently improve indexing. (Chapter 4)

1.5 Limitations

- The dimensionality reduction method that focuses on reducing the Euclidean distance between the original time series and reconstructed time series from the representation coefficients is not sufficient enough. For the time series that needs nonlinear time warping of the local compression or expansion of the time scale. Dynamic Time Warping (*DTW*) distance measurement is better than Euclidean distance measurement.
- This thesis could not propose a dimensionality reduction method that has better max deviation than the adaptive-length segment dimensionality reduction method *APLA* and is faster than the equal-length segment dimensionality reduction method, which has $O(n)$ time complexity.
- This thesis only proposes an adaptive-length segment dimensionality reduction method for the x -axis dimensionality reduction, not for the y -axis dimensionality reduction.
- This thesis has no research on higher-level data mining tasks like calculating the matrix profile for motif discovery. There are no higher-level data mining tasks such as classification, motif discovery, or anomaly detection.
- The proposed adaptive-length segment dimensionality reduction method applies a conditional upper bound for max deviation reduction, not an unconditional upper bound.
- The similarity search in this thesis is whole sequence matching, not sub-sequence matching.
- The DBCH-tree constructs the index structure based on distance. If the entry does not consider the distance between each other or cannot get accurate distance measures, DBCH-tree will put dissimilarity entries together and damage the similarity search process.

1.6 Thesis Outline

This thesis starts with an introduction, followed by datasets used in this thesis, literature review, proposed adaptive-length segment dimensionality reduction method Self Adaptive Piecewise-Linear Approximation (*SAPLA*), adaptive-length lower bounding distance measure $Dist_{PAR}$, distance-based node splitting and branch picking in DBCH-tree and conclusions. Following are individual chapter summaries:

Chapter 2 provides a comprehensive literature review of the classic dimensionality reduction methods, distance measures, and the R-tree index.

Chapter 3 proposes the adaptive-length segment dimensionality reduction method Self Adaptive Piecewise-Linear Approximation (*SAPLA*).

Chapter 4 proposes adaptive-length lower bounding distance measure $Dist_{PAR}$ that lower bound Euclidean distance. $Dist_{PAR}$ is a tight approximation of the Euclidean distance. We propose a DBCH-tree that node splitting and branch picking algorithms based on the distance to solve overlap problems in homogeneous datasets.

Chapter 5 discusses the conclusions and future work.

Chapter 2

Literature Review

We have analyzed and evaluated seven classic dimensionality reduction technologies (*PAA* [39], *APCA* [40], *PLA* [15], *CHEBY* [9], *APLA* [48], *PAALM* [61], *SAX* [60]) through real homogeneous datasets [18]. We implemented these methods by C++ language. We have found some limitations and making possible improvements. We introduce the basic knowledge of time series, distance measure and lower bound lemma in this chapter.

2.1 Definition & Lemma

Definition 2.1.1. Time Series (C). Time series is a set of sequences c_i , and each one is recorded at a specified time t_i . Time series are made at fixed time intervals [8]. Time series is like $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$). t_i is the time stamp, and c_i is the value of the data point. If two time series have the same length and time stamp, we can regard them as two data streams in ascending order, like $c_i = (t_i, c_i)$. The whole time series could be presented as $C = \{c_0, c_1, \dots, c_{n-1}\}$.

Definition 2.1.2. Representation (\hat{C}). \hat{C} is an N segments sequence as a representation of the original time series C . $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ ($N \leq n$). For an adaptive-length linear curve, $\hat{c}_i := \langle a_i, b_i, r_i \rangle$ represents the i^{th} segment.

Definition 2.1.3. Reconstructed Time Series (\check{C}). The reconstructed time series is from representation coefficients, defined as $\check{C} = \{(t_0, \check{c}_0), (t_1, \check{c}_1), \dots, (t_{n-1}, \check{c}_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) = $\{\check{c}_0, \check{c}_1, \dots, \check{c}_{n-1}\} = \{\check{C}_0, \check{C}_1, \dots, \check{C}_{N-1}\}$. The i^{th} reconstructed segment is $\check{C}_i = \{\check{c}_{r_{i-1}+1}, \dots, \check{c}_{r_i}\}$. Like +s in Figure 2.4.

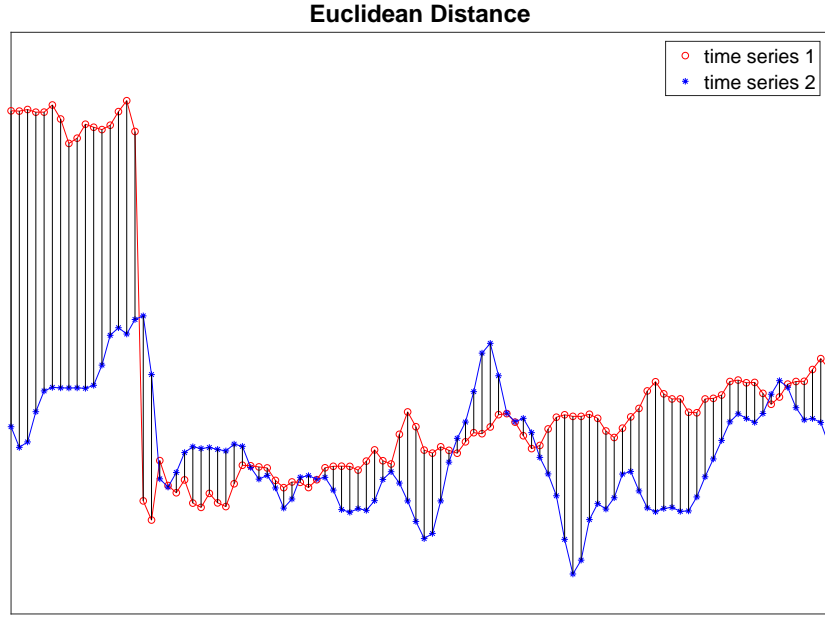


Figure 2.1: The Euclidean distance can be visualized as the square root of the sum of the squared lengths of the black lines.

Definition 2.1.4. Euclidean Distance. There are two time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_i, c_i), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_2 < \dots < t_{n-1}$) and $D = \{(t_0, d_0), (t_1, d_1), \dots, (t_j, d_j), \dots, (t_{n-1}, d_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$). Eq. 2.1 shows how to compute the Euclidean distance between two time series. An example of Euclidean distance computation is shown in Figure 2.1. The Euclidean distance between two time series is the length of the black lines segment connecting them.

$$Dist_{euc}(C, D) = \sqrt{\sum_{i=0}^{n-1} (c_i - d_i)^2} \quad (2.1)$$

Definition 2.1.5. Max Deviation (θ_i). Max deviation [9] is the maximum absolute difference between original time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ and reconstructed time series $\check{C} = \{(t_0, \check{c}_0), (t_1, \check{c}_1), \dots, (t_{n-1}, \check{c}_{n-1})\}$ from representation $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$. For the i^{th} segment representation \hat{c}_i , its segment maximum deviation is $\theta_i := \max_{t=r_{i-1}+1}^{r_i} |c_t - \check{c}_i|$. This means the maximum difference between two time series. The maximum equation is shown in Eq. 2.2. An example of maximum deviation is shown in Figure 2.2.

$$maximun\ deviation = \max |c_i - \check{c}_i|, i \in [0, \dots, n-1] \quad (2.2)$$

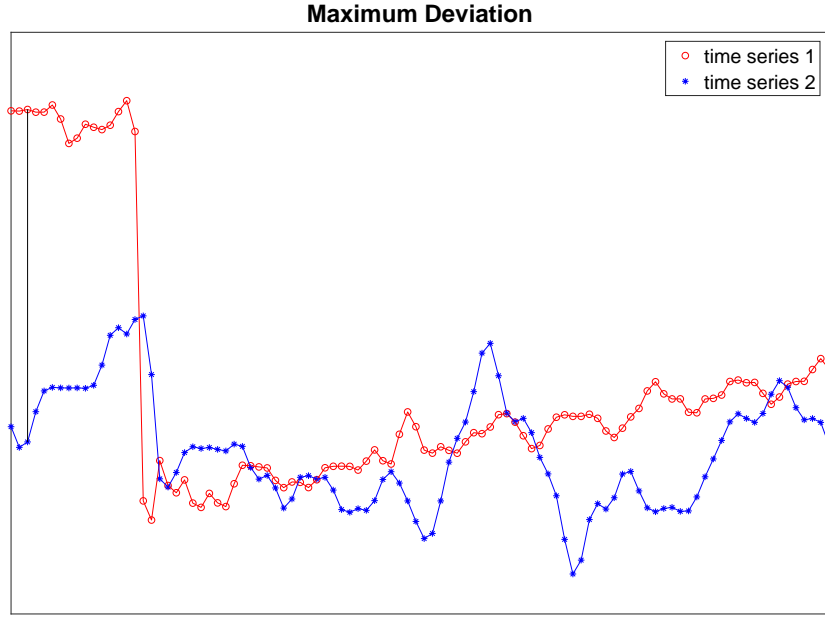


Figure 2.2: The maximum deviation can be visualized as the black line.

Definition 2.1.6. k -Nearest-Neighbor(k -NN). Classification is a fundamental and straightforward non-parametric method. For query object Q , k -NN algorithm searches k (k is a positive integer) nearest neighbours.

Lemma 1. Lower Bounding Lemma [23]. There two time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) and $D = \{(t_0, d_0), (t_1, d_1), \dots, (t_{n-1}, d_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), the representation of these two time series are $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ and $\hat{D} = \{\hat{d}_0, \hat{d}_1, \dots, \hat{d}_{N-1}\}$. $N \ll n$. The reconstructed time series of these representations \hat{C} and \hat{D} are $\check{C} = \{(t_0, \check{c}_0), (t_1, \check{c}_1), \dots, (t_{n-1}, \check{c}_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) and $\check{D} = \{(t_0, \check{d}_0), (t_1, \check{d}_1), \dots, (t_{n-1}, \check{d}_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) The lower bounding of them is in Eq. 2.3.

$$Dist_{index}(\hat{C}, \hat{D}) = Dist_{euc}(\check{C}, \check{D}) \leq Dist_{euc}(C, D) \quad (2.3)$$

Lower bounding means a distance metric between two reconstructed time series is always less than or equal to the Euclidean distance between these original time series. The importance of lower bounding is that when indexing time series, there is no false negative phenomenon.

2.2 Relevant Dimensionality Reduction Methods [62]

Plenty of dimensionality reduction methods are developed [5] for reducing the dimension (the number of data points) in the original time series. Figure 2.3 shows a classification of classic dimensionality reduction methods. There are two domains for time series dimensionality reduction: time domain and frequency domain. The frequency domain contains two classic methods, Discrete Fourier Transform (*DFT*) [23, 58] and Discrete Wavelet Transform (*DWT*) [37, 14]. We focus on time domain reduction in this thesis. We can reduce time series through the x-axis and y-axis. For y-axis reduction methods, one method is Symbolic Aggregate Approximation (*SAX*) [46, 26]. For x-axis reduction methods, one choice is the equal-length segment reduction method, such as Piecewise Aggregate Approximation (*PAA*) [68, 39, 38], Chebyshev polynomials (*CHEBY*) [52, 10], Indexable Piecewise Linear Approximation (*PLA*) [51, 15] and Piecewise Aggregate Approximation Lagrangian Multipliers (*PAALM*) [61]. For the adaptive-length segment method, we implemented Adaptive Piecewise Constant Approximation (*APCA*) [24, 40] and Adaptive Piecewise-Linear Approximation (*APLA*) [48].

Seven classic dimensionality reduction methods are implemented, including *PAA*, *APCA*, *PLA*, *APLA*, *CHEBY*, *PAALM* and *SAX*. Every method will be introduced in the following sub-sections. We analyze these dimensionality methods and point out the advantages and disadvantages of these methods. We make some improvements during our implementation and evaluation. Table 2.1 shows the summaries of eight dimensionality reduction methods.

Name	Time	Coeffici.	Seg. Num	Seg. Size	Dim.
* <i>SAPLA</i>	$O(n(N + \log n))$	a_i, b_i, r_i	$N = M/3$	Adaptive	x-axis
<i>APLA</i>	$O(Nn^2)$	a_i, b_i, r_i	$N = M/3$	Adaptive	x-axis
<i>APCA</i>	$O(n \log n)$	v_i, r_i	$N = M/2$	Adaptive	x-axis
<i>PLA</i>	$O(n)$	a_i, b_i	$N = M/2$	Equal	x-axis
<i>PAA</i>	$O(n)$	v_i	$N = M$	Equal	x-axis
<i>PAALM</i>	$O(n)$	v_i	$N = M$	Equal	x-axis
<i>CHEBY</i>	$O(Nn)$	che_i	$N = M$	Equal	x-axis
<i>SAX</i>	$O(n)$	alphabet	$N = M$	Equal	y-axis

Table 2.1: Dimensionality Reduction Methods Comparison

2.2.1 Piecewise Aggregate Approximation (*PAA*) [39]

PAA [62, 39] uses the average value of equal-length segment to represent the original time series C . *PAA* has $O(n)$ time complexity. The time cost of reduced time series is faster than sequential scanning original time series [39, 38]. Its segment number is three times that of *SAPLA*, as Table

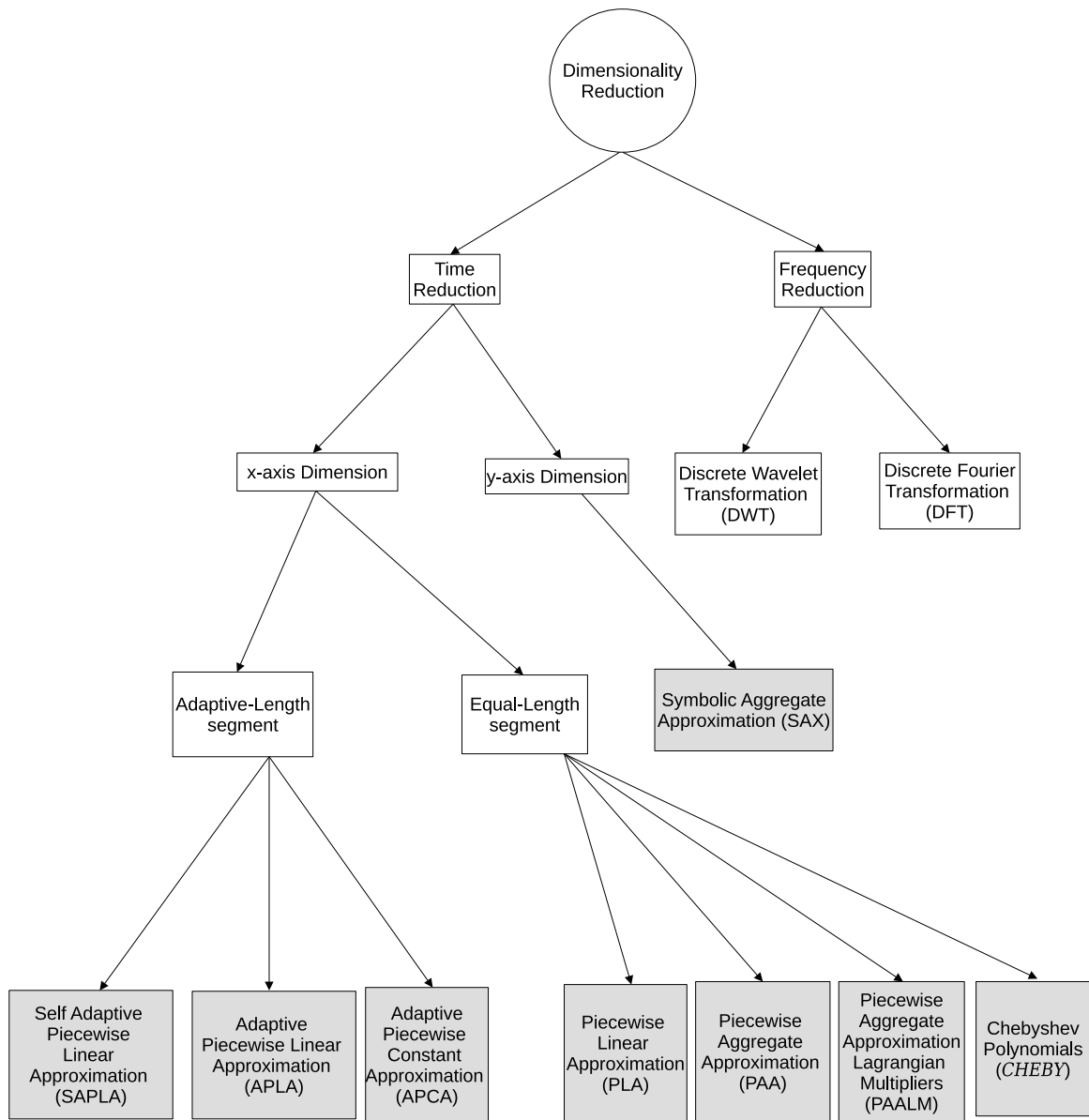


Figure 2.3: Categorization of dimensionality reduction methods including time reduction and frequency reduction. The grey colour rectangle ■ is what this thesis compares.

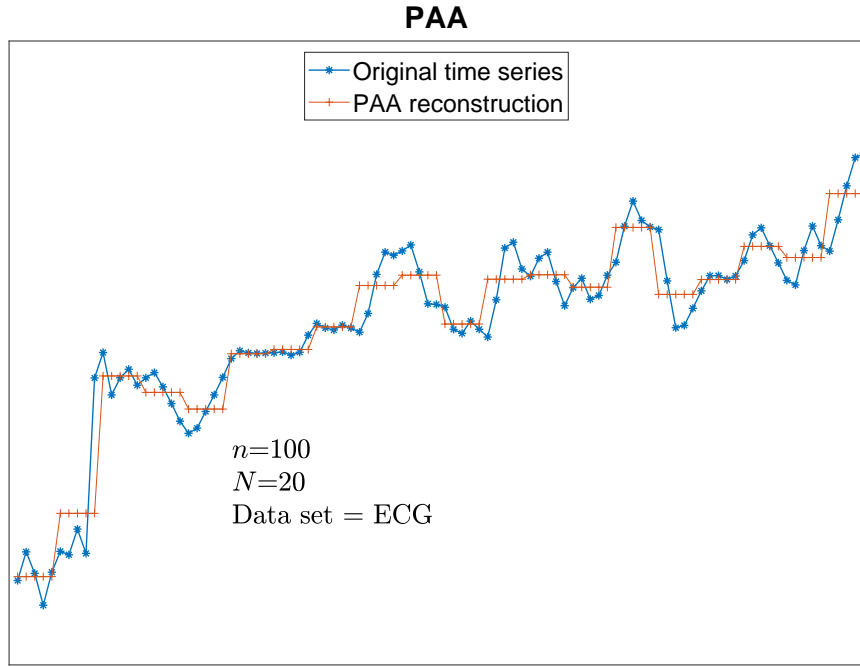


Figure 2.4: This figure is a visual illustration of the equal-length segment dimensionality reduction method *PAA*.

2.1 shows.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), the length of this time series is n and the segment number is N . The reduced time series is $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$, \hat{C} has N segments. \hat{c}_i is the mean value of the i^{th} segment. $\frac{n}{N}$ is the point number of the i^{th} segment. The representation coefficient number $M = N$. Figure 2.4 shows an example of the *PAA* reconstruction time series. A time series consisting of one hundred points are projected into twenty segments. The time series is divided into twenty segments, and the mean value of each segment is calculated. Each segment contains five points. The equation of every line segment is like $y = b$ in the linear function $a * t + b$. For the i^{th} segment representation coefficient, the equation is shown in Eq. (2.4) [39]. Algorithm 2.2.1 shows the *PAA* computation of the algorithm. The mean value transformation of a time series can be calculated in $O(n)$.

$$\hat{c}_i = \left\lfloor \frac{N}{n} \right\rfloor \sum_{j=\lfloor \frac{n}{N} \rfloor * i}^{\lfloor \frac{n}{N} \rfloor * (i+1) - 1} c_j \quad (2.4)$$

Example 1. There is a time series $C = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7\} = \{8, 6, 6, 4, 4, 4, 5, 7\}$.

Algorithm 2.2.1: Compute PAA // $O(n)$

```

input :  $C = \{c_0, c_1, \dots, c_{n-1}\}$ : original time series;
 $n$ : time series length;
 $N$ : the user defined number of segments;
output:  $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ : PAA representation of the original time series  $C$ ;
1 The length of each segment is  $l = \lfloor \frac{n}{N} \rfloor$ ;
2 for  $i \leftarrow 0$ ;  $i < N$ ;  $i++$  do //  $O(N)$ 
3    $sum \leftarrow 0$ ;
4   for  $j \leftarrow l * i$ ;  $i < l * (i + 1) - 1$ ;  $i++$  do //  $O(l)$ 
5      $sum += c_j$ ;
6    $\hat{c}_i \leftarrow \frac{sum}{l}$ ;
7 return  $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$  //  $O(n) = O(N * l)$ 

```

The time series length n is eight. The user defined segment number N is four. The length of each segment l is $n \div N = 8 \div 4 = 2$. We first compute the summation of each segment is $\{8 + 6 = 14, 6 + 4 = 10, 4 + 4 = 8, 5 + 7 = 12\}$. Then we will get the PAA representation by computing the mean value of each segment $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3\} = \{14 \div 2 = 7, 10 \div 2 = 5, 8 \div 2 = 4, 12 \div 2 = 6\} = \{7, 5, 4, 6\}$. The time complexity of computing the mean value of each segment is $O(l)$.

2.2.2 Adaptive Piecewise Constant Approximation (APCA) [40]

The equal-length segment dimensionality reduction methods have limitations. For example, when the part of original time series in two adjacent segments are similar, it still needs two segments to represent them. APCA represents the original time series by varying length segments. APCA [40] uses the average value v_i of adaptive-length to represent the original time series C through Haar wavelet transformation. APCA has $O(n \log n)$ time complexity, and APCA focuses on improving the tightness of individual segment with adaptive-length and constant value. Besides the mean value to represent every segment, APCA also needs to record the position of every segment.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), after transformation, APCA representation looks like $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\} = \{\langle cv_0, cr_0 \rangle, \langle cv_1, cr_1 \rangle, \dots, \langle cv_{N-1}, cr_{N-1} \rangle\}$. Where cr_i records the position of the i^{th} segment's right endpoint, cv_i records the mean value of the i^{th} segment. The length of every segment can be got by $l_i = cr_i - cr_{i-1}$. The representation coefficient number $M = N * 2$. Figure 2.5 shows an example of the APCA reconstruction time series. The idea of APCA dimensionality reduction makes use of the Haar wavelet transformation method [13]. This method depends on decreasing the normalized magnitude coefficient to decide which level of compression is optimal [50]. For represented segments, APCA

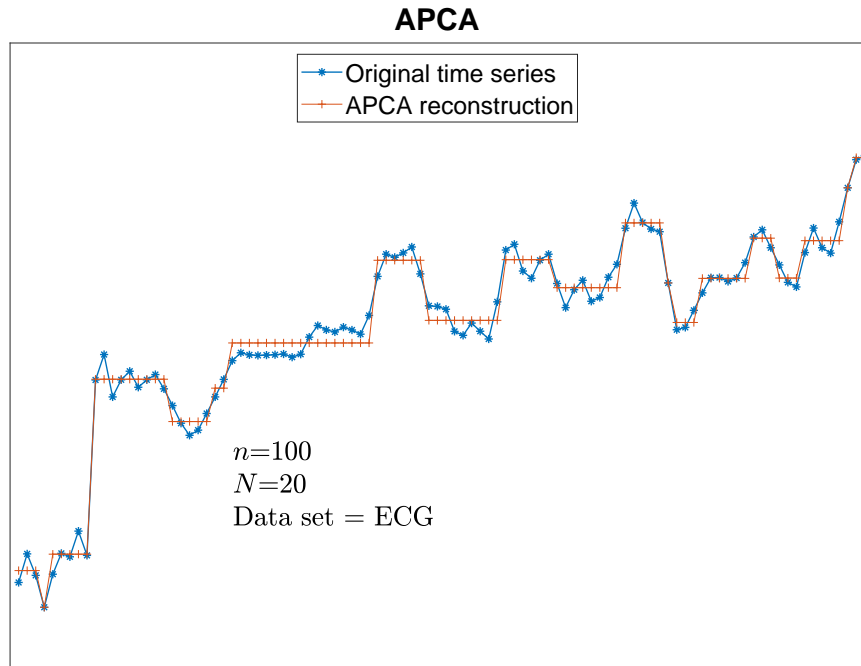


Figure 2.5: This figure is a visual illustration of adaptive-length segment dimensionality reduction method *APCA*.

replaces represented mean value with the exact mean value.

Algorithm 2.2.2 shows how to compute *APCA*. *APCA* first converts original time series into wavelet compression problem and gets optimal solutions, then converts the solution back to *APCA* dimensionality reduction [13]. Haar wavelet transformation has $O(n)$ time complexity. An optimal reconstruction for every compression level can be achieved by sorting the coefficients in order to decrease normalized magnitude and discarding small coefficients. If the number of segments is more than coefficients' number N , merge adjacent segments until getting N segments.

Example 2. (Computing *APCA* Dimensionality Reduction [40]). For a time series $C = \{8, 6, 6, 4, 4, 4, 5, 7\}$, Table 2.2 makes use of Haar wavelet transformation of time series. First, we pairwise averaged the value to get a lower-resolution representation of the data with the following average values $[(8 + 6) \div 2, (6 + 4) \div 2, (4 + 4) \div 2, (5 + 7) \div 2] = [7, 5, 4, 6]$. Second, *APCA* in order to reconstruct the original time series C , the differences of the second of the averaged values from the pairwise average are computed, which is $[7 - 6, 5 - 4, 4 - 4, 6 - 7] = [1, 1, 0, -1]$. We apply pairwise averaging and different processes recursively on the lower resolution array. We get the complete transformation shown in Table 2.2. We already have the overall average value of the time

Algorithm 2.2.2: Compute *APCA* // $O(n \log n)$

input : $C = \{c_0, c_1, \dots, c_{n-1}\}$: original time series;
 n : time series length;
 N : the number of segments;
output: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$: *APCA* representation of C ;

- 1 **if** $n \neq 1$ **then**
- 2 └ Pad it with zeros to make the length of the time series C is a power of two.
- 3 We do Haar Discrete Wavelet Transform on the time series C .
- 4 We sort coefficients in order of decreasing normalized magnitude, and we retain the first N Haar wavelet coefficients.
- 5 We reconstruct *APCA* representation of the time series C from retained N Haar wavelet coefficients.
- 6 **if** n was padded with zeros **then**
- 7 └ Truncate it to the original length.
- 8 Replace represent segment mean values with exact mean values.
- 9 **while** the number of segments is greater than N **do**
- 10 └ Merge the pair of segments that can be merged with the least rise in error.

series C is 5.5 in the lowest resolution and seven coefficients. The wavelet transform is $[5.5, 0.5, 1, -1, 1, 1, 0, -1]$. We normalized the wavelet transform by dividing each coefficient by $2^{resolution \div 2}$. We will get $[5.5 \div 2^0, 0.5 \div 2^0, 1 \div 2^{\frac{1}{2}}, -1 \div 2^{\frac{1}{2}}, 1 \div 2, 1 \div 2, 0 \div 2, -1 \div 2] = [5.5, 0.5, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0.5, 0.5, 0, -0.5]$. Suppose the user defined segment number is $N = 3$. So we keep the three wavelet coefficients with the highest normalized magnitude, the first, third and fourth wavelet coefficients $\{5.5, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\}$. We will get $[5.5, 0, 1, -1, 0, 0, 0, 0]$, and we reconstruct the time series by these three wavelet coefficients. We will get approximate mean values $[6.5, 4.5, 4.5, 6.5]$. We replace the approximate mean value with the exact mean value from the original time series C , and the *APCA* representation is $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3\} = \{\langle cv_0, cr_0 \rangle, \langle cv_1, cr_1 \rangle, \langle cv_2, cr_2 \rangle, \langle cv_3, cr_3 \rangle\} = \{\langle 7, 1 \rangle, \langle 5, 3 \rangle, \langle 4, 5 \rangle, \langle 6, 7 \rangle\}$. Finally, we merge the second and the third segment to get user defined segment number $N = 3$. The final *APCA* representation is $\hat{C} = \{\langle 7, 1 \rangle, \langle 4.5, 5 \rangle, \langle 6, 7 \rangle\}$.

Resolution	Mean Values	Coefficients
3	[8, 6, 6, 4, 4, 4, 5, 7]	————
2	[7, 5, 4, 6]	[1, 1, 0, -1]
1	[6, 5]	[1, -1]
0	[5.5]	[0.5]

Table 2.2: Haar Wavelet Transform

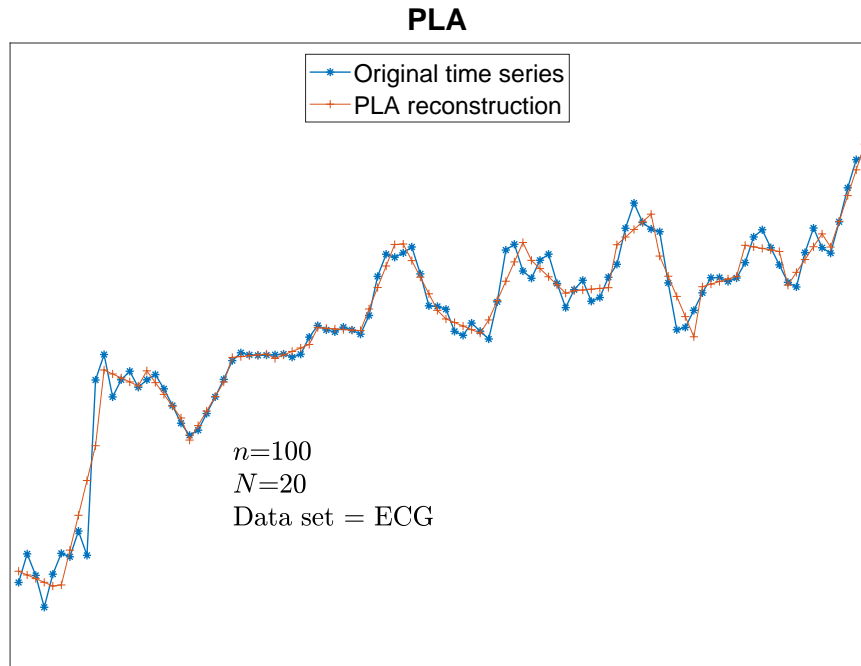


Figure 2.6: The x -axis represents the time, and the y -axis represents the real value, the orange line represents represented line segments, the equation of every line segment is $y = a * t + b$.

2.2.3 Piecewise Linear Approximation (PLA) [15]

Above *PAA* and *APCA* both apply mean values to represent the original time series C . *PLA* [15] represents the original time series C with linear function. *PLA* divides the original time series C into several equal-length segments as *PAA* does. Then every segment is represented by a linear function $\check{c}_t = a \times t + b, t \in [0, l]$ [55]. Let c_t denote the point value of position t in time series C . a is a slope, b is a y -intercept in a linear function [54, 41]. For the i^{th} segment, the length of each segment is $l = \frac{n}{N}$. *PLA* has $O(n)$ time complexity.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), and a user-defined segment number N , the linear function $f(t) = a * t + b, t \in [0, 1, \dots, l]$ can represent every segment. Thus after *PLA* transformation, we will get *PLA* representation like $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\} = \{\langle a_0, b_0 \rangle, \langle a_1, b_1 \rangle, \dots, \langle a_{N-1}, b_{N-1} \rangle\}$. The representation coefficient number $M = N * 2$. Figure 2.6 shows an example of the *PLA* reconstruction time series. *PLA* representation coefficients a_i and b_i can be got by Eq. (2.5) and Eq. (2.6). Algorithm 2.2.3 shows the *PLA* computation of the algorithm.

$$a_i = \frac{12 \sum_{j=il}^{(i+1)l-1} (j - il - \frac{l-1}{2}) c_j}{l(l-1)(l+1)} \quad i = 0, 1, 2, \dots, N-1 \quad (2.5)$$

$$b_i = \frac{2 \sum_{j=il}^{(i+1)l-1} (2l-1-3(j-il)) c_j}{l(l+1)} \quad i = 0, 1, 2, \dots, N-1 \quad (2.6)$$

Algorithm 2.2.3: Compute *PLA* // $O(n)$

input : $C = \{c_0, c_1, \dots, c_{n-1}\}$: original time series;

n : time series length;

N : the user defined number of segments;

output: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$: *PLA* representation of the original time series C ;

1 The length of each segment is $l = \lfloor \frac{n}{N} \rfloor$;

2 **for** $i \leftarrow 0$; $i < N$; $i++$ **do** // $O(N)$

3 $a_i :=$ Equation (2.5);

4 $b_i :=$ Equation (2.6);

5 $\hat{c}_i := \langle a_i, b_i \rangle$;

6 **return** $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$ // $O(n) = O(N * l)$

Example 3. There is a time series $C = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, \dots, c_{19}\} = \{7, 8, 20, 15, 18, 8, 8, 15, 10, 1, 4, 3, 3, 5, 4, 9, 2, 9, 10, 10\}$. The time series length n is twenty. The user defined segment number N is four. The length of each segment l is $n \div N = 20 \div 4 = 5$. We compute the linear function coefficients a_i and b_i of the i^{th} segment by Eq. (2.5) and Eq. (2.6). We will get the *PLA* representation $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3\} = \{\langle a_0, b_0 \rangle, \langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_3, b_3 \rangle\} = \{\langle 2.9, 7.8 \rangle, \langle -1.2, 10.8 \rangle, \langle 0.2, 3.4 \rangle, \langle 1, 6 \rangle\}$. The time complexity of computing the *PLA* representation coefficients a and b of each segment is $O(l)$.

2.2.4 Adaptive Piecewise-Linear Approximation (*APLA*) [48]

Above *APCA* uses adaptive-length segment and constant value to represent the original time series. *PLA* uses equal-length segment and a linear function $a * t + b$ to represent the original time series. Thus, *APLA* uses adaptive-length segment and linear function to represent the original time series. *APLA* has $O(Nn^2)$ time complexity.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), and a user defined segment number N , linear function $f(t) = a * t + b$ can represent every segment. Thus after transformation, we will get *APLA* representation like $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\} = \{\langle a_0, b_0, r_0 \rangle, \langle a_1, b_1, r_1 \rangle, \dots, \langle a_{N-1}, b_{N-1}, r_{N-1} \rangle\}$. For the i^{th} segment, the length of each segment is

$l_i = r_i - r_{i-1}$. The representation coefficient number $M = N * 3$. Figure 2.7 shows an example of the *APLA* reconstruction time series.

APLA builds a maximum deviation matrix $\varpi[n, N]$ [48]. $\varpi[m, i]$ is the maximum deviation of the best i -segment *APLA* representation to points $\{0, \dots, m\}$. m is the right endpoint r_i of the i^{th} segment. Once the best $(i-1)$ -segment *APLA* representation $\{\langle a_0, b_0, r_0 \rangle, \dots, \langle a_{i-1}, b_{i-1}, r_{i-1} \rangle\}$ is known for each prefix of the points $\{0, \dots, m-1\}$, the best i -segment representation for points $0, \dots, m$ can be computed through $\varpi[m, i] = \min_{\alpha=i}^{m-1} (\varpi[\alpha, i-1] + \varpi_i)$. ϖ_i is the maximum deviation of the i^{th} segment where the segment right endpoint $r_i = m$, so the i^{th} *APLA* segment representation is $\langle a_i, b_i, m \rangle$. $\alpha = r_{i-1} + 1$ is the left endpoint of the i^{th} segment. The $(i-1)^{\text{th}}$ segment representation is regarded as $\langle a_{i-1}, b_{i-1}, \alpha - 1 \rangle$. We can get $\varpi[m, i]$, which is the maximum deviation for the optimal i -segment *APLA* representation $\{\langle a_0, b_0, r_0 \rangle, \dots, \langle a_{i-1}, b_{i-1}, \alpha - 1 \rangle, \langle a_i, b_i, m \rangle\}$. *APLA* has guaranteed maximum deviation bounds in the dimensionality reduction process. However, *APLA* has $O(Nn^2)$ time complexity. *APLA* combines the virtues of other dimensionality reduction methods. *APLA* reduces the original time series adaptively. *APLA* represents each segment by a linear function $a * t + b$, which is better than a constant value. The algorithm of *APLA* is shown in Algorithm 2.2.4.

Example 4. There is a time series $C = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, \dots, c_{19}\} = \{7, 8, 20, 15, 18, 8, 8, 15, 10, 1, 4, 3, 3, 5, 4, 9, 2, 9, 10, 10\}$. The time series length n is twenty. The user defined segment number N is four. We compute the linear function coefficients a_i and b_i of the i^{th} segment by Eq. (2.5) and Eq. (2.6). *APLA* builds a maximum deviation matrix $\varpi[20, 4]$, an *APLA* representation matrix $\hat{C}[20, 4]$, and a segment left endpoint position matrix $A[20, 4]$. *APLA* first computes the a_0 and b_0 of the first segment representation \hat{c}_0 . The segment right endpoint r_0 is from 1 to $n-1$. So, the $\hat{C}[m, 0]$ ($m = 1, \dots, 19$) is $\langle 1, 7, 1 \rangle, \langle 6.5, 5.2, 2 \rangle, \langle 3.6, 7.1, 3 \rangle, \langle 2.9, 7.8, 4 \rangle, \langle 0.86, 10.52, 5 \rangle, \langle 0.035, 11.9, 6 \rangle, \langle 0.27, 11.4, 7 \rangle, \langle 0.03, 11.98, 8 \rangle, \langle -0.58, 13.6, 9 \rangle, \langle -0.75, 14.14, 10 \rangle, \langle -0.86, 14.5, 11 \rangle, \langle -0.9, 14.6, 12 \rangle, \langle -0.84, 14.4, 13 \rangle, \langle -0.8, 14.25, 14 \rangle, \langle -0.66, 13.6, 15 \rangle, \langle -0.68, 13.6, 16 \rangle, \langle -0.56, 13, 17 \rangle, \langle -0.45, 12.4, 18 \rangle, \langle -0.36, 11.86, 19 \rangle$. The $\hat{\varpi}[m, 0]$ ($m = 1, \dots, 19$) is 0, 3.66667, 5.7, 6.4, 7.7619, 8.03571, 8.03571, 7.95556, 7.54545, 7.37273, 7.5, 7.63736, 7.4, 7.36429, 7.76765, 7.70588, 8.10664, 8.51228, 8.86015. The $A[m, 0]$ ($m = 1, \dots, 19$) is . We continue computing the second segment coefficients $\hat{C}[m, 1]$, $\hat{\varpi}[m, 1]$ and $A[m, 1]$, the third segment coefficients $\hat{C}[m, 2]$, $\hat{\varpi}[m, 2]$ and $A[m, 2]$ and so on.

We will get the *APLA* representation $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3\} = \{\langle a_0, b_0, r_0 \rangle, \langle a_1, b_1, r_1 \rangle, \langle a_2, b_2, r_2 \rangle, \langle a_3, b_3, r_3 \rangle\} = \{\langle 6.5, 5.16667, 2 \rangle, \langle -1, 14.8, 7 \rangle, \langle -0.571429, 5.7619, 13 \rangle, \langle 1.14286, 4.47619, 19 \rangle\}$.

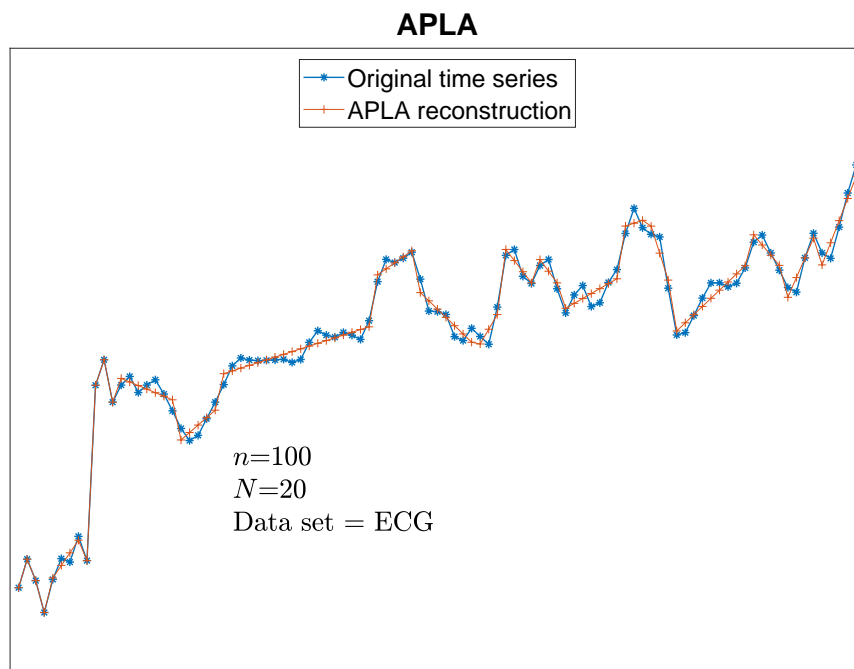


Figure 2.7: An illustration of data reduction method *APLA*, which is the adaptive-length segment approach.

Algorithm 2.2.4: Compute *APLA* [48] // $O(Nn^2)$

input : $C = \{c_0, c_1, \dots, c_{n-1}\}$: original time series;
 n : length of time series;
 N : number of segment;

output: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$: *APLA* representation;

- 1 $\hat{C}[n, N]$: *APLA* representation matrix;
- 2 $\bar{\omega}[n, N]$: maximum deviation matrix;
- 3 $A[n, N]$: segment left endpoint position matrix;
- 4 **for** $m = 1$ to $n - 1$ **do** // $O(n * (n - 1)) = O(n^2)$
- 5 Computing the first segment *APLA* representation \hat{c}_0 to points c_0 through c_m ; // $O(l)$
- 6 $\hat{C}[m, 0] \leftarrow \hat{c}_0$;
- 7 $\bar{\omega}[m, 0] \leftarrow$ Computing the maximum deviation of \hat{c}_0 ; // $O(m)$
- 8 $A[m, 0] \leftarrow 0$; // $2O(Nn^2) = O(2Nn^2) = O(Nn^2)$
- 9 **for** $i = 1$ to $N - 1$ **do** // $2O(Nn^2) = O(2Nn^2) = O(Nn^2)$
- 10 **for** $m = i + 1$ to $n - 1$ **do**
- 11 **for** $\alpha = i$ to $m - 1$ **do**
- 12 Computing the i^{th} segment *APLA* representation \hat{c}_i
- 13 to points c_α through c_m ; // $O(l)$
- 14 $\bar{\omega}_i \leftarrow$ Computing the maximum deviation of \hat{c}_i ; // $O(m - \alpha)$
- 15 $\bar{\omega}_{\max} \leftarrow \max\{\bar{\omega}[\alpha, i - 1], \bar{\omega}_i\}$;
- 16 **if** $\alpha == i$ or $\bar{\omega}_{\max} < \bar{\omega}[m, i]$ **then**
- 17 $\hat{C}[m, i] \leftarrow \hat{c}_i$;
- 18 $\bar{\omega}[m, i] \leftarrow \bar{\omega}_{\max}$;
- 19 $A[m, i] \leftarrow \alpha$;
- 20 $i \leftarrow N - 1$;
- 21 $m \leftarrow n - 1$;
- 22 **while** $i \geq 0$ **do** // $O(N)$
- 23 $\hat{c}_i \leftarrow \hat{C}[m, i]$;
- 24 $m \leftarrow A[m, i]$;
- 25 $i --$;
- 26 **return** $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$;
- 27 **Time Complexity :** $O(n^2) + O(2Nn^2) + O(N) = O(Nn^2)$

The time complexity of computing the *APLA* representation coefficients a and b of each segment is $O(l)$.

2.2.5 Chebyshev Polynomials Approximation (*CHEBY*) [10]

PAA, *APCA* and *PLA* are discontinuous functions because they divide the original time series into several segments. *CHEBY* is a continuous function. *CHEBY* [9] uses Chebyshev polynomial coefficient che_i to represent the original time series C . The authors declared that Chebyshev coefficients should be smaller than 25. Our evaluation in Section 3.5 shows that *CHEBY* falls into the “dimensionality curse” when $N > 25$. *CHEBY* has $O(Nn)$ time complexity.

If the linear function can be used to represent the original time series, other functions may also have a chance to represent the original time series [63]. Thus, the Chebyshev polynomial is proposed. Chebyshev polynomial is called a minimax polynomial, and is helpful for indexing. The reason for choosing the Chebyshev polynomial is that it is easy to compute and has optimal minimax properties. Furthermore, the most optimal polynomials cost much computing time, while simple polynomials cannot satisfy maximum deviation. Moreover, authors have proposed multi-dimension time series dimensionality reduction [10].

Given a time series $C = \{(t_0, v_0), (t_1, v_1), \dots, (t_{n-1}, v_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) can be regarded as a discrete function. So we need to transfer the time series from a discontinuous function to an interval function. Chebyshev polynomial is $P_m(t) = \cos(m \cos^{-1}(t))$, $t \in [-1, 1]$. We need to normalize the time stamp t into $-1 \leq t_0 < \dots < t_{n-1} \leq 1$. Algorithm 2.2.5 shows the *CHEBY* computation of the algorithm. Time series discrete function is like Eq. (2.7).

$$S(t) = \begin{cases} v_j & \text{if } t = t_j \\ \text{undefined} & \text{otherwise} \end{cases} \quad (2.7)$$

Second, we divide the normalized $t \in [-1, 1]$ into n disjoint segments like Eq. (2.8):

$$I_j = \begin{cases} \left[-1, \frac{t_0 + t_1}{2} \right) & \text{if } j = 0 \\ \left[\frac{t_{j-1} + t_j}{2}, \frac{t_j + t_{j+1}}{2} \right) & \text{if } 1 \leq j \leq n-2 \\ \left[\frac{t_{n-2} + t_{n-1}}{2}, 1 \right] & \text{if } j = n-1 \end{cases} \quad (2.8)$$

When we get several intervals, we make use of point value v_j to represent interval value like Eq.

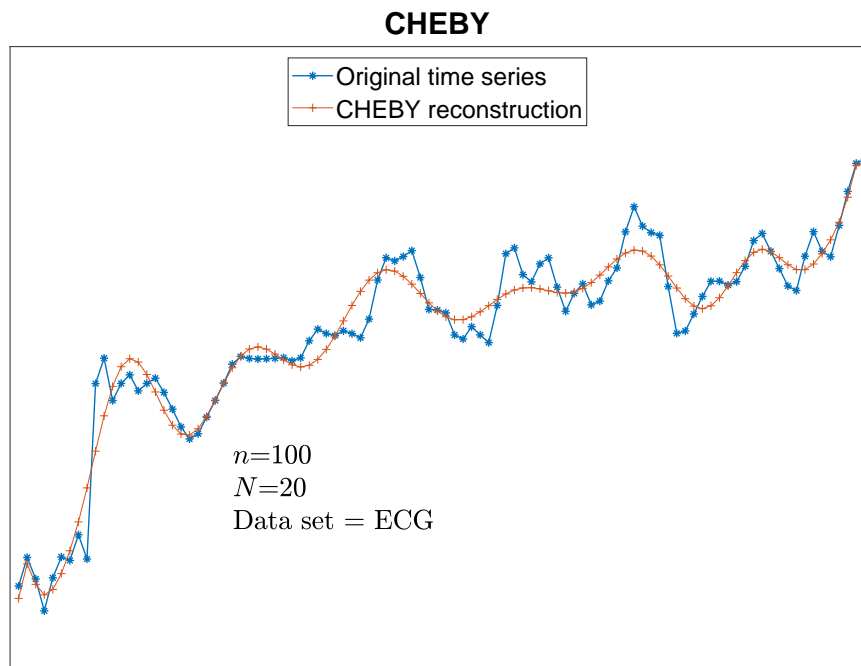


Figure 2.8: The x -axis represents the time, and the y -axis represents the real value. The orange line represents the reconstruction of Chebyshev coefficients.

(2.9).

$$g(t) = v_j, \quad \text{if } t \in I_j, \quad 0 \leq j \leq n-1 \quad (2.9)$$

However, Eq. (2.9) is too simple to satisfy Lower Bounds Lemma. The weight function $w(t) = \frac{1}{\sqrt{1-t^2}}$ and the length of each segment are introduced, like Eq. (2.10).

$$f(t) = \frac{g(t)}{\sqrt{w(t)|I_j|}}, \quad \text{if } t \in I_j, \quad 0 \leq j \leq n-1 \quad (2.10)$$

Because $f(t)$ is an interval function to be represented, we could compute the Chebyshev coefficient for every segment, like Eq. (2.11) and Eq. (2.12).

$$c_0 = \frac{1}{n} \sum_{j=0}^{n-1} f(t_j) P_0(t_j) \quad (2.11)$$

$$c_i = \frac{2}{n} \sum_{j=0}^{n-1} f(t_j) P_i(t_j) \quad (2.12)$$

For every segment $I_i, i \in \{0, 1, \dots, n-1\}$, t_j is the time stamp in Chebyshev polynomial function $P_i(t_j)$, like $t_j = \cos \frac{(j-0.5)\pi}{n}$. However, we only compute first N Chebyshev polynomial coefficients to avoid segment number curses and reduce computing time ($N \leq 25$).

Example 5. (Computing *CHEBY* Dimensionality Reduction). Given a time series $C = \{8, 6, 6, 4, 4, 4, 5, 7\}$. The user defined segment number N is three. First, we normalize the time stamp j into $[-1, 1]$, $t = \{t_0, \dots, t_{n-1}\} = \{-1, -0.714286, -0.428571, -0.142857, 0.142857, 0.428571, 0.714286, 1\}$. We divide the interval $[-1, 1]$ into n disjoint sub-intervals, $I = \{I_0, \dots, I_{n-1}\} = \{[-1, -0.857143), [-0.857143, -0.571429), [-0.571429, -0.285714), [-0.285714, 0), [0, 0.285714), [0.285714, 0.571429), [0.571429, 0.857143), [0.857143, 1]\}$. We compute the time stamp rt in Chebyshev polynomial $P_i(t_j)$, $rt = \{rt_0, \dots, rt_{n-1}\} = \{0.980785, 0.83147, 0.55557, 0.19509, -0.19509, -0.55557, -0.83147, -0.980785\}$. We compute the rt in which interval $I = \{7, 6, 5, 4, 3, 2, 1, 0\}$. We compute the interval function $f = \{8.18022, 6.97226, 6.82366, 7.41107, 7.41107, 10.2355, 8.36671, 9.34882\}$. The final *CHEBY* representation is $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2\} = \{8.09366, -1.05028, 0.460641\}$.

Algorithm 2.2.5: Compute *CHEBY* // $O(Nn)$

```

input :  $C = \{c_0, c_1, \dots, c_{n-1}\}$ : original time series;
 $n$  : length of time series;
 $N$  : number of segment;
output:  $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ : CHEBY representation;
1 for  $j = 0 : n - 1$  do // Normalize the time stamp  $j$  into  $[-1, 1]$ .  $O(n)$ 
2    $t_j \leftarrow 2 * \frac{j-1}{n-1} - 1$ ;
3  $I_0 \leftarrow -1$ ;
4  $I_1 \leftarrow \frac{t_0+t_1}{2}$ ;
5 for  $j = 1 : n - 2$  do // Divide into  $n$  disjoint sub-intervals.  $O(n)$ 
6    $I_{2*j} \leftarrow \frac{t_{j-1}+t_j}{2}$ ;
7    $I_{2*j+1} \leftarrow \frac{t_j+t_{j+1}}{2}$ ;
8  $I_{n*2-2} \leftarrow \frac{t_{n-2}+t_{n-1}}{2}$ ;
9  $I_{n*2-1} \leftarrow 1$ ;
10 for  $j = 1 : n$  do //  $rt_j$  is time stamp in Chebyshev polynomial.  $O(n)$ 
11    $rt_{j-1} \leftarrow \cos \frac{(j-0.5)\pi}{n}$ ;
12 for  $j = 0$  to  $n - 1$  do //  $rt_j$  in which interval.  $O(n)$ 
13   if  $rt_j == 1$  then
14      $rt_j \leftarrow n - 1$ ;
15      $li \leftarrow 0$ ;
16      $ri \leftarrow n - 1$ ;
17     while  $li \leq ri$  do
18        $mi \leftarrow \frac{ri+li}{2}$ ;
19       if  $I_{2*mi} \leq rt_j < I_{2*mi+1}$  then
20          $rt_j \leftarrow mi$ ;
21       else if  $rt_j < I_{2*mi}$  then
22          $ri \leftarrow mi - 1$ ;
23       else
24          $li \leftarrow mi + 1$ ;
25 for  $j = 0 : n - 1$  do //  $f_j$  is an interval function to be represented.  $O(n)$ 
26    $f_j \leftarrow c_{rt_j} \div \sqrt{\frac{I_{2*rt_j+1}-I_{2*rt_j}}{\sqrt{1-rt_j^2}}}$ ;
27 for  $i = 0 : N - 1$  do //  $O(Nn)$ 
28    $\hat{c}_i \leftarrow \frac{2}{n} \sum_{j=0}^{n-1} f_j * P_i[rt_j]$ ; //  $P_i(rt_j)$  is Chebyshev Polynomial;
29   if  $!i$  then
30      $c_i / = 2$ ;
31 return  $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$ ;

```

2.2.6 Piecewise Aggregate Approximation Lagrangian Multipliers (PAALM) [61]

PAALM [61] applies PAA and Lagrangian Multipliers on the original time series C . PAALM has $O(n)$ time complexity. PAALM represents continuous data as a series of patterns, which means it does not focus on max deviation reduction. Thus we will evaluate it in the k -NN search to show the importance of max deviation. Figure 2.9 shows the process of PAALM.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), and a user defined segment number N , PAALM first applies PAA dimensionality reduction method in Section 2.2.1. PAA uses the mean value of each segment by Eq. (2.4) to represent the original time series C . We will get $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ after PAA dimensionality reduction. PAALM uses the Lagrangian multiplier for solving optimisation problems without finding parametric equations for the functions. PAALM has two advantages: reducing the time series length and showing the differences between different time series without scaling problem by using unit vectors.

There is a function $E(u) = E(u_0, u_1, \dots, u_{N-1})$. PAALM tries to find the extreme subject to a single constraint $g(u) = 0$. PAALM defines a new function $L(u, \lambda) = E(u) - \lambda * g(u)$. PAALM then find the extreme L with respect to both x and λ ($\frac{dL}{d\lambda} \leftarrow 0$ and $\frac{dL}{du} \leftarrow 0$). PAALM finds a unit vector $\vec{u} = \{u_0, u_1, \dots, u_{N-1}\}$ for the PAA representation \vec{C} which maximises the dot product ($\vec{u} \cdot \vec{C}$). Eq. 2.13 and Eq. 2.14 show how to compute constraints.

$$\|\vec{u}\| = \sqrt{\sum_{i=0}^{N-1} u_i^2} = 1 \quad (2.13)$$

$$g(u) = \sum_{i=0}^{N-1} u_i^2 - 1 = 0 \quad (2.14)$$

Eq. 2.15 shows the Lagrangian function:

$$\begin{aligned} L(u, \lambda) &= \vec{u} \cdot \vec{C} - \lambda * g(u) \\ &= \sum_{i=0}^{N-1} u_i \hat{c}_i - \lambda * \left(\sum_{i=0}^{N-1} u_i^2 - 1 \right) \end{aligned} \quad (2.15)$$

To solve Eq. 2.15, PAALM set ∇L equal to zero in Eq. 2.16 and Eq. 2.17.

$$\frac{dL}{du_i} = \hat{c}_i - 2\lambda * u_i = 0, \text{ for } i = 0, 1, 2, \dots, N-1 \quad (2.16)$$

$$\frac{dL}{d\lambda} = -\left(\sum_{i=0}^{N-1} u_i^2 - 1\right) = 0 \quad (2.17)$$

PAALM will get Eq. 2.18 by computing Eq. 2.16 and Eq. 2.17. Eq. 2.18 means \vec{u} is proportional to $\vec{\hat{C}}$ and they are in the same direction. *PAALM* will have a unit vector with the properties in Eq. 2.19 by normalising $\vec{\hat{C}}$. Therefore, \vec{u} is a unit vector projection of *PAA* representation $\vec{\hat{C}}$.

$$u_i = \frac{1}{2\lambda} \hat{c}_i, \text{ for } i = 0, 1, 2, \dots, N-1 \quad (2.18)$$

$$\vec{u} = \frac{\vec{\hat{C}}}{\|\vec{\hat{C}}\|} \quad (2.19)$$

Algorithm 2.2.6: Lagrangian representation algorithm in *PAALM*

input : *PAA*: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$;

output: u ;

1 $N \leftarrow$ segment number in *PAA*

2 unit vector variable: $u = \{u_0, u_1, \dots, u_{N-1}\}$

3 $g(u_0, u_1, \dots, u_{N-1}) \leftarrow u_0^2 + u_1^2 + \dots + u_{N-1}^2 - 1$

4 $f(\hat{C}, u) \leftarrow \hat{c}_0 u_0 + \hat{c}_1 u_1 + \dots + \hat{c}_{N-1} u_{N-1}$

5 $L(\hat{C}, u, \lambda) \leftarrow f(\hat{C}, u) - \lambda * g(u)$

6 **for** $i = 0 : N - 1$ **do**

7 $\text{partial_derivative}_i = \hat{c}_i - 2\lambda * u_i$

8 $\text{partial_derivative}_\lambda = -u_0^2 - u_1^2 - \dots - u_{N-1}^2 + 1$

9 $u \leftarrow \text{Solve}(\text{partial_derivative}_0 = 0, \dots, \text{partial_derivative}_{N-1} = 0, \text{partial_derivative}_\lambda = 0)$

Example 6. (Computing *PAALM* Dimensionality Reduction). Figure 2.9 shows an example of *PAALM*. The user defined segment number N is nine. Given a *PAA* representation $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_8\} = \{0.093, 1.243, -0.181, 0.617, 1.531, -0.086, -1.113, -0.771, -1.070\}$. First, we compute the Lagrangian multiplier $\lambda = 1 \div 2 \sqrt{\frac{\sum_{i=0}^{N-1} (\hat{c}_i^2)}{4}} \leftarrow 0.37$. The final *PAALM* representation is $\hat{C}' = \{\hat{c}'_0, \hat{c}'_1, \dots, \hat{c}'_8\} = \{\lambda * \hat{c}_0, \lambda * \hat{c}_1, \dots, \lambda * \hat{c}_8\} = \{0.034, 0.46, -0.067, 0.228, 0.567, -0.032, -0.412, -0.285, -0.396\}$.

The change point detection for the original time series C is one of the key issues. Some change point detection methods identify a region where the change happens. Some change point detection methods identify the point in the section that the stationary proprieties change. The cumulative sum (*CUSUM*) [44] is based on the self-starting method that integrates the *CUSUM* of the Q chart and

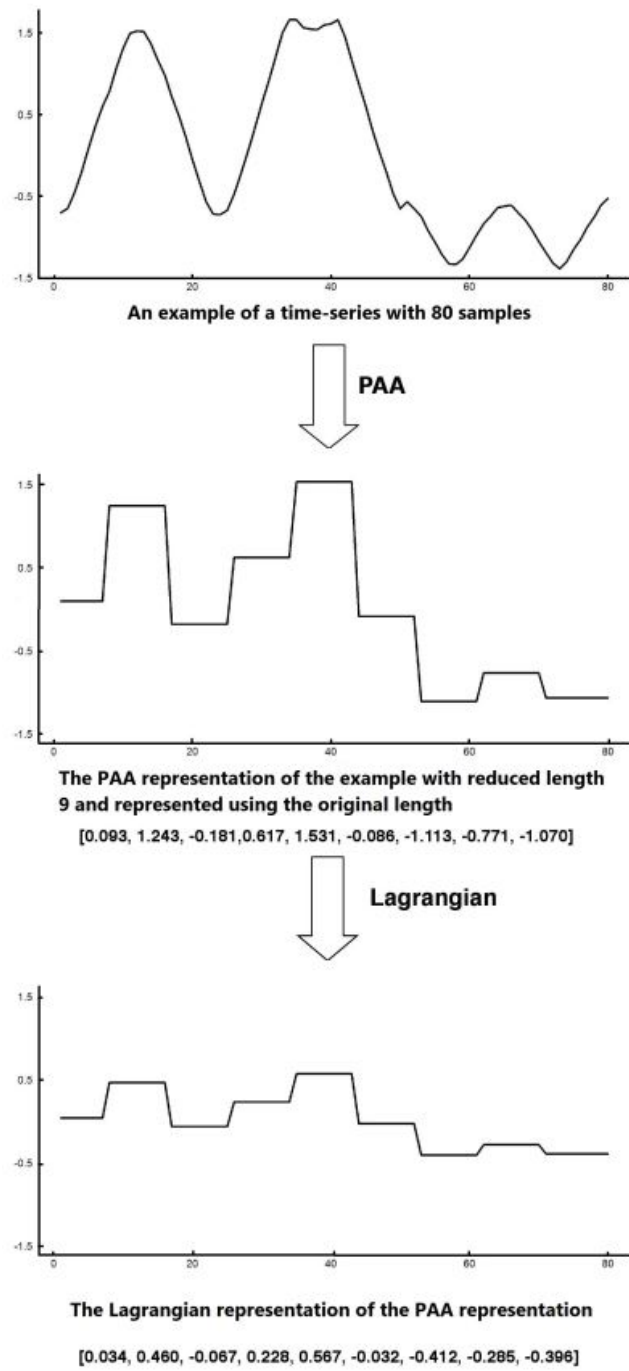


Figure 2.9: [61] One example of PAALM.

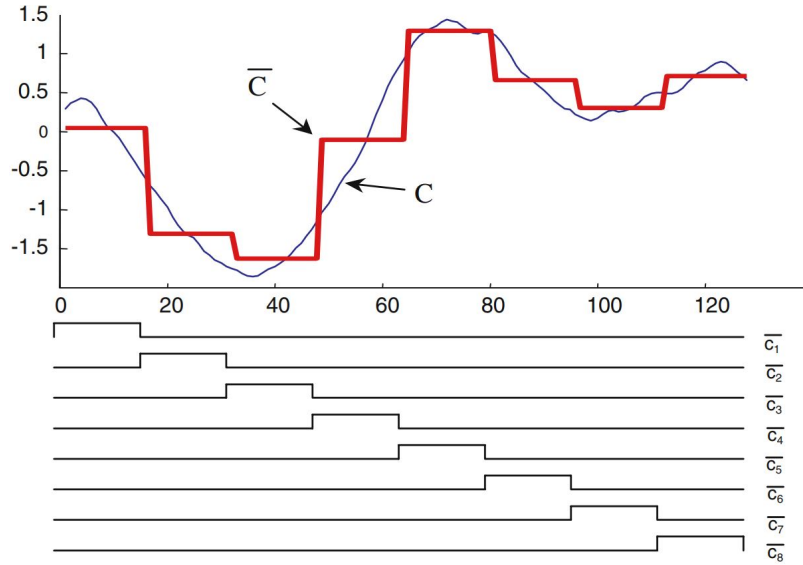


Figure 2.10: [46] The PAA can be regarded as an attempt to model an original time series with a linear combination of box basis functions. The length of the original time series is 128. The segment number is eight.

the feature of adaptively varying the reference value. *CUSUM* uses the probability distribution and the threshold to detect the change. The Bayesian change point detection method divides the original time series C into several segments with a probability distribution [2]. Kullback-Leibler Importance Estimation Procedure (*KLIEP*) [47] finds abrupt changes in the properties of the original time series. We focus on reducing the maximum deviation of the dimensionality reduction method from the user defined segment number N in this paper. Our proposed algorithms are based on the *GEMINI* structure. So, the change point detection methods are out of the scope of this thesis.

2.2.7 Symbolic Aggregate Approximation (SAX) [45]

SAX [60] first transforms the original time series C into *PAA* representation and then symbolizes the *PAA* into a discrete string. *SAX* has $O(n)$ time complexity. Figure 2.11 shows the process of *SAX*.

Given a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), and a user defined segment number N , *SAX* first applies the *PAA* dimensionality reduction method in Section 2.2.1. *PAA* uses the mean value of each segment by Eq. (2.4) to represent the original time series C . We will get $\hat{C} = \{\hat{c}_1, \hat{c}_2, \dots, \hat{c}_N\}$ after *PAA* dimensionality reduction. *SAX* could be visualized as an attempt to represent the C with a linear combination of box basis functions as shown in Figure 2.10.

We could apply a further transformation for *PAA* to get a discrete representation. If the normalized time series have a Gaussian distribution, *SAX* will produce symbols with equiprobability.

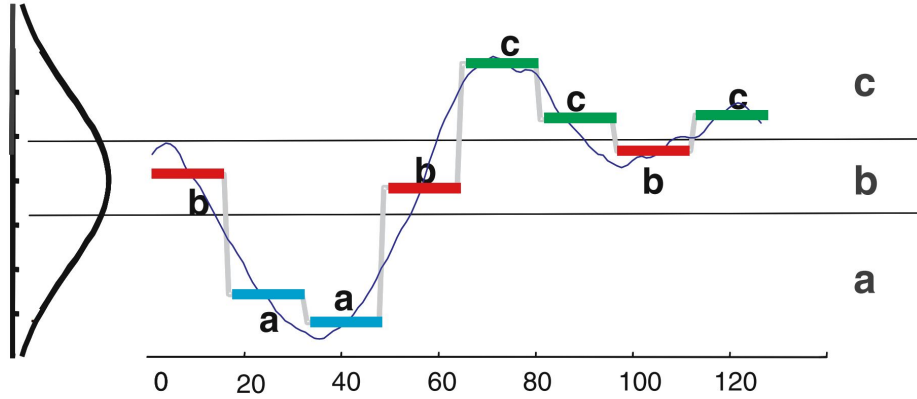


Figure 2.11: [46] One example of SAX.

If the time series do not have Gaussian distribution, the representation efficiency is slightly deteriorated; however, the correctness of the SAX is unaffected [45]. SAX decides the “breakpoints” which will produce an equal-sized area under the Gaussian curve. Breakpoints are a list of values $\{\Psi_i, \dots, \Psi_{H-1}\}$ on the y-axis that the area under the Gaussian curve. For example, Table 2.3 gives the breakpoints for values of $H \in [3, 10]$.

Ψ_i	3	4	5	6	7	8	9	10
Ψ_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
Ψ_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
Ψ_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
Ψ_4			0.84	0.43	0.18	0	-0.14	-0.25
Ψ_5				0.97	0.57	0.32	0.14	0
Ψ_6					1.07	0.67	0.43	0.25
Ψ_7						1.15	0.76	0.52
Ψ_8							1.22	0.84
Ψ_9								1.28

Table 2.3: [46] contains the breakpoints that divide a Gaussian distribution $([3, 10])$ of equiprobable regions.

Example 7. (Computing SAX Dimensionality Reduction). Given a time series $C = \{1.65359, 0.330719, 0.330719, -0.992157, -0.992157, -0.992157, -0.330719, 0.992157\}$. The user defined segment number N is four. The PAA representation $\bar{C} = \{\bar{c}_0, \bar{c}_1, \bar{c}_2, \bar{c}_3\} = \{0.992, -0.33, -0.992, 0.33\}$. The break point list is $\Psi = \{-inf, -0.67, 0, 0.67\}$. The representation coefficient list is $alpha = \{a, b, c, d\}$. The average value of the time series C is $\mu = 0$. The standard deviation of the time series C is $\sigma = 0.85$. The normalized PAA representation is $\bar{C} = \{\bar{c}_0, \bar{c}_1, \bar{c}_2, \bar{c}_3\} = \{1.1619, -0.387298, -1.1619, 0.387298\}$. The final SAX representation is $\hat{C} = \{\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3\} = \{d, b, a, c\}$.

Algorithm 2.2.7: Compute *SAX* // $O(n)$

input : $C = \{c_0, c_1, \dots, c_{n-1}\}$: original time series;
 n : length of time series;
 N : number of segment;
output: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$: *SAX* representation;

- 1 $alpha := \{a, b, c, \dots, z\}$;
- 2 Transfer the original time series C into *PAA* representation $\bar{C} = \{\bar{c}_0, \bar{c}_1, \dots, \bar{c}_{N-1}\}$ by
Algorithm 2.2.1; // $O(n)$
- 3 $\Psi.size() \leftarrow N$;
- 4 $\mu \leftarrow \frac{\sum_{i=0}^{N-1} \bar{c}_i}{N}$; // $O(N)$
- 5 $\sigma \leftarrow \sqrt{\frac{\sum_{i=0}^{N-1} (\bar{c}_i - \mu)^2}{N}}$; // $O(N)$
- 6 **for** $i = 0 : N - 1$ **do** // $O(N)$
- 7 $\bar{c}_i \leftarrow \frac{\bar{c}_i - \mu}{\sigma}$;
- 8 **for** $i = 0 : N - 1$ **do** // $O(N)$
- 9 **for** $j = 1 : N - 1$ **do**
- 10 **if** $\Psi_{j-1} \leq \bar{c}_i < \Psi_j$ **then**
- 11 $\hat{c}_i \leftarrow alpha_j$;
- 12 **return** $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$;

The *SAX* dimensionality reduction method is extended to Indexable Symbolic Aggregate Approximation (*iSAX*) [66]. The *SAX* representation coefficients are presented as binary forms, such as $SAX(C, N, a) = \{11, 10, 01, 00\}$. The C is original time series with length n . The N is the number of representation coefficients. The a is the size of cardinality. *iSAX* supports the indexing of massive datasets. Extension of Symbolic Aggregate Approximation (*Extended SAX*) [49]. *SAX* is based on the *PAA* dimensionality reduction method. The average value based representation misses important information in some original time series, such as financial time series. *Extended SAX* uses the minimum, maximum and average values as string of symbols to represent the shape in each segment. The Enhanced Symbolic Aggregate Approximation (*Enhanced SAX*) [4] uses the minimum, maximum and average values as the vector of time series data points. Segmentation Based Symbolic Representations (*SBSR*) [35] uses the symbolic alphabet and the episode boundaries to keep more information during the dimensionality reduction process. *Persist* [53] is an unsupervised representation method. *Persist* represents the original time series to maximize the persistence of each symbol. Symbolic Fourier Approximation (*SFA*) [64] is based on Discrete Fourier Transform (*DFT*) and multiple coefficient binning for dimensionality reduction. Genetic Algorithms-Based Symbolic Aggregate Approximation (*GASAX*) [27] does not require the “high Gaussianity” of the original normalized time series. *GASAX* applies genetic algorithms to find the breakpoints in di-

dimensionality reduction process. Adaptive SAX (*aSAX*) [57] uses the average value of each segment as training input and the k -means algorithm to get adaptive “breakpoints”. Adaptive Brownian Bridge-Based Symbolic Aggregation (*ABBA*) [21] uses the adaptive polygonal chain to represent the original time series C into a sequence of tuples and uses mean-based clustering to obtain the symbolic representation.

2.3 Lower Bounding Distance Measures for Dimensionality Reduction Methods

Similarity search methods need effective distance measures to compare the similarity of two time series. The above dimensionality reduction methods and distance measurements are all applied to the whole sequence match, which means all time series are compared with the same length. R-tree is proposed [30] and could be used for spatial access methods. k -NN [40] search algorithm is proposed for similarity search.

We have defined the lower bounding lemma in Lemma 1. There are many lower bounding measures for equal-length segment dimensionality reduction methods, such as $Dist_{PAA}$ [39], $Dist_{PLA}$ [15], $Dist_{SAX}$ [46] and $Dist_{CHEBY}$ [9]. The lower bounding measure for adaptive-length segment dimensionality reduction methods is complicated. *APCA* [13] proposes $Dist_{AE}$ for a tight approximation but not always lower bound the Euclidean distance and $Dist_{LB}$ for a less tight approximation but can guarantee a lower bound of the Euclidean distance. *PAALM* uses the $Dist_{PAA}$ for lower bounding distance computation. *APLA* and our proposed *SAPLA* use the $Dist_{LB}$ as the baseline lower bounding distance computation.

($Dist_{PAA}$) Suppose we get two *PAA* representations $\hat{Q} = \{\hat{q}_0, \hat{q}_1, \dots, \hat{q}_{N-1}\}$, and $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$. The lower bounds distance $Dist_{PAA}(\hat{Q}, \hat{C})$ between them is defined by Eq. (2.20).

$$Dist_{PAA}(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{N} \sum_{i=0}^{N-1} (\hat{q}_i - \hat{c}_i)^2} \quad (2.20)$$

($Dist_{PLA}$) Suppose we get two *PLA* representations $\hat{Q} = \{\langle a_{10}, b_{10} \rangle, \langle a_{11}, b_{11} \rangle, \dots, \langle a_{1N-1}, b_{1N-1} \rangle\}$ and $\hat{C} = \{\langle a_{20}, b_{20} \rangle, \langle a_{21}, b_{21} \rangle, \dots, \langle a_{2N-1}, b_{2N-1} \rangle\}$, The lower bounds distance $dist_{PLA}(\hat{Q}, \hat{C})$ between them is defined by Eq. (2.21).

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Table 2.4: A lookup table used by the $Dist_{SAX}$ function

$$Dist_{PLA}(\hat{Q}, \hat{C}) = \sqrt{\sum_{i=0}^{N-1} \sum_{j=0}^{\frac{n}{N}-1} ((a_{1i} - a_{2i}) \times j + b_{1i} - b_{2i})^2} \quad (2.21)$$

($Dist_{CHEBY}$) There are two time series Q and C . After Chebyshev polynomial dimensionality reduction, we can get Chebyshev coefficients $\hat{Q} = \{\hat{q}_0, \hat{q}_1, \dots, \hat{q}_{N-1}\}$ and $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$. $Dist_{CHEBY}$ is shown in Eq. (2.22). $\frac{\pi}{2}$ is the weight function to satisfy the lower bound lemma.

$$Dist_{CHEBY}(\hat{Q}, \hat{C}) = \sqrt{\frac{\pi}{2} \sum_{i=0}^{N-1} (\hat{q}_i - \hat{c}_i)^2} \quad (2.22)$$

($Dist_{SAX}$) There are two original time series Q and C . we will get $\hat{Q} = \{\hat{q}_0, \hat{q}_1, \dots, \hat{q}_{N-1}\}$ and $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ after symbolic representation. $Dist_{SAX}$ is shown in Eq. 2.23. The $dist()$ [46, 26] function could be implemented by applying a table which is illustrated in Table 2.4. This table is for an alphabet of the cardinality of four, such as d is four. The distance between two symbols can be got by finding the corresponding row and column. For example, $dist(a, b) = 0$ and $dist(a, c) = 0.67$.

$$Dist_{SAX}(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{N} \sum_{i=0}^{N-1} dist(\hat{q}_i, \hat{c}_i)^2} \quad (2.23)$$

($Dist_{AE}$ [40]) $Dist_{AE}$ is used for adaptive-length segment dimensionality reduction methods. $Dist_{AE}$ has tight distance approximation but not always lower bound the Euclidean distance. For an original time series $Q = \{(t_0, q_0), (t_1, q_1), \dots, (t_{n-1}, q_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) and an APCA representation $\hat{C} = \{\langle cv_0, cr_0 \rangle, \langle cv_1, cr_1 \rangle, \dots, \langle cv_{N-1}, cr_{N-1} \rangle\}$ representation, Eq. (2.24) shows how to compute $Dist_{AE}$. Because $Dist_{AE}$ needs to scan each point in the original time series C , the time complexity of distance computation is $O(n)$.

$$Dist_{AE}(Q, \hat{C}) = \sqrt{\sum_{i=1}^N \sum_{k=1}^{cr_i - cr_{i-1}} (q_{k+cr_{i-1}} - cv_i)^2} \quad (2.24)$$

Because $Dist_{AE}$ computes the distance between one original time series and the reconstruction

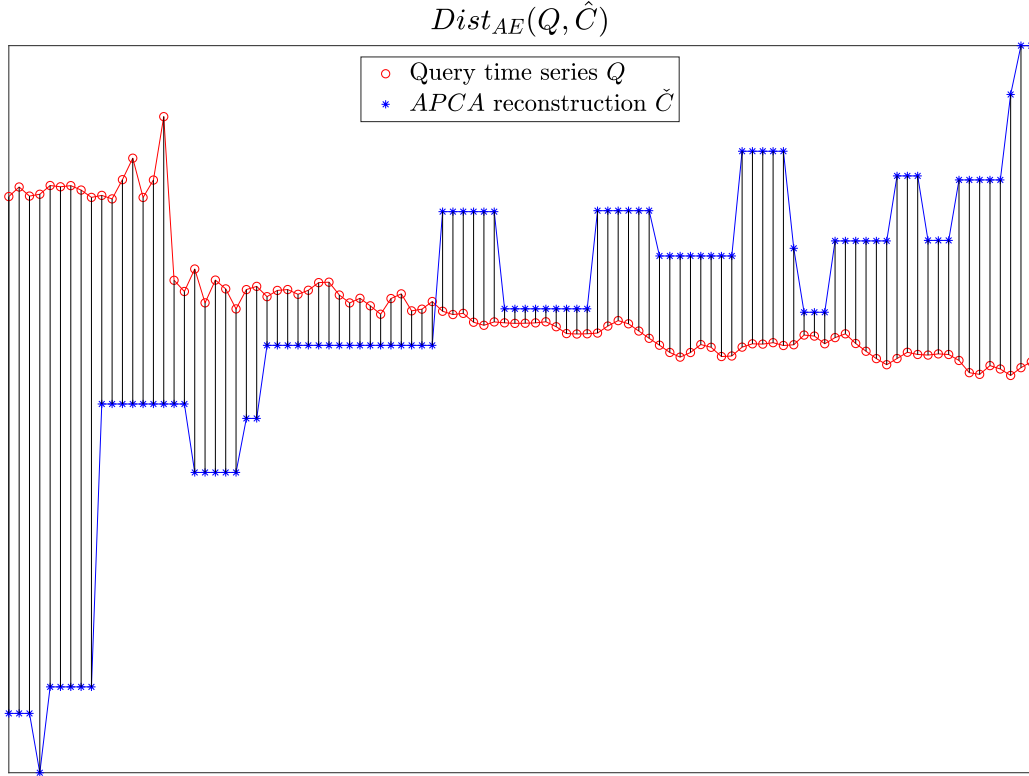


Figure 2.12: The $Dist_{AE}$ measure can be visualized as the Euclidean distance between Q and the reconstructed time series of \hat{C} .

of representation coefficients, it can tightly approximate the Euclidean distance. It does not satisfy the triangular inequality [40].

($Dist_{LB}$ [40]) $Dist_{LB}$ is used for adaptive-length segment dimensionality reduction methods. $Dist_{LB}$ is a less tight approximation of the Euclidean distance and follows the lower bound lemma. Given an original query time series $Q = \{q_0, q_1, \dots, q_{n-1}\}$, and an APCA representation $\hat{C} = \{\langle cv_0, cr_0 \rangle, \langle cv_1, cr_1 \rangle, \dots, \langle cv_{N-1}, cr_{N-1} \rangle\}$. First, we need to map the segment endpoint r_i of the APCA representation \hat{C} onto the original time series Q , which means we will get an APCA representation \hat{Q} . It looks like $\hat{Q} = \{\langle qv_0, qr_0 \rangle, \langle qv_1, qr_1 \rangle, \dots, \langle qv_{N-1}, qr_{N-1} \rangle\}$. \hat{C} and \hat{Q} have the same segment right endpoints, which means $\{qr_0 = cr_0, qr_1 = cr_1, \dots, qr_{N-1} = cr_{N-1}\}$. The time complexity to get an APCA representation \hat{Q} is $O(n)$. The time complexity of $Dist_{LB}$ computation is $O(N)$.

$$Dist_{LB}(\hat{Q}, \hat{C}) = \sqrt{\sum_{i=0}^{N-1} (qr_i - qr_{i-1})(qv_i - cv_i)^2} \quad (2.25)$$

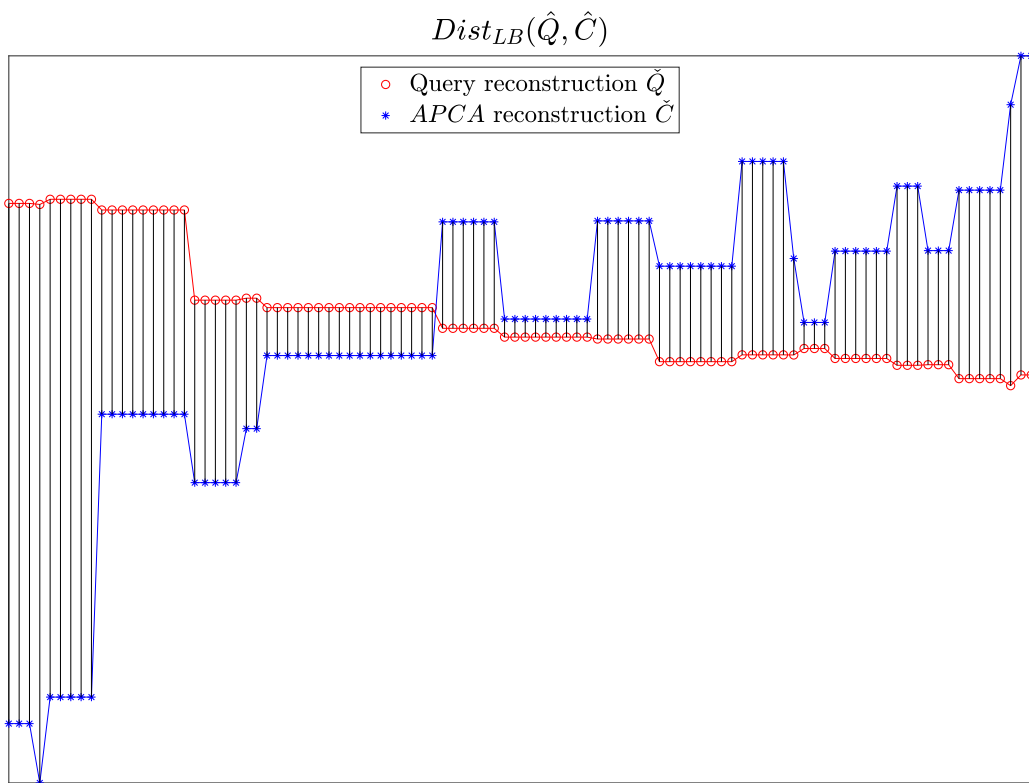


Figure 2.13: \hat{Q} is obtained by projecting the endpoints of \hat{C} onto Q and calculating the mean values of the sections falling within the projected lines.

2.4 Index Structure & k -NN Search

For the index building method, R-tree is a height-balanced tree extended from B-tree. It is better for spatial indexing. The “R” in R-tree means the rectangle. The leaf node of the R-tree records the identifier of every time series. The non-leaf node records the identifier of the child nodes and the bounding box of all datasets within the child node. In Figure 2.14, we introduce one R-tree index structure. Figure 2.15 and Figure 2.16 show the visualization of the R-tree for 2-D rectangles and 3-D data points. The 2-D rectangle is called the minimum bounding rectangle. The minimum bounding rectangle (*MBR*) of the polygon is a common set-up for polygon data. We store the original time series in hardware and store the R-tree index structure in memory in this thesis. If datasets are stored in an index structure, search requirements will cost fewer disk pages according to their locations in the index structure [30]. For similarity search methods, the K nearest neighbours (k -NN) search method is responsible for searching K most similar time series and applies top-down traverse at every step by distance measures [33].

The segment style of *PLA* and *CHEBY* differs from *PAA* and *APCA*. *PAA* and *APCA* use mean values to represent the original time series. However, *PLA* and *CHEBY* use coefficients to represent the original time series. k -NN is instance-based learning and among the simplest of all machine learning algorithms. There are two varieties of k -NN algorithms. The first is shown in Algorithm 2.4.1, which is applied to *SAPLA*, *APLA*, *PAALM*, *SAX*, *PAA*, *APCA* and *PLA*. The second is shown in Algorithm 2.4.3, which is applied to *CHEBY*.

2.4.1 R-tree Indexing & k -NN Search for *APCA*

For *APCA* dimensionality reduction, a time series $C = \{(t_0, v_0), (t_1, v_1), \dots, (t_{n-1}, v_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$) is represented to $\hat{C} = \{\langle cv_0, cr_0 \rangle, \langle cv_1, cr_1 \rangle, \dots, \langle cv_{N-1}, cr_{N-1} \rangle\}$. *APCA* regards the N -dimensional space as the *APCA* space and the points as *APCA* points. For Algorithm 2.4.1, the distance between *APCA* point \hat{C} and \hat{Q} is defined by $Dist_{LB}(\hat{Q}, \hat{C})$. While the distance between non-leaf points U and Q is defined by the minimum distance, $MINDIST(Q, R)$. R is called the minimum bounding rectangle (*MBR*). Because *PAA* can be regarded as *APCA* with equal size segment. Thus *PAA* and *APCA* can use the same index structure.

In R-tree index, the entry of points consisting a minimum bounding rectangle (*MBR*) [31, 40]. Figure 2.17a shows an example of one *MBR*. The original time series is $C = \{c_0, \dots, c_{19}\} = \{7, 8, 20, 15, 18, 8, 8, 15, 10, 1, 4, 3, 3, 5, 4, 9, 2, 9, 10, 10\}$. The length of time series is twenty.

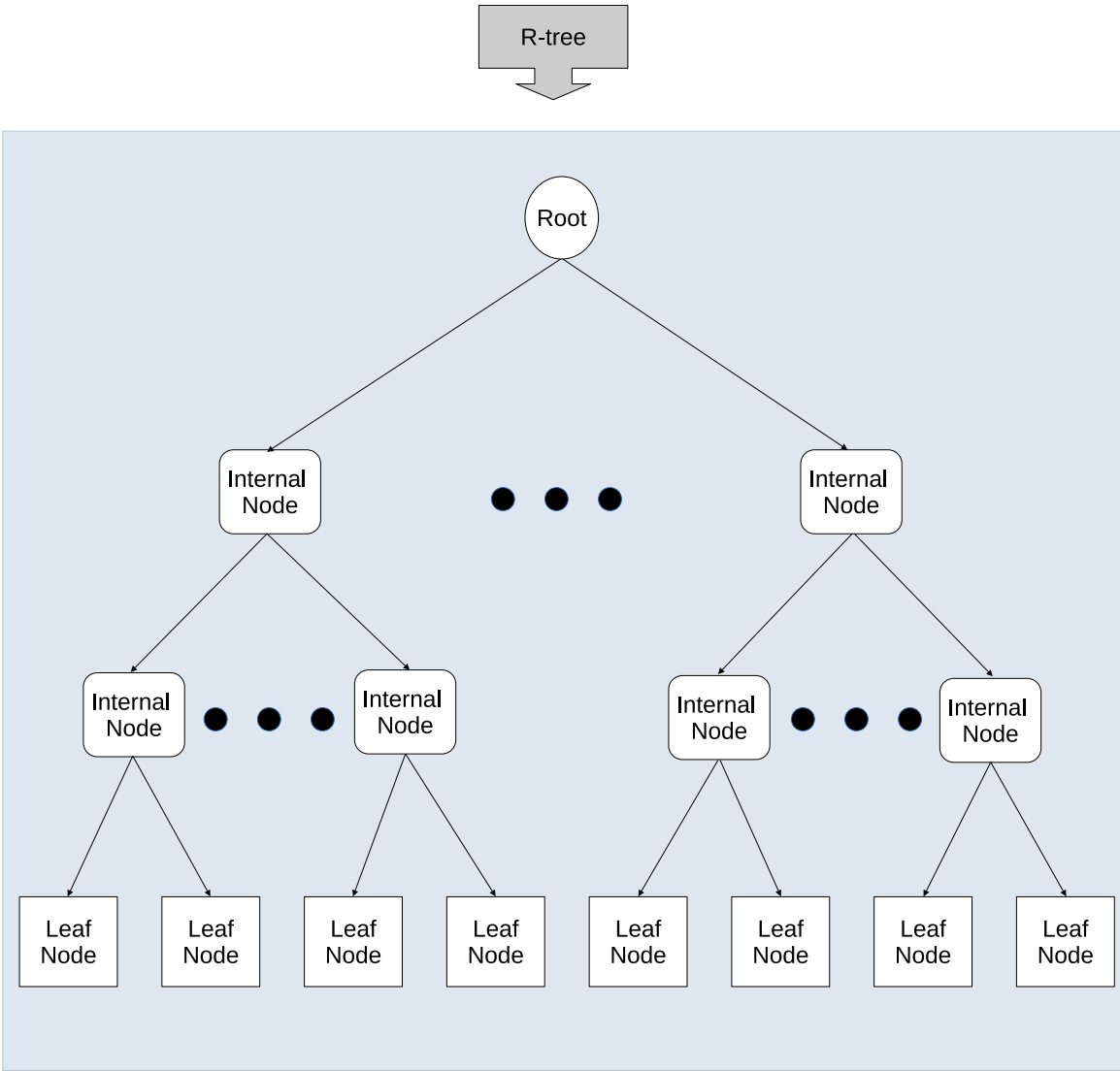


Figure 2.14: R-tree index structure.

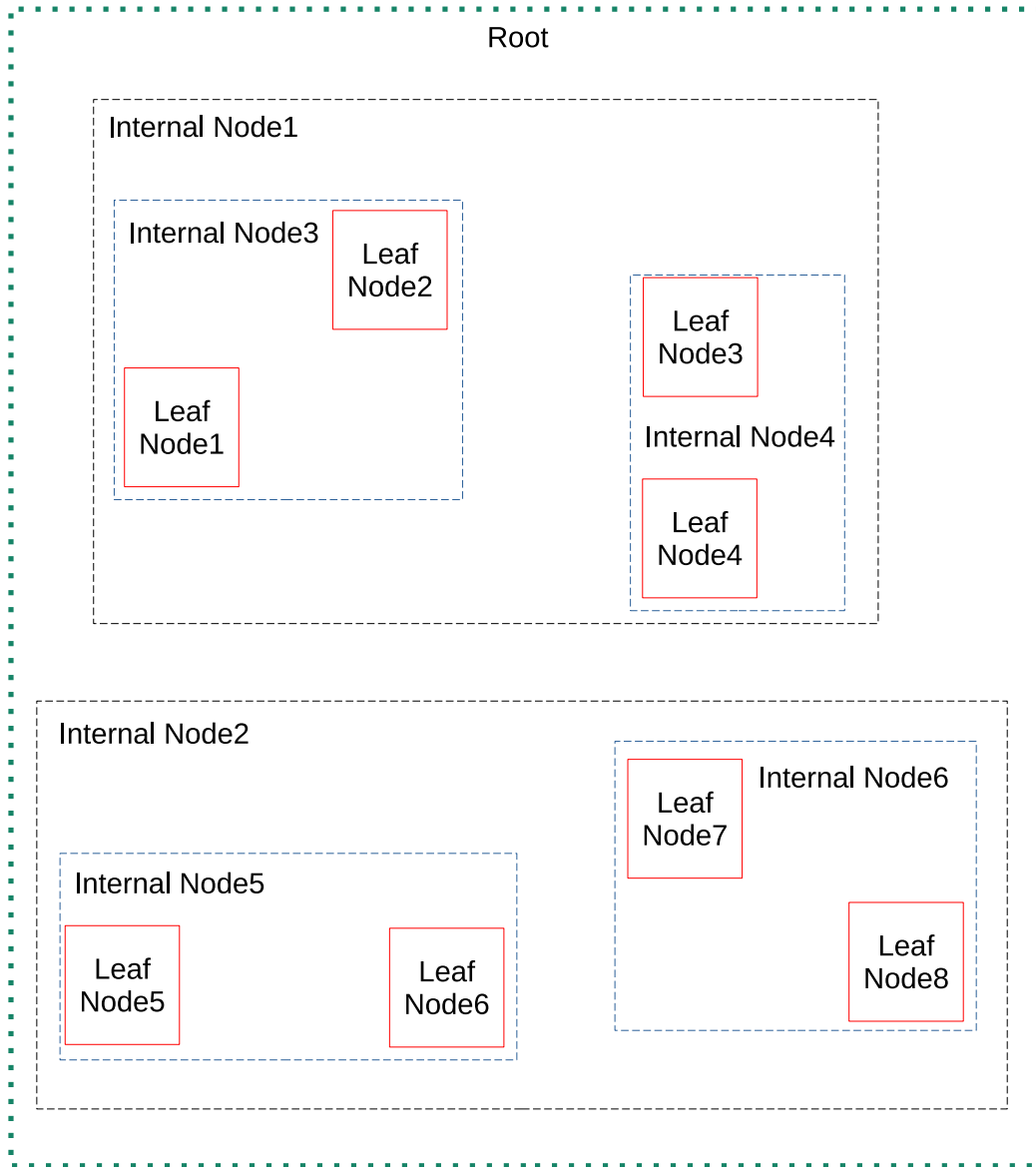


Figure 2.15: An R-tree with 2-D rectangles.

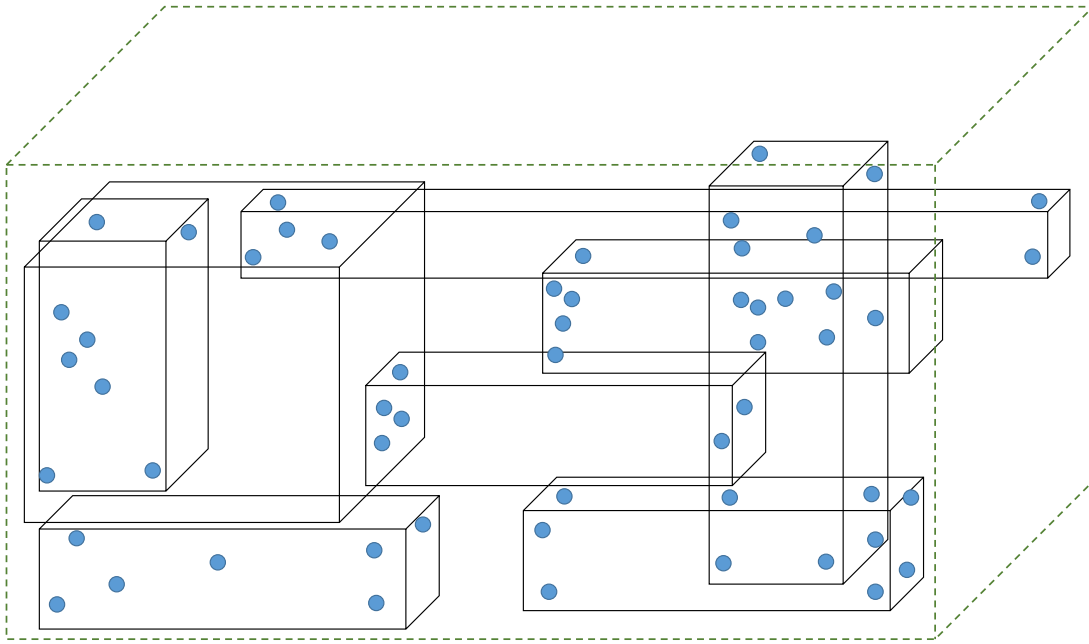


Figure 2.16: An R-tree with 3-D points.

Algorithm 2.4.1: k -NN Search(Q, K) for APCA

```

input : Variable queue: MinPriorityQueue;
        Temp_queue: MinPriorityQueue;
        Result: list;
1 queue.push(root_node_of_index,0);
2 while not queue.IsEmpty() do
3   top = queue.Top();
4   while not temp_queue.IsEmpty() and temp_queue.top.dist  $\leq$  top.dist do
5     Add temp_queue.top to result;
6     if |result| =  $K$  then return result;
7     temp_queue.pop();
8   queue.pop();
9   if top is an APCA point  $C'$  then
10    Retrieve full time series  $C$  from database;
11    temp_queue.push( $C, Dist_{euc}(Q, C)$ );
12  else if top is a leaf node then
13    foreach data item  $C$  in top do queue.push( $C, D_{LB}(Q', C')$ );
14  else top is a non-leaf node
15    for each child node  $U$  in top do
16    | queue.push( $U, MINDIST(Q, R)$ ) //  $R$  is MBR of  $U$ 

```

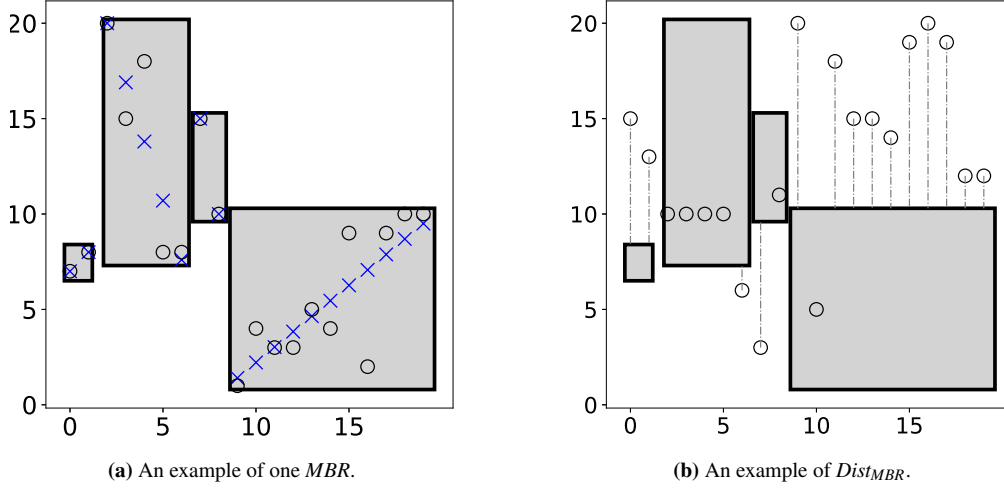


Figure 2.17: The white colour circle \circ is the original time series point. The length of time series is twenty. The reduced dimension is four. \times is a reconstructed time series by *SAPLA* representation. (a) shows an example of one time series *MBR*. It is consist of 4 gray color rectangles \blacksquare s, denoted as $G = \{G_i[0], G_i[1], G_i[2], G_i[3]\}$, ($0 \leq i \leq 3$). (b) shows an example of distance between one time series and *MBR*, denoted as $Dist_{MBR}$.

It was reduced to four dimensions by *SAPLA* dimensionality reduction method and the *SAPLA* representation is $\hat{C} = \{\langle a_0, b_0, r_0 \rangle, \langle a_1, b_1, r_1 \rangle, \langle a_2, b_2, r_2 \rangle, \langle a_3, b_3, r_3 \rangle\} = \{\langle 1, 7, 1 \rangle, \langle -3.09, 20, 6 \rangle, \langle -5, 15, 8 \rangle, \langle 0.809, 1.409, 19 \rangle\}$. So the *MBR* [31, 40] of this *SAPLA* point is denoted as $G = \{G_i[0], G_i[1], G_i[2], G_i[3]\}$, ($0 \leq i \leq 3$). Let i denote the i^{th} segment. Each segment is consist of maximum point $\max_{t=r_{i-1}+1}^{r_i} c_t$, minimum point $\min_{t=r_{i-1}+1}^{r_i} c_t$ and segment length l_i , which means $G_i[0] = \min_{t=r_{i-1}+1}^{r_i} c_t$, $G_i[1] = r_{i-1} + 1$, $G_i[2] = \max_{t=r_{i-1}+1}^{r_i} c_t$ and $G_i[3] = r_i$. For example, the first segment is $\hat{c}_0 = \langle a_0, b_0, r_0 \rangle = \langle 1, 7, 1 \rangle$ and the *MBR* of the first segment is $\{G_0[0], G_0[1], G_0[2], G_0[3]\} = \{7, 0, 8, 1\}$. Figure 2.17b shows an example of distance between one time series and one *MBR*, denoted as $Dist_{MBR}$ [31, 40]. We denote one *MBR* as G . If the time series point locates in G , the point difference is 0. If the time series point locates outside G , we will compute the difference between the time series point and the border line of G .

2.4.2 R-tree Indexing & k -NN Search for *PLA*

PLA makes use of representation coefficients to reconstruct the original time series. So, according to our implementation, we first regard the coefficient as both the maximum and minimum value, then insert coefficients into R-tree. They look like $rec.max = \{\langle a_0, b_0 \rangle, \langle a_1, b_1 \rangle, \dots, \langle a_{N-1}, b_{N-1} \rangle\}$ and $rec.min = \{\langle a_0, b_0 \rangle, \langle a_1, b_1 \rangle, \dots, \langle a_{N-1}, b_{N-1} \rangle\}$. Let m denote one *MBR* node in one *PLA* indexing structure. The distance between one represented query time series Q_{PLA} q and one m node needs a detailed explanation. Because *MBR* m is the boundary of several *PLA* representations, and $Dist_{MBR}$

is the sum of every segment minimum $Dist_{PLA}$, it looks like $Dist_{MBR} = \sum_{i=0}^{N-1} Dist_{PLA}(q^{(i)}, x^{(i)})$ $x^{(i)} \in m$. $x^{(i)}$ is the closest segment in m with $q^{(i)}$, where $x_a \in [a_{min}, a_{max}]$, $x_b \in [b_{min}, b_{max}]$. Eq. (2.26) ($i = 0, \dots, N-1$) shows how to compute segment Dis_{PLA} .

$$\begin{aligned} dist_{PLA}^2(q^{(i)}, x^{(i)}) &= \left(\underbrace{\sqrt{l} \frac{l-1}{2} (x_a - q_a)}_{u_A} - \underbrace{\sqrt{l} (-x_b + q_b)}_{u_B} \right)^2 + \underbrace{\frac{l^3-l}{12} (x_a - q_a)^2}_{v_A^2} \\ &= (u_A - u_B)^2 + (v_A - v_B)^2 \end{aligned} \quad (2.26)$$

$$u_A = \sqrt{l} \frac{l-1}{2} (x_a - q_a) \quad (2.27)$$

$$u_B = \sqrt{l} (-x_b + q_b) \quad (2.28)$$

$$v_A = \sqrt{\frac{l^3-l}{12}} (x_a - q_a) \quad (2.29)$$

$$v_B = 0 \quad (2.30)$$

Eq. (2.26) can be transformed to Euclidean distance between two points, $A(u_A, v_A)$ and $B(u_B, v_B)$. Through Eq. (2.27)-(2.30), point A can consist of a line segment $v = (\sqrt{\frac{l^2-1}{12}} / \frac{l-1}{2}) \times u$. $u \in [\sqrt{l} \times \frac{l+1}{2} \times (a_{min} - q_a), \sqrt{l} \times \frac{l+1}{2} \times (a_{max} - q_a)]$. In the $u-v$ space, point B is a horizontal segment lying on the u -axis. The scale of this segment is $u \in [\sqrt{l} \times (-b_{max} + q_b), \sqrt{l} \times (-b_{min} + q_b)]$. So $dist_{PLA}^2(q^{(i)}, x^{(i)})$ can be transferred to line segments minimum distance in $u-v$ space. Figure 2.18 shows an example of two line segments, A_1A_2 and B_1B_2 are corresponding to points A and B [15]. We define $A_1(u_{A1}, v_{A1})$ as the bottom-left point of segment A_1A_2 , $A_2(u_{A2}, v_{A2})$ as the top-right point. $B_1(u_{B1}, 0)$ as the left point of the segment B_1B_2 , $B_2(u_{B2}, 0)$ as the right point. PLA classifies their relative position into three major cases [15] to get the minimum distance between segments A_1A_2 and B_1B_2 . Segment A_1A_2 only falls into the first and third quadrants in the $u-v$ space. For cases 1-3, they further divide every case into five subcases, shown in Figure 2.18.

- **Case 1:** Segment A_1A_2 is completely contained in the third quadrant of $u-v$ space.
- **Case 2:** Segment A_1A_2 is partly contained in the first and third quadrants of $u-v$ space.
- **Case 3:** Segment A_1A_2 is completely contained in the first quadrant of $u-v$ space.

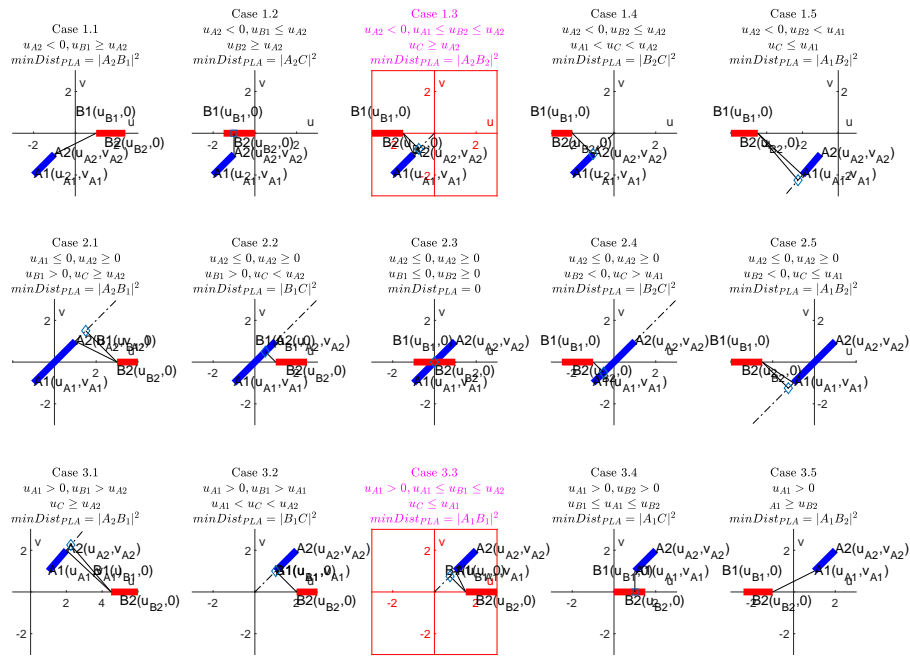


Figure 2.18: Three cases: line Segment A_1A_2 is completely in the third quadrant, partially in the first and third quadrants, or completely in the first quadrants. Case 1.3 and Case 3.3 miss special cases, as Figure 2.21 shows.

2.4.3 R-tree Indexing & k -NN Search for CHEBY

We have introduced the lower bounding distance computation $Dist_{CHEBY}(\hat{Q}, \hat{C})$. When we get the CHEBY representation coefficients $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ from the original time series $C = \{(t_0, v_0), (t_1, v_1), \dots, (t_{n-1}, v_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$). We regard the Chebyshev coefficients as both maximum and minimum value, then we insert these coefficients into R-tree. They look like $rec.max = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ and $rec.min = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$. The range search Algorithm 2.4.2 is applied in CHEBY k -NN search Algorithm 2.4.3.

Algorithm 2.4.2: RangeSearch($Q, Index, r$) [9]

input : Q : a d -dimensional query trajectory;

$Index$: the index of Chebyshev coefficients;

r : a radius for range search.

output: all trajectories within a distance r from Q with respect to $Dist_{euc}$

- 1 Apply Eq. (2.9) to Eq. (2.12) to obtain the vector of coefficients for Q
 - 2 Find all trajectories in $Index$ within r of Q using $Dist_{cby}$
 - 3 Retrieve from disk the corresponding (full) trajectories
 - 4 Compute the true distances using $Dist_{euc}$ and discard all the false positives
 - 5 **return** all trajectories within a distance r from Q with respect to $Dist_{euc}$
-

Algorithm 2.4.3: kNNSearch($Q, Index, k$) for CHEBY [9]

input : Q : a d -dimensional query trajectory;

$Index$: the index of Chebyshev coefficients;

r : a radius for range search.

output: the k most similar trajectories to Q with respect to $Dist_{euc}$

- 1 Apply Eq. (2.9) to Eq. (2.12) to obtain the vector of coefficients for Q
 - 2 Find the k -nearest neighbours to Q in $Index$ using $Dist_{cby}$
 - 3 Retrieve from disk the corresponding (full) trajectories
 - 4 Compute the true distances using $Dist_{euc}$ and record the maximum max
 - 5 Invoke the range search RangeSearch($Q, Index, max$) in Algorithm 2.4.2
 - 6 Retrieve from disk the corresponding (full) trajectories
 - 7 Compute the true distances using $Dist_{euc}$ and retain the nearest k trajectories
 - 8 **return** the k most similar trajectories to Q with respect to $Dist_{euc}$
-

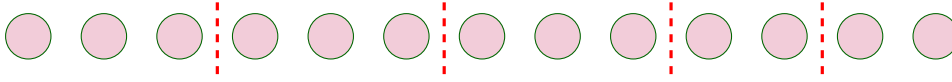


Figure 2.19: This figure is the first example of the *PAA* remainder assignment.

2.5 Analysis of Dimensionality Reduction Methods & R-tree Index Structure & k -NN Search Method

2.5.1 Analysis of *PAA* Dimensionality Reduction Method

PAA is a fast dimensionality reduction method. *PAA* is straight to understand and implement. *PAA* provides several advantages over competing schemes. We find some disadvantages during our implementation and evaluation.

- Advantages
 - Fixed segment length makes the algorithm process quickly.
 - Available for large scale datasets.
- Disadvantages
 - The mean value of the segment cannot hold all critical information.
 - Individual reconstruction error could be high.
 - Wave crest or wave trough would be diluted.
 - Fixed length of segment lacks flexibility.
 - n may not be divisible by N .

For the last disadvantage, we propose two methods in case n cannot be divisible by N . For a time series $C = \{(t_0, c_0), (t_1, c_1), \dots, (t_{n-1}, c_{n-1})\}$ ($t_0 < t_1 < \dots < t_{n-1}$), the remainder is $n \% N$.

1. First improvement: For the first $n \% N$ segments, every segment plus one.

Example 8. Like Figure 2.19, for time series C with length $n = 13$. Segment number $N = 5$. We can get $n \% N = 3$, $\lfloor \frac{n}{N} \rfloor = 2$. For the beginning three segments, the length of every segment is three. For the last two segments, the length of every segment is two. The improved *PAA* algorithm is shown in Algorithm 2.5.1.

Algorithm 2.5.1: Improved PAA 1

input : C : original time series
 n : time series length
 N : the number of segments;
output: \hat{C} : PAA of C ;

- 1 $l = \lfloor \frac{n}{N} \rfloor$;
- 2 $remainder = n \% N$;
- 3 first $remainder_{th}$ segments length = $l + 1$;
- 4 **for** $i = 0$; $i < N$; $i++$ **do**
- 5 $sum =$ every point in the i^{th} segment;
- 6 $\hat{c}_i = sum \div l$;
- 7 **return** $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$;

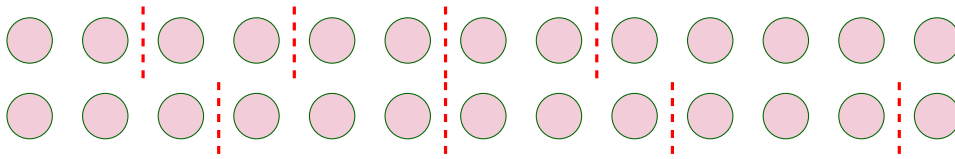


Figure 2.20: This figure is the second example of the PAA remainder assignment.

2. Second improvement: Merge remainder except for the last segment.

Example 9. Like Figure 2.20, for time series C with length 13. Segment number $N = 5$. We can get $n \% N = 3$, $\lfloor \frac{n}{N} \rfloor = 2$. For the beginning four segments, the length of every segment is 2. For the ending segment, the length is five. However, the ending segment is much bigger than other segments. So, except for the last segment, other segments length $l = 2 + 1 = 3$. The improved PAA algorithm is shown in Algorithm 2.5.2.

Algorithm 2.5.2: Improved PAA 2

input : C : original time series;
 n : time series length;
 N : the number of segments;
output: \hat{C} : PAA of C ;

- 1 $l = \lfloor \frac{n}{N} \rfloor$;
- 2 **if** $l + n \% N > N - 1$ **then**
- 3 The length of the first $N - 1$ segments is $l + 1$;
- 4 **for** $i = 0$; $i < N$; $i++$ **do**
- 5 $sum =$ every point in the i^{th} segment;
- 6 $\hat{c}_i = sum \div l$;
- 7 **return** $\hat{C} = \{\hat{c}_0, \dots, \hat{c}_{N-1}\}$;

2.5.2 Analysis of *APCA* Dimensionality Reduction Method

APCA represents the original time series by varying length segments. The number of coefficients cannot be confirmed during Haar wavelet transformation, so they keep the most N coefficients [13]. Because one Haar wavelet coefficient could produce one segment, two segments or three segments *APCA* representation, the reconstruction operation will produce an *APCA* representation with the segment number between N and $3N$. If N is too large or too small, N will be high, resulting in high query costs [13]. Thus it will perform worse than a sequential scan.

- Advantages
 - This dimensionality reduction can minimize individual reconstruction errors.
 - Save space cost.
- Disadvantages
 - Haar wavelet is not possible to get optimal compression level for all datasets.
 - If *APCA* meets the worst case, time series reconstructs to the second level, *APCA* will be the same as *PAA*.
 - The second level of the Haar wavelet makes use of the mean value of two adjacent data points.
 - Padding zero may influence the magnitude of the coefficient.

When implementing the *APCA* dimensionality reduction method, we met several problems which will influence the dimensionality reduction performance:

- When sorting coefficients according to normalized magnitudes, the normalized magnitudes could be equal.
- There is no merging choice is the best when the slightest rise in error is same.
- When computing minimum reconstruction error, whether we should get the whole minimum reconstruction error or get minimum reconstruction error for every merge step.
- For getting all merging cases, recursion is an effective way. However, for large data scales, recursion costs too much time.

2.5.3 Analysis of PLA Dimensionality Reduction Method

Compared with *PAA* and *APCA*, *PLA* uses the coefficients of the linear function to represent the original time series for the dimensionality reduction method.

- Advantages
 - The linear function consists a slant segment, and *PLA* should get a tight dimensionality reduction.
- Disadvantages
 - Computation of coefficients a and b needs to scan each point of the segment.
 - Authors miss a special case when segment length is very short for transferring $dist_{PLA}^2(q^{(i)}, x^{(i)})$ to line segments minimum distance in u-v space [15].
 - The equal-length segment may influence individual reconstruction error and cause space waste.

The condition of every case is listed in Table 2.5. For cases 1.3 and 3.3. We made improvements during implementation, which are $\{u_{A2} < 0, u_{B2} < u_{A2}, u_C \geq U_{A2}\}$ and $\{u_{A1} > 0, u_{A1} \leq u_{B1}, u_C \leq U_{A1}\}$. The reason is that for case 1.3, if the length of segment A_1A_2 is very short, like a point, u_{B2} will be on the left side of u_{A1} . This problem is shown in Figure 2.21 and Figure 2.22. We use Heron's formula to calculate the distance between one point and a segment for further improvement. The improved conditions of case 1 and case 3 are shown in Table 2.6.

2.5.4 Analysis of APLA Dimensionality Reduction Method

APLA represents the original time series by adaptive-length segments and linear function $a * t + b$.

- Advantages
 - *APLA* has guaranteed error bounds in the dimensionality reduction process.
 - *APLA* combines virtues of other dimensionality reduction methods, such as *APCA* and *PLA*.
 - *APLA* represents each segment by a linear function $a * t + b$, better than the constant value.

Cases	Switching Conditions	$mindist_{PLA}^2(q^{(i)}, e^{(i)})$
1.1	$u_{A2} < 0, u_{B1} \geq u_{A2}$	$ A_2B_1 ^2$
1.2	$u_{A2} < 0, u_{B1} \leq u_{A2}, u_{B2} \geq u_{A2}$	$ A_2C ^2$
1.3	$u_{A2} < 0, u_{A1} \leq u_{B2} \leq u_{A2}, u_C \geq u_{A2}$	$ A_2B_2 ^2$
1.4	$u_{A2} < 0, u_{B2} \leq u_{A2}, u_{A1} < u_C < u_{A2}$	$ B_2C ^2$
1.5	$u_{A2} < 0, u_{B2} < u_{A1}, u_C \leq u_{A1}$	$ A_1B_2 ^2$
2.1	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B1} > 0, u_C \geq u_{A2}$	$ A_2B_1 ^2$
2.2	$u_{A2} \leq 0, u_{A2} \geq 0, u_{B1} > 0, u_C < u_{A2}$	$ B_1C ^2$
2.3	$u_{A2} \leq 0, u_{A2} \geq 0, u_{B1} \leq 0, u_{B2} \geq 0$	0
2.4	$u_{A2} \leq 0, u_{A2} \geq 0, u_{B2} < 0, u_C > u_{A1}$	$ B_2C ^2$
2.5	$u_{A2} \leq 0, u_{A2} \geq 0, u_{B2} < 0, u_C \leq u_{A1}$	$ A_1B_2 ^2$
3.1	$u_{A1} > 0, u_{B1} > u_{A2}, u_C \geq u_{A2}$	$ A_2B_1 ^2$
3.2	$u_{A1} > 0, u_{B1} > u_{A1}, u_{A1} < u_C < u_{A2}$	$ B_1C ^2$
3.3	$u_{A1} > 0, u_{A1} \leq u_{B1} \leq u_{A2}, u_C \leq u_{A1}$	$ A_1B_1 ^2$
3.4	$u_{A1} > 0, u_{B2} > 0, u_{B1} \leq u_{A1} \leq u_{B2}$	$ A_1C ^2$
3.5	$u_{A1} > 0, u_{A1} \geq u_{B2}$	$ A_1B_2 ^2$

Table 2.5: Switching conditions for different cases

Cases	Switching Conditions	$mindist_{PLA}^2(q^{(i)}, e^{(i)})$
1.1	$u_{A2} < 0, u_{B2} < u_{A2}$	Compute distance between $B2$ and segment A_1A_2
1.2	$u_{A2} < 0, u_{B2} \geq u_{A2}$	Compute distance between $A2$ and segment B_1B_2
3.1	$u_{A1} > 0, u_{B1} > u_{A2}$	Compute distance between $B1$ and segment A_1A_2
3.2	$u_{A1} > 0, u_{B1} \leq u_{A1}$	Compute distance between $A1$ and segment B_1B_2

Table 2.6: Improvement by our implementation for case 1 and 3

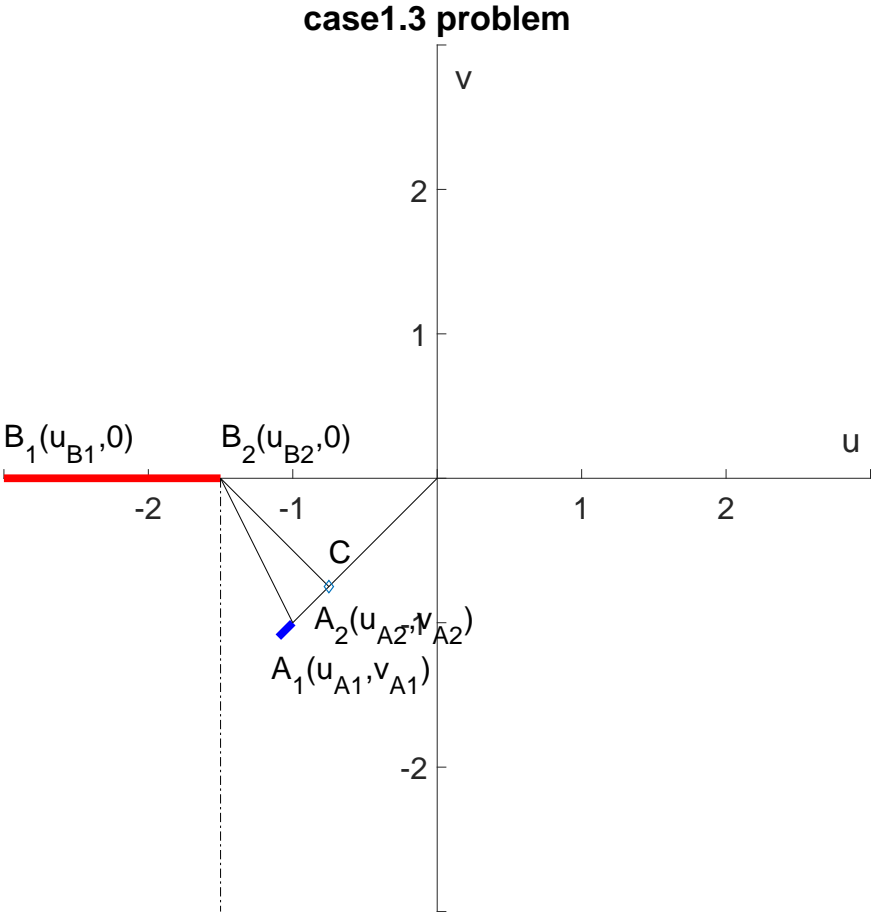


Figure 2.21: For case 1.3, when $|A_1A_2|$ is too short, the condition $u_{A1} \leq u_{B2} \leq u_{A2}$ is wrong.

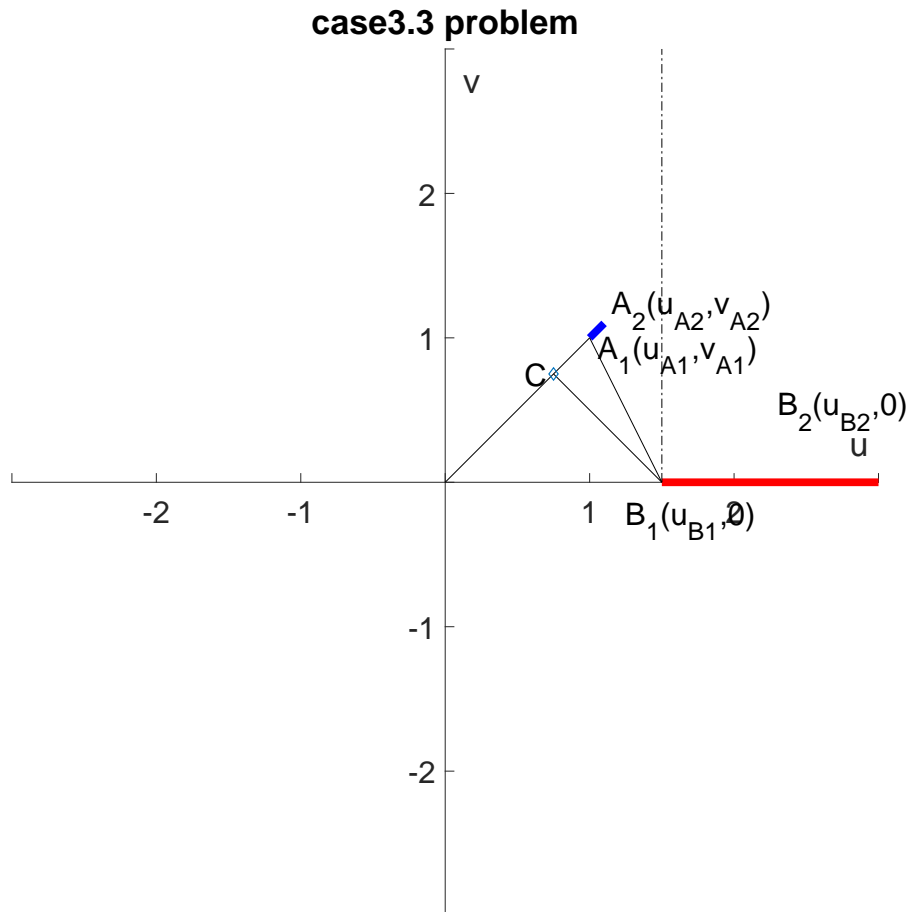


Figure 2.22: For case 3.3, when $|A_1A_2|$ is too short, the condition $u_{A1} \leq u_{B1} \leq u_{A2}$ is wrong.

- Given user defined dimensionality reduction coefficient number M , *APLA* segment number $N = \frac{1}{3} * M$. *APLA* could use few segment number to represent original time series.
- Disadvantages
 - *APLA* has $O(Nn^2)$ time complexity. n is the original time series length. N is the segment number.

When implementing *APLA* dimensionality reduction method, we improve some steps:

- We apply our proposed equations Eq. (3.2), (3.3), (3.18), (3.19), (3.20), (3.21), (3.22), (3.23) in lines 5 and line 13 of the Algorithm 2.2.4 to speed up coefficients a and b computation. Let l denote the segment length. The time complexity of coefficients a and b computation is from $O(l)$ to $O(1)$.

2.5.5 Analysis of *CHEBY* Dimensionality Reduction Method

- Advantages
 - Chebyshev polynomial is a continuous function. Thus the reconstruction effectiveness is good from a visual perspective, which is shown in Figure 2.8.
 - Multi-dimensional time series k -NN search was proposed.
- Disadvantages
 - The maximum deviation and sum deviation of *CHEBY* is not always better than *PAA*, *APCA* and *PLA*.
 - Computing procedure where Eq. (2.11) and Eq. (2.12) may cost much time if the number of coefficients is large.
 - The number of coefficients should be smaller than 25. Otherwise, this method will be involved in the dimensionality curse.
 - $Dist_{MBR}$ algorithm has not been provided so far [10], cannot get the distance between an internal node and query time series.
 - Reconstruction has to avoid weight function $w(t) = \frac{1}{\sqrt{1-t^2}}$.

Name	Lower Bounding Lemma	Segment Size
$Dist_{PAR}$	✓	Adaptive-Length
$Dist_{LB}$	✓	Adaptive-Length
$Dist_{AE}$		Adaptive-Length
$Dist_{PLA}$	✓	Equal-Length
$Dist_{CHEBY}$	✓	Equal-Length
$Dist_{PAA}$	✓	Equal-Length
$Dist_{SAX}$	✓	Equal-Length

Table 2.7: Lower Bounding Measures Comparison.

2.5.6 Analysis of SAX Dimensionality Reduction Method

- Advantages
 - As an equal-length segment dimensionality reduction method, *SAX* has a smaller dimensionality reduction time than adaptive-length segment dimensionality reduction methods.
 - *SAX* proposes a lower bounding measures for symbolic dimensionality reduction methods.
- Disadvantages
 - Some time series datasets do not have the Gaussian distribution.
 - The user defines the number of symbols.
 - The reconstruction of *SAX* is from symbol to number.

2.5.7 Analysis of Lower Bound Distance Measurements

Though $Dist_{AE}$ and $Dist_{LB}$ are good lower bound distance measurements for adaptive-length segment dimensionality reduction methods, $Dist_{AE}$ and $Dist_{LB}$ have limitations. $Dist_{AE}$ cannot keep lower bounding lemma, and $Dist_{LB}$ cannot get tight Euclidean distance approximation. We propose the lower bound distance measurement $Dist_{PAR}$ for adaptive-length segment dimensionality reduction methods with tight approximation and a lower bounding lemma. The comparison of equal-length segment lower bound distance measures and adaptive-length segment lower bound distance measures is shown in Table 2.7.

2.5.8 Analysis of R-tree Index Structure

- For *APCA* index calculation, *APCA* proposes a new minimum bounding rectangle (*MBR*) for index distance calculation. *APCA* tries to get the maximum value and minimum value for every segment. However, we should be aware when the maximum value is equal to the minimum value during our implementation. Volume calculation should miss this situation.
- For *PLA* and *CHEBY* implementation, we did not find how to insert dimensionality reduction coefficients into the R-tree index. For *CHEBY*, we do not find how to get index distance like *MBR*. When the maximum value is equal to the minimum value, we propose $|maximum\ value|$ as the factor of volume calculation.

2.5.9 Analysis of k -NN Search Method

(K-nearest-neighbor(k -NN)) Classification is a fundamental and simple non-parametric method. For query object Q , k -NN algorithm searches K (K is a positive integer) nearest neighbours. k -NN is instance-based learning and among the simplest of all machine learning algorithms. The k -NN algorithms for time series are shown in Algorithm 2.4.1 [40], which is applied to *PAA*, *APCA* and *PLA*. The second is shown in Algorithm 2.4.3, which is applied to *CHEBY*.

During our implementation and evaluation, we find Algorithm 2.4.1 [40] could be improved to speed up the k -NN search. The original algorithm uses lists data structure to store temp Euclidean distance and result Euclidean distance. Thus, line 4 in Algorithm 2.4.1 has to scan all records in the temp list during the while loop. The time complexity is $O(temp.size)$. However, We do not need to scan each record if we use a priority queue to store the temp Euclidean distance. The time complexity could be reduced to $O(\log(temp.size))$.

2.6 Conclusion

In this chapter, we introduce the background of time series and several dimensionality reduction methods. For the whole sequence similarity matching, R-tree index and k -NN algorithms are implemented. A comprehensive revision of seven classic dimensionality reduction methods is given in this chapter. We evaluated these methods with real datasets. We concluded that no method has an absolute advantage and is always better than other methods.

Name	Frequency	Time	Single Dim	Xaxis	Yaxis	Equal Size
<i>APLA</i>		✓	✓	✓		
<i>PAA</i>		✓	✓	✓		✓
<i>PAALM</i>		✓	✓	✓		✓
<i>APCA</i>		✓	✓	✓		
<i>PLA</i>		✓	✓	✓		✓
<i>CHEBY</i>		✓	✓	✓		✓
<i>DWT</i>	✓		✓	✓		✓
<i>DFT</i>	✓		✓	✓		✓
<i>SAX</i>		✓	✓		✓	✓

Table 2.8: Comparison of Dimensionality Reduction Methods

Chapter 3

Dimensionality Reduction Method

SAPLA

3.1 Overview

Similarity search over time series is essential in many applications, but it may cause a “dimensionality curse” due to the high dimensionality of time series. Various dimensionality reduction methods have been developed. Some of them sacrifice max deviation to get faster dimensionality reduction. One method, Adaptive Piecewise-Linear Approximation (*APLA*), uses guaranteed upper bounds for the best max deviation with slow dimensionality reduction time. We propose an adaptive-length dimensionality reduction method called Self Adaptive Piecewise-Linear Approximation (*SAPLA*). It consists of 1) initialization, 2) split & merge iteration, 3) segment endpoint movement iteration. Increment area, reconstruction area and several equations are applied to prune redundant computations. Initialization scans time series once to get initial dimensionality reduction. An increment area threshold is proposed to get segment endpoints. Split & merge iteration focus on sum upper bound reduction and adjusting segment number to user-defined number N . Upper bound is proposed as threshold for candidate split segment and iteration. Reconstruction area threshold is proposed to find candidate adjacent segments to be merged. Segment endpoint movement iteration is proposed to adjust segment left and right endpoints for sum upper bound reduction. Several equations are introduced to speed up segment coefficients computation. In the next chapter, we propose a lower bound distance measure between two time series to guarantee no false dismissals and tightness for adaptive-length dimensionality reduction methods. When mapping time series into the proposed

DBCH-tree, we split node and pick branch by lower bounding distance, not waste area. we experimentally verify different dimensionality reduction technologies for time series data in terms of their max deviation, dimensionality reduction time.

3.1.1 Motivation

APLA combines the virtues of *APCA* [40] and *PLA* [15] for the smaller max deviation. Because *APLA* has guaranteed error bounds (scan each point to get max deviation) in the reduction process, *APLA* has $O(Nn^2)$ time complexity (n is the original time series length and N denote the segment number after the reducing process). Our experiment shows that *APLA* is much slower than other methods. *APLA* [48] tries to minimize the segment max deviation, not the minimum of the sum max deviation. For example, two adjacent segments are denoted as \hat{c}_i and \hat{c}_{i+1} . \hat{c}_i already has the minimum max deviation. It is possible that a movement of \hat{c}_i the right endpoint would reduce the sum max deviation.

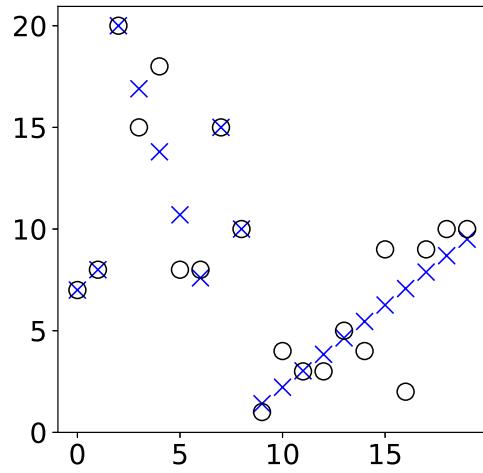
Fig. 3.1 illustrates a virtual comparison of *SAPLA* (Fig. 3.1a), *APLA* (Fig. 3.1b), *APCA* (Fig. 3.1c) and *PLA* (Fig. 3.1d). We compare the sum max deviation between the original and reconstructed time series. The length of original time series is 20. For a fair comparison, these methods have the same representation coefficients number $M = 12$ but not the same segment number (denoted as N). *SAPLA* and *APLA* use fewer segments to get better sum max deviations than *APCA* and *PLA*.

APCA [40] proposes two lower bounding distance measures that make adaptive-length dimensionality reduction methods indexable. One keeps lower bounding lemma, called $Dist_{LB}$, another has tight Euclidean distance approximation but non-lower bounding, called $Dist_{AE}$. We propose $Dist_{PAR}$, which has guaranteed lower bounding lemma and tightness.

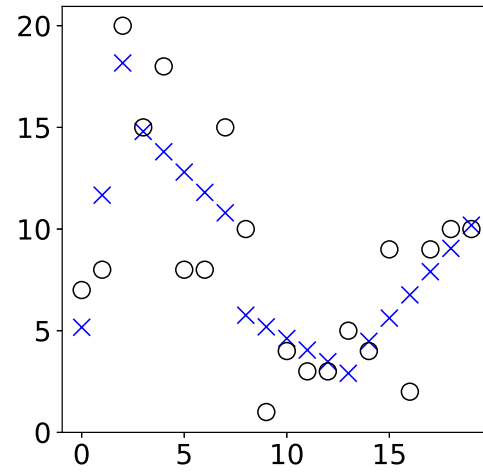
The rest of this chapter is organized as follows. Section 3.2 reviews related methods. Section 3.3 gives an overview of definitions. Section 3.4 introduces *SAPLA*. Section 4 introduces proposed lower bounding distance measure and improved node split algorithms. Section 3.5 demonstrates experiment results.

3.2 Related Work

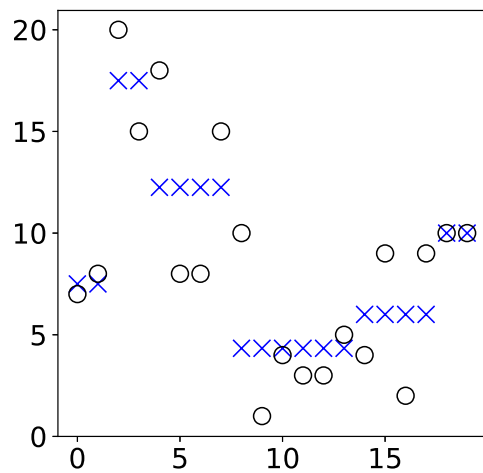
We have evaluated seven relevant methods, *PLA*, *APLA*, *APCA*, *PAA*, *CHEBY*, *PAALM*, *SAX* and our proposed *SAPLA*. This section provides a quick review of relevant methods. Let M denotes the



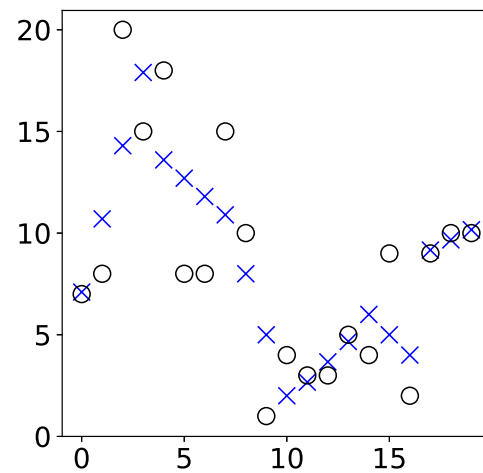
(a) Reconstruction of SAPLA. Max Deviation = 9.27273. $N = 4$



(b) Reconstruction of APLA. Max Deviation = 17.4667. $N = 4$



(c) Reconstruction of APCA. Max Deviation = 18.4167. $N = 6$



(d) Reconstruction of PLA. Max Deviation = 19.3999. $N = 6$

Figure 3.1: A visual comparison of time series dimensionality reduction methods. \circ is original time series. \times is reconstructed time series from representation. N is segment number.

representation coefficient number. Table 2.1 shows the summaries of eight dimensionality reduction methods.

3.3 Preliminaries

We introduced the definitions of time series (C), representation (\hat{C}), reconstructed time series (\check{C}), and max deviation (θ_i) in Section 2.1.

Definition 3.3.1. Segment Upper Bound (β_i). β_i is proposed to bound segment max deviation at different stages. β is the sum upper bound that $\beta = \sum_{i=0}^{N-1} \beta_i$. There are four stages to compute β_i in this thesis. 1) Computing β_i when *SAPLA* initializes C into \hat{C} . 2) Computing β_i from a merge operation. 3) Computing β_i from a split operation. 4) Computing β_i from segment endpoints movements.

3.4 Self Adaptive Piecewise Linear Approximation (*SAPLA*)

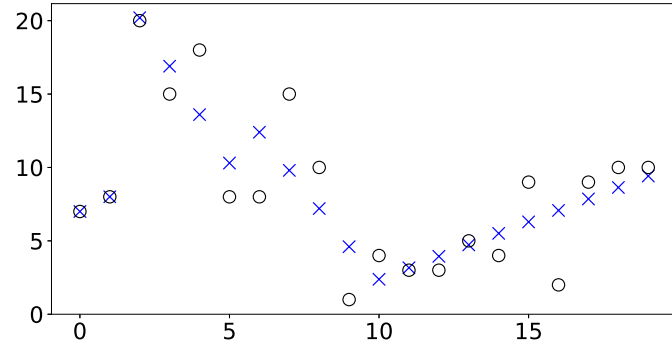
SAPLA focuses on finding segment endpoints to reduce the sum upper bound of segment max deviation. *SAPLA* consists of initialization; split & merge iteration; segment endpoint movement iteration. Fig. 3.2 shows an example of the *SAPLA* process.

Fig. 3.3 shows the framework of *SAPLA*. In the initialisation stage, users set a segment number N . *SAPLA* initializes time series C into \hat{C} . In the second stage, split & merge iteration reduces the sum upper bound by splitting a segment with the maximum upper bound into two segments and merging two adjacent segments with the minimum reconstruction area. Finally, the segment endpoint movement iteration helps to reduce the sum upper bound. Finally, we will get *SAPLA* representation $\hat{C} = \{\langle a_0, b_0, r_0 \rangle, \dots, \langle a_{N-1}, b_{N-1}, r_{N-1} \rangle\}$.

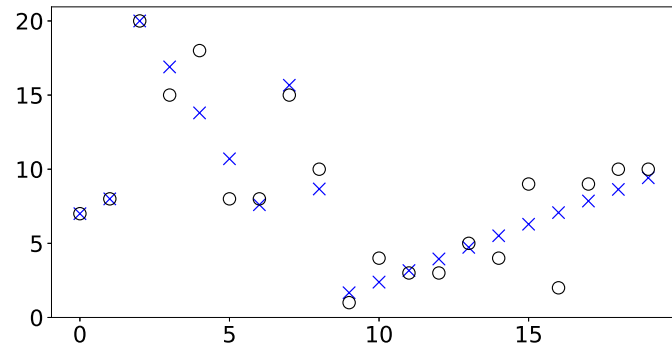
3.4.1 Proposed Equations and Area Computation

Increment Area

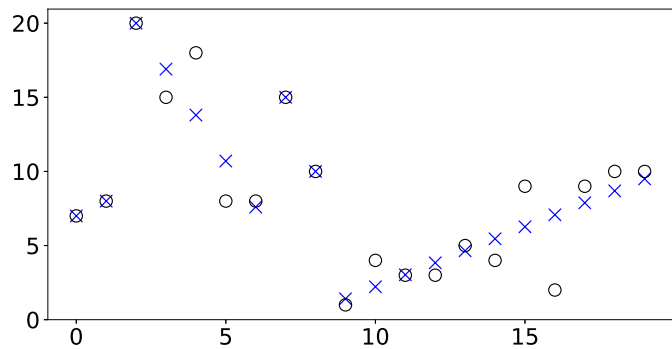
SAPLA represents the i^{th} segment by $\hat{c}_i = \langle a_i, b_i, r_i \rangle$. Let r_i denote the right endpoint position of this segment. Let l_i denote the segment length. Let C_i denote the original time series in this segment. Let \check{C}_i denote the reconstructed time series in this segment. Let r'_i denote the right point position followed by \hat{c}_i that $r'_i = r_i + 1$. Thus, the following original point value is $c_{r'_i}$ and $l'_i = l_i + 1$. Suppose there is a new segment called Increment Segment whose original time series is $C'_i = \{C_i, c_{r'_i}\}$. Its



(a) After initialization, max deviation = 14.6727.



(b) After split & merge iteration, max deviation = 10.6061.



(c) After segment endpoint movement, max deviation = 9.27273.

Figure 3.2: An example of the SAPLA process. Grey circle \circ is the original time series point. $n = 20$, $N = 4$. The blue cross \times is reconstructed time series point from SAPLA.

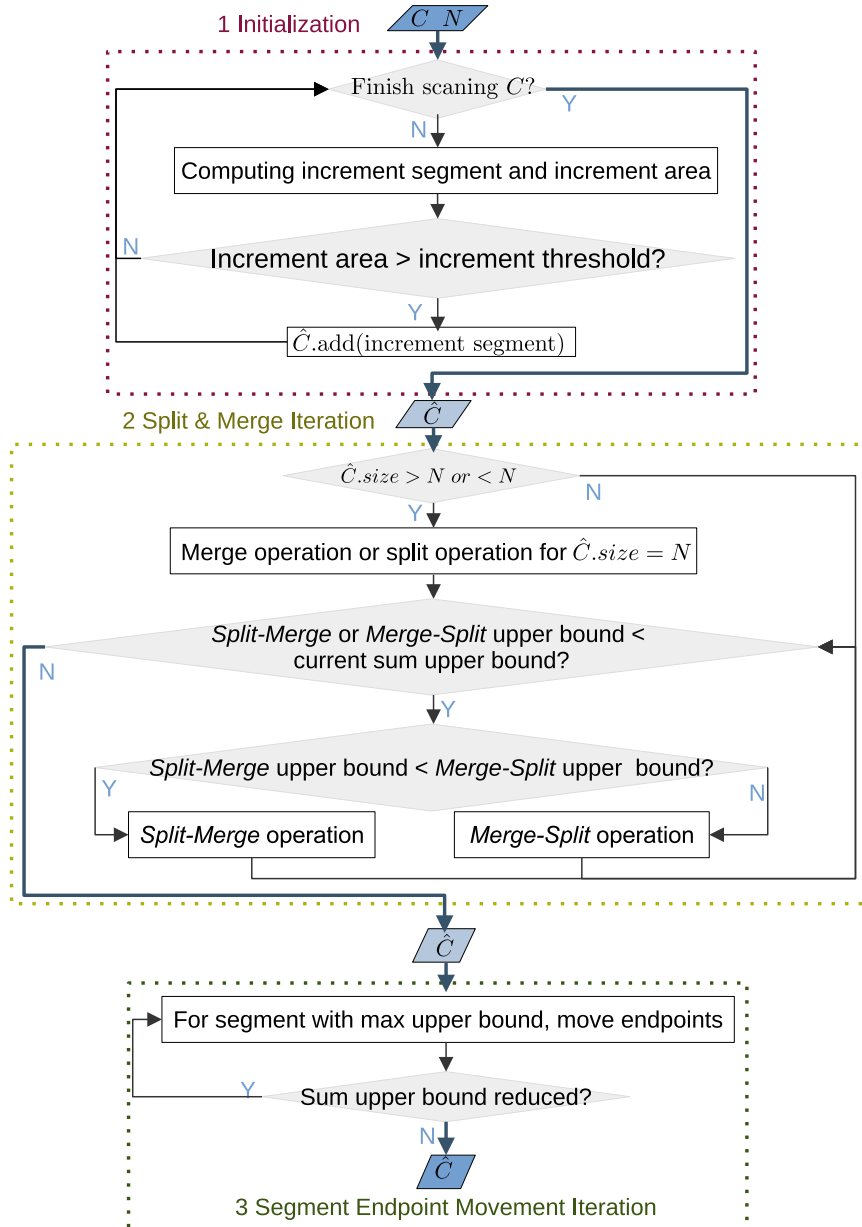


Figure 3.3: Framework of SAPLA: 1) Initializing C into \hat{C} . 2) Split & merge iteration reduces β of \hat{C} . 3) Segment endpoint movement iteration reduces β of \hat{C} .

SAPLA representation is $\hat{c}'_i = \langle a'_i, b'_i, r'_i \rangle$. The computation of a'_i, b'_i by Eq. (2.5) and (2.6) is $O(l'_i)$ time complexity. Thus we propose an extended equation of Eq. (2.5) and (2.6) in Eq. (3.2) and Eq. (3.3) that its time complexity is $O(1)$.

$$\begin{aligned} sum_i &= l_i * \left[\frac{(l_i - 1)}{2} * a_i + b_i \right] \\ &= \frac{l_i(l_i - 1)}{2} * a_i + l_i * b_i \end{aligned} \quad (3.1)$$

$$\begin{aligned} a'_i &= \frac{12 \sum_{t=0}^{l_i} (t - \frac{l_i}{2}) c_{t+r_{i-1}+1}}{l_i(l_i + 1)(l_i + 2)} \\ &= \frac{12 \sum_{t=0}^{l_i-1} [(t - \frac{l_i}{2}) c_{t+r_{i-1}+1}]}{l_i(l_i + 1)(l_i + 2)} + \frac{12(l_i - \frac{l_i}{2}) * c_{r'_i}}{l_i(l_i + 1)(l_i + 2)} \\ &= \frac{l_i - 1}{l_i + 2} * \frac{12 \sum_{t=0}^{l_i-1} [(t - \frac{l_i-1}{2} - \frac{1}{2}) c_{t+r_{i-1}+1}]}{l_i(l_i - 1)(l_i + 1)} + \frac{12(l_i - \frac{l_i}{2}) * c_{r'_i}}{l_i(l_i + 1)(l_i + 2)} \\ &= \frac{l_i - 1}{l_i + 2} * \frac{12 \sum_{t=0}^{l_i-1} [(t - \frac{l_i-1}{2}) c_{t+r_{i-1}+1}]}{l_i(l_i - 1)(l_i + 1)} \\ &\quad - \frac{l_i - 1}{l_i + 2} * \frac{12 \sum_{t=0}^{l_i-1} [(\frac{1}{2}) c_{t+r_{i-1}+1}]}{l_i(l_i - 1)(l_i + 1)} + \frac{12(l_i - \frac{l_i}{2}) * c_{r'_i}}{l_i(l_i + 1)(l_i + 2)} \\ &= \frac{l_i - 1}{l_i + 2} * a_i + \frac{6(l_i * c_{r'_i} - sum_i)}{l_i(l_i + 1)(l_i + 2)} \\ &= \frac{(l_i - 2)(l_i - 1)a_i}{(l_i + 1)(l_i + 2)} + \frac{6(c_{r'_i} - b_i)}{(l_i + 1)(l_i + 2)} \end{aligned} \quad (3.2)$$

$$\begin{aligned} b'_i &= \frac{2 \sum_{t=0}^{l_i} [(2l_i + 1 - 3t) c_{t+r_{i-1}+1}]}{(l_i + 1)(l_i + 2)} \\ &= \frac{2 \sum_{t=0}^{l_i} [(2l_i - 1 - 3t + 2) c_{t+r_{i-1}+1}]}{(l_i + 1)(l_i + 2)} + \frac{2(1 - l_i) c_{r'_i}}{(l_i + 1)(l_i + 2)} \\ &= \frac{l_i}{l_i + 2} * \frac{2 \sum_{t=0}^{l_i-1} [(2l_i - 1 - 3t) c_{t+r_{i-1}+1}]}{l_i(l_i + 1)} \\ &\quad + \frac{4 \sum_{t=0}^{l_i-1} c_{t+r_{i-1}+1}}{l_i(l_i + 1)} + \frac{2(1 - l_i) c_{r'_i}}{(l_i + 1)(l_i + 2)} \\ &= \frac{l_i}{l_i + 2} * b_i + \frac{4sum_i + 2(1 - l_i) c_{r'_i}}{(l_i + 1)(l_i + 2)} \\ &\quad + \frac{2(l_i - 1)(a_i l_i - c_{r'_i}) + (l_i + 5) l_i b_i}{(l_i + 1)(l_i + 2)} \end{aligned} \quad (3.3)$$

Let \check{C}'_i denote the reconstructed segment time series from \hat{c}'_i . Thus, $\check{c}'_{r'_i} = a'_i * l_i + b'_i$ is the last point in \check{C}'_i , and $\check{c}_{r'_i} = a_i * l_i + b_i$ is the extended point from \check{C}_i . We could get $\check{C}_i^e = \{\check{C}_i, \check{c}_{r'_i}\}$, the extended segment of \check{C}_i , called Extended Segment. Fig. 3.4 presents an example of \check{C}_i^e and \check{C}'_i . We find \check{C}_i^e and \check{C}'_i always intersect. Thus, we can get Lemma 3.4.1.

Lemma 3.4.1. Increment segment \check{C}'_i and Extended segment \check{C}_i^e have one intersection point, denoted as τ .

Proof. For two segments \check{C}'_i and \check{C}_i^e in Figure 3.4. $\check{C}'_i[1] = b'_i$ is the left endpoint of \check{C}'_i . $\check{C}_i^e[1] = b_i$ is the left endpoint of \check{C}_i^e . Let d_1 denote their left endpoint difference. Eq. (3.4) shows how to compute d_1 .

$$\begin{aligned}
 d_1 &= |\check{C}'_i[1] - \check{C}_i^e[1]| \\
 &= b'_i - b_i \\
 &= a'_i l_i + b'_i - (a_i l_i + b_i) \\
 &= \frac{(l_i - 1)((a_i * l_i + b_i) - c_{r'_i})}{(l_i + 1)(l_i + 2)} \\
 &= \frac{(l_i - 1)(\check{c}_{r'_i} - c_{r'_i})}{(l_i + 1)(l_i + 2)}
 \end{aligned} \tag{3.4}$$

$\check{C}'_i[l'_i] = \check{c}'_{r'_i}$ is the right endpoint of \check{C}'_i . $\check{C}_i^e[l'_i] = \check{c}_{r'_i}$ is the right endpoint of \check{C}_i^e . Let d_4 denote their right endpoint difference. Eq. (3.5) shows how to compute d_4 .

$$\begin{aligned}
 d_4 &= |\check{C}'_i[l'_i] - \check{C}_i^e[l'_i]| \\
 &= \check{c}'_{r'_i} - \check{c}_{r'_i} \\
 &= a'_i l_i + b'_i - (a_i l_i + b_i) \\
 &= \frac{2(2l_i + 1)(c_{r'_i} - (a_i * l_i + b_i))}{(l_i + 1)(l_i + 2)} \\
 &= \frac{2(2l_i + 1)(c_{r'_i} - \check{c}_{r'_i})}{(l_i + 1)(l_i + 2)}
 \end{aligned} \tag{3.5}$$

We could get $d_1 * d_4 \leq 0$ in Eq. (3.6). So, \check{C}'_i and \check{C}_i^e have one intersection point τ unless $d_1 == d_4$.

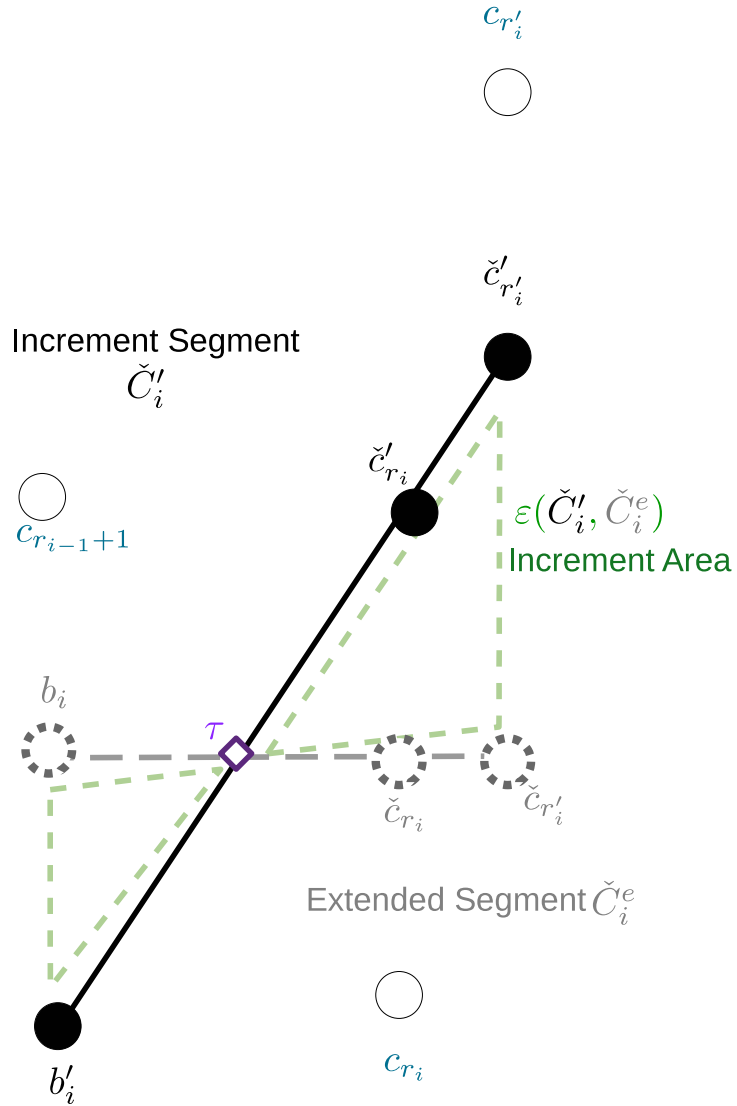


Figure 3.4: Example of Increment Area $\varepsilon(\check{C}'_i, \check{C}^e_i)$ simplified as two green triangles \triangle . \diamond is intersection point τ . The black circle \circ is original point c_t in C . The grey dashed dot \check{c}_t is reconstructed point \check{c}_t in \check{C}^e_i . The black dot \bullet is reconstructed point \check{c}'_t in \check{C}'_i . $t \in [0, n - 1]$.

$$\begin{aligned}
d_1 * d_4 &= \frac{(l_i - 1)(\check{c}_{r'_i} - c_{r'_i})}{(l_i + 1)(l_i + 2)} \\
&\quad * \frac{2(2l_i + 1)(c_{r'_i} - \check{c}_{r'_i})}{(l_i + 1)(l_i + 2)} \\
&\quad \geq 0 \\
&= \frac{-2(2l_i + 1)(l_i - 1)(c_{r'_i} - \check{c}_{r'_i})^2}{\underbrace{(l_i + 1)^2(l_i + 2)^2}_{>0}}
\end{aligned} \tag{3.6}$$

□

Because of Lemma 3.4.1, we can define the area between \check{C}'_i and \check{C}^e_i in Definition 3.4.1, called Increment Area. Let ε denote the summation of absolute difference between two time series, that is $\varepsilon(C, \check{C}) := \sum_{t=0}^{n-1} |c_t - \check{c}_t|$.

Definition 3.4.1. Increment Area ($\varepsilon(\check{C}'_i, \check{C}^e_i)$). $\varepsilon(\check{C}'_i, \check{C}^e_i)$ is an area between the Increment Segment \check{C}'_i and Extended Segment \check{C}^e_i . $\varepsilon(\check{C}'_i, \check{C}^e_i)$ can be simplified as an area of 2 triangles. Fig. 3.4 presents an example of $\varepsilon(\check{C}'_i, \check{C}^e_i)$, $\varepsilon(\check{C}'_i, \check{C}^e_i)$ is two green triangles.

β_i Segment Upper Bound in Initialization

Algorithm 3.4.1 introduces `get_max()` function for upper bound computation. Let $[]$ denote the order of points in one segment, such as $\check{c}_{r_{i-1}+1} = \check{C}_i[1]$, $\check{c}_{r_i} = \check{C}_i[l_i]$. Let max_d denote temp max value during increment process. We will get upper bound like $\beta_i = \max(\text{get_max}([1, l_i, l'_i], C'_i, \check{C}'_i, \check{C}^e_i), max_d) * l_i$. Theorem 3.4.2 proves we do not need to consider four point differences because they are always smaller than $|C'_i[l'_i] - \check{C}^e_i[l'_i]|$. Theorem 3.4.3 provides the conditions that make $\beta_i \geq \varepsilon_i$.

Algorithm 3.4.1: `get_max()` denoted as max_d_i

input : $v :=$ vector of id to compute;

$C_i, Q_i, T_i :=$ segment time series;

output: Maximum absolute difference

1 **Function** `get_max`(v, C_i, Q_i, T_i):

2 $m \leftarrow 0$;

3 **foreach** k in v **do**

4 $m \leftarrow \max(m, |C_i[k] - Q_i[k]|, |C_i[k] - T_i[k]|, |Q_i[k] - T_i[k]|)$;

5 **return** m ;

Theorem 3.4.2. *Because two segments \check{C}'_i and \check{C}^e_i have one intersection point (Lemma 3.4.1) in Figure 3.4. $\check{C}'_i[1] = b'_i$ is the left endpoint value of \check{C}'_i . $\check{C}^e_i[1] = b_i$ is the left endpoint value of \check{C}^e_i . Let d_1 denote their left endpoint difference. We can get $d_1 = |\check{C}'_i[1] - \check{C}^e_i[1]| = |b'_i - b_i|$. $\check{C}'_i[l_i] = \check{c}'_{r_i}$ is the left point of right endpoint in segment \check{C}'_i . $\check{C}^e_i[l_i] = \check{c}_{r_i}$ is the left point of right endpoint in segment \check{C}^e_i . We can get $d_2 = |\check{C}'_i[l_i] - \check{C}^e_i[l_i]| = |\check{c}'_{r_i} - \check{c}_{r_i}|$. $\check{C}'_i[l'_i] = \check{c}'_{r'_i}$ is the right endpoint of segment \check{C}'_i . $C'_i[l'_i] = c_{r'_i}$ is the right endpoint of original time series segment C'_i . We can get $d_3 = |\check{C}'_i[l'_i] - C'_i[l'_i]| = |c_{r'_i} - \check{c}'_{r'_i}|$. $\check{C}^e_i[l'_i] = \check{c}_{r'_i}$ is the right endpoint value of segment \check{C}^e_i . We can get $d_4 = |\check{C}'_i[l'_i] - \check{C}^e_i[l'_i]| = |\check{c}'_{r'_i} - \check{c}_{r'_i}|$. The right endpoints difference between original time series segment C'_i and extended segment \check{C}^e_i is denoted as $d_5 = |\check{C}^e_i[l'_i] - C'_i[l'_i]|$.*

So, we could get $d_4 \geq d_1$, $d_4 \geq d_2$ and $d_5 = d_3 + d_4$.

Proof. $l_i > 1$ in this thesis. As Fig. 3.4 shows, Eq. (3.7) shows $d_4 \geq d_1$. Eq. (3.8) shows $d_4 \geq d_2$. Eq. (3.9) shows $d_5 = d_3 + d_4$.

$$\begin{aligned}
d_4 \geq d_1 &\Rightarrow \\
|a'_i l_i + b'_i - (a_i l_i + b_i)| &\geq |b'_i - b_i| \Rightarrow \\
\frac{2(2l_i + 1)|c_{r'_i} - (a_i * l_i + b_i)|}{(l_i + 1)(l_i + 2)} &\geq \frac{(l_i - 1)|c_{r'_i} - (a_i * l_i + b_i)|}{(l_i + 1)(l_i + 2)} \Rightarrow \\
\frac{2(2l_i + 1)|c_{r'_i} - \check{c}'_{r'_i}|}{(l_i + 1)(l_i + 2)} &\geq \frac{(l_i - 1)|c_{r'_i} - \check{c}'_{r'_i}|}{(l_i + 1)(l_i + 2)} \Rightarrow \\
2(2l_i + 1) &> l_i - 1 \Rightarrow \\
l_i + 1 &> 0
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
d_4 \geq d_2 &\Rightarrow \\
|a'_i l_i + b'_i - (a_i l_i + b_i)| &\geq |a'_i(l_i - 1) + b'_i - (a_i(l_i - 1) + b_i)| \Rightarrow \\
\frac{2(2l_i + 1)|c_{r'_i} - (a_i * l_i + b_i)|}{(l_i + 1)(l_i + 2)} &\geq \frac{4(l_i - 1)|c_{r'_i} - (a_i * l_i + b_i)|}{(l_i + 1)(l_i + 2)} \Rightarrow \\
\frac{2(2l_i + 1)|c_{r'_i} - \check{c}'_{r'_i}|}{(l_i + 1)(l_i + 2)} &\geq \frac{4(l_i - 1)|c_{r'_i} - \check{c}'_{r'_i}|}{(l_i + 1)(l_i + 2)} \Rightarrow \\
2(2l_i + 1) &> 4(l_i - 1) \\
6 &> 0
\end{aligned} \tag{3.8}$$

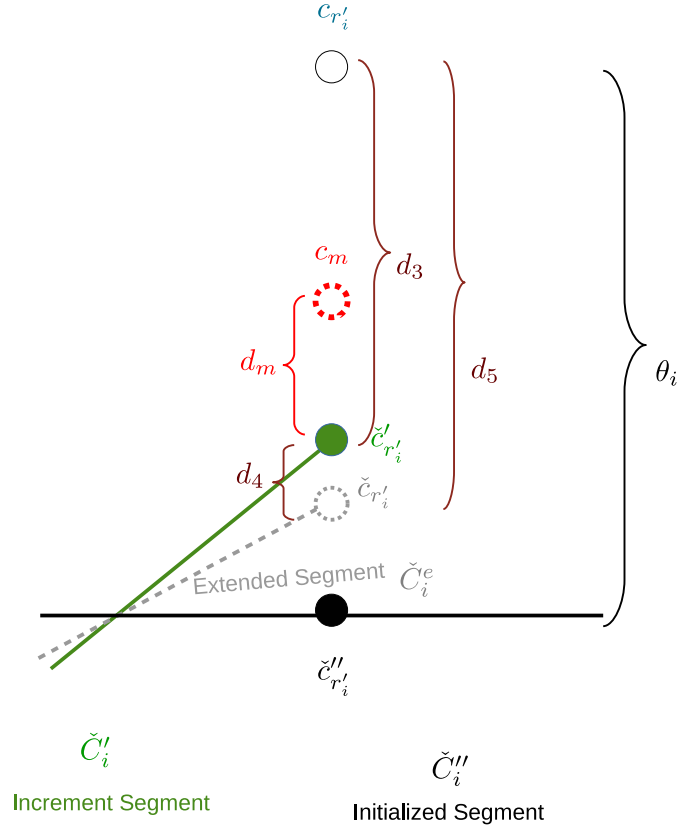


Figure 3.5: Example of proof $\beta_i \geq \theta_i$ in general cases, suppose $\theta_i = c_{r'_i} - \check{c}''_{r'_i}$. The black circle \circ is original point c_t in $C, t \in [0, n-1]$. The grey dashed dot \check{c} is reconstructed point \check{c}_t in \check{C}'_i . The green dot \bullet is reconstructed point \check{c}'_t in \check{C}'_i . The black dot \bullet is reconstructed point \check{c}''_t in \check{C}''_i . $c_m = \frac{c_{r'_i} + \check{c}''_{r'_i}}{2}$.

$$\begin{aligned}
 d_3 + d_4 &= \\
 \frac{l_i(l_i - 1) + 2(2l_i + 1)}{(l_i + 1)(l_i + 2)} |c_{r'_i} - \check{c}''_{r'_i}| &= \\
 \frac{l_i^2 + 3 * l_i + 1}{(l_i + 1)(l_i + 2)} |c_{r'_i} - \check{c}''_{r'_i}| &= \\
 \frac{(l_i + 1)(l_i + 2)}{(l_i + 1)(l_i + 2)} |c_{r'_i} - \check{c}''_{r'_i}| &= \\
 |c_{r'_i} - \check{c}''_{r'_i}| &= \\
 &= d_5
 \end{aligned} \tag{3.9}$$

□

Theorem 3.4.3 (In initialization part, upper bound is bigger than max deviation ($\beta_i \geq \theta_i$) in general

case). In figure 3.5, let c_m denote middle point value between $c_{r'_i}$ and $\check{c}''_{r'_i}$. We can get $c_m = \frac{c_{r'_i} + \check{c}''_{r'_i}}{2}$. Let d_3 denote the point difference between $c_{r'_i}$ and $\check{c}'_{r'_i}$. We can get $d_3 = c_{r'_i} - \check{c}'_{r'_i}$. Let d_4 denote the point difference between $\check{c}'_{r'_i}$ and $\check{c}_{r'_i}$. We can get $d_4 = \check{c}'_{r'_i} - \check{c}_{r'_i}$. When $d_5 = c_{r'_i} - \check{c}_{r'_i} \geq 0$, three factors support upper bound is bigger than max deviation : 1) $d_m = \check{c}'_{r'_i} - c_m = \check{c}'_{r'_i} - \frac{c_{r'_i} + \check{c}''_{r'_i}}{2} \leq 0$, which means $\check{c}'_{r'_i}$ is lower than the middle point of $c_{r'_i}$ and $\check{c}''_{r'_i}$; 2) Section 3.4.1 proves $l''_i = 3 \Rightarrow$ get $\max([1, 2, 3], C'_i, \check{C}'_i, \check{C}^e_i) = \theta_i$. When the length of segment is three, upper bound is equal to max deviation; 3) Section 3.4.1 shows $\frac{\beta_i}{l''_i - 1} \geq d_5$. In Eq. (3.10), $l''_i \in [3, n]$. When $d_5 < 0$, situation is similar.

Proof. As Fig. 3.5 shows, \check{C}''_i is an initialized segment. When $\theta_i = c_{r'_i} - \check{c}''_{r'_i}$, we will prove $\beta_i \geq \theta_i$ in general cases. Since $l_i \geq 2$, $l''_i - 3 \geq 0$. When $d_5 = c_{r'_i} - \check{c}_{r'_i} \geq 0$, $d_3 = c_{r'_i} - \check{c}'_{r'_i} \geq 0$ in Theorem 3.4.2, we can prove

$$\begin{aligned} \beta_i &\geq \underbrace{(l''_i - 1)d_3}_{\text{proof this part}} \geq \theta_i \Rightarrow \\ (l''_i - 2)d_3 + \check{c}''_{r'_i} - \check{c}'_{r'_i} &\geq 0 \Rightarrow \\ \underbrace{\frac{(l''_i - 3)d_3}{2}}_{\geq 0} &\geq \check{c}'_{r'_i} - \underbrace{\frac{c_{r'_i} + \check{c}''_{r'_i}}{2}}_{c_m} \Rightarrow \\ \frac{(l''_i - 3)d_5}{2} &\geq d_m \end{aligned} \tag{3.10}$$

We already know $d_5 > d_3$ in theorem 3.4.2. One special case $l''_i = 4$, $\frac{(l''_i - 3)d_5}{2} \geq d_m \Rightarrow d_5 \geq \frac{\theta_i}{2}$. If $\frac{\beta_i}{3} = d_5$ and $d_5 < \frac{\theta_i}{2}$, we will get $\beta_i < \theta_i$. Another special case is $d_m > 0$, so $\frac{(l''_i - 3)d_5}{2}$ may be smaller than d_m , especially when $\frac{\beta_i}{l''_i - 1} = d_5$. During our experiment, we have not found these two special cases that cause $\beta_i < \theta_i$. When $d_5 < 0$, the proof is similar to Eq. (3.10). We will not discuss this in details. \square

Reconstruction Area ($\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$)

There are two adjacent segments \hat{c}_i and \hat{c}_{i+1} . Let C'_{i+1} denote the part of time series that cover C_i and C_{i+1} that is $C'_{i+1} = \{C_i, C_{i+1}\}$. It is easy to know $r'_{i+1} = r_{i+1}$, $l'_{i+1} = l_i + l_{i+1} = r'_{i+1} - r_{i-1}$. We propose Eq. (3.11) and (3.12) that computation of a'_{i+1} and b'_{i+1} are $O(1)$. We use \check{C}'_{i+1} instead

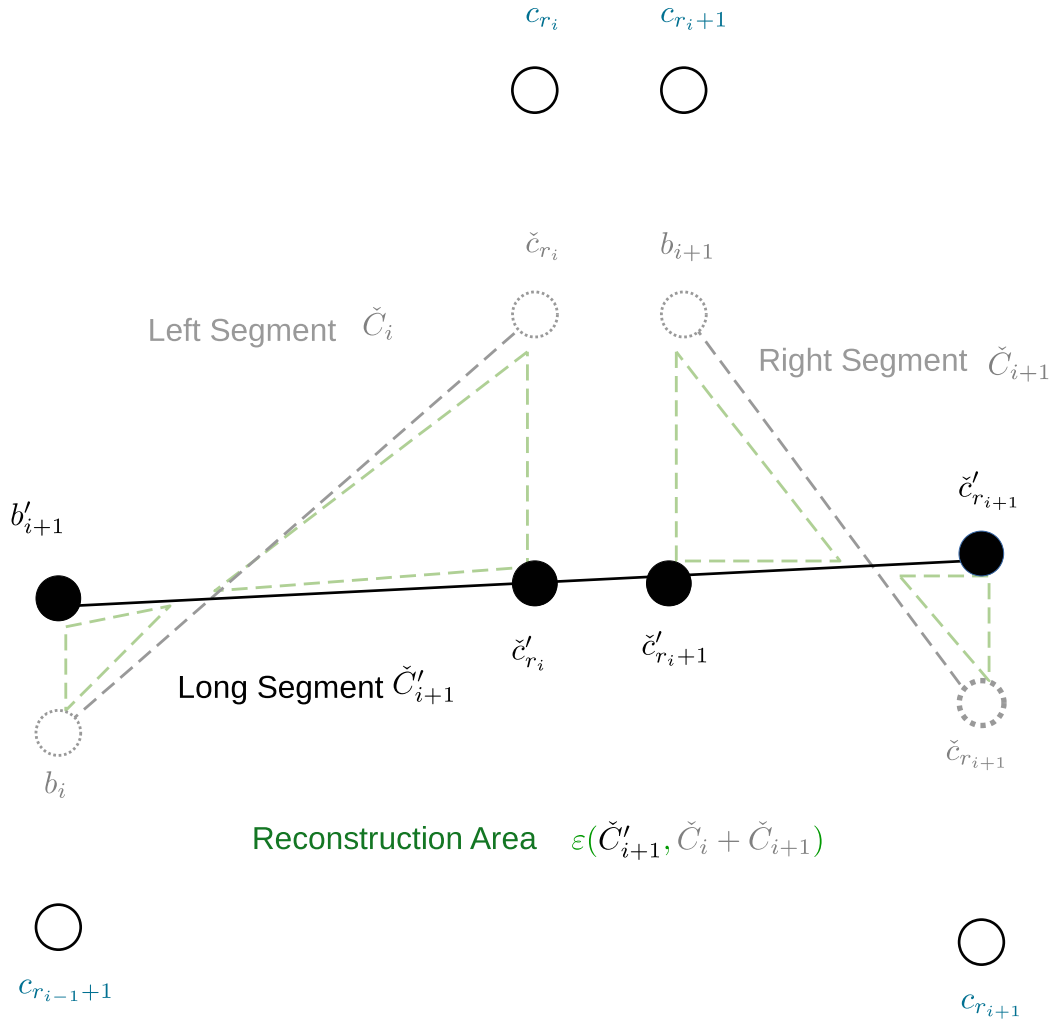


Figure 3.6: Example of Reconstruction Area $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ simplified as four green triangles \triangle . The black circle \circ is the original point c_t in C . The grey dashed dot \check{c}_t is the reconstructed point \check{c}_t in $\check{C}_i + \check{C}_{i+1}$. The black dot \bullet is the reconstructed point \check{c}'_t in \check{C}'_{i+1} .

of $\check{C}_i, \check{C}_{i+1}$ in \hat{C} is called merge operation. \check{C}'_{i+1} and $\check{C}_i + \check{C}_{i+1}$ will form a reconstruction area $(\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}))$ in Definition 3.4.4).

Definition 3.4.4. Reconstruction Area $(\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}))$ $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ is an area between $\check{C}_i + \check{C}_{i+1}$ and \check{C}'_{i+1} . $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ can be simplified as an area of several triangles or parallelograms. Fig. 3.6 provides an example of $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ as 4 green triangles in Fig. 3.6.

$$\begin{aligned}
a'_{i+1} &= \frac{l_i(l_i-1)(l_i+1)}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} a_i \\
&+ \frac{12}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} \left(\frac{l_i-1}{2} - \frac{l'_{i+1}-1}{2} \right) * sum_i \\
&\quad + \frac{l_{i+1}(l_{i+1}-1)(l_{i+1}+1)}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} a_{i+1} \\
&+ \frac{12}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} \left[l_i + \left(\frac{l_{i+1}-1}{2} - \frac{l'_{i+1}-1}{2} \right) \right] * sum_{i+1} \\
&= \frac{l_i(l_i-1)(l_i+1)}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} a_i - \frac{6l_{i+1}}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} * sum_i \\
&+ \frac{l_{i+1}(l_{i+1}-1)(l_{i+1}+1)}{l(l-1)(l+1)} a_{i+1} + \frac{6l_i}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} * sum_{i+1} \\
&= \frac{a_i l_i(l_i-1)(l_i+1) - 6l_{i+1} l_i}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} \\
&\quad + \frac{a_{i+1} l_{i+1}(l_{i+1}-1)(l_{i+1}+1) + 6l_i l_{i+1} b_i}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} \\
&= \frac{a_i * l_i(l_i-1)(l_i+1) - 6l_{i+1} l_i}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} + \frac{a_{i+1} * l_{i+1}(l_{i+1}-1)(l_{i+1}+1) + 6l_i l_{i+1} b_i}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} \\
&+ \frac{(b_{i+1} - b_i) * 6l_i l_{i+1}}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)} + \frac{a_{i+1} l_{i+1}(l_{i+1}-1)(l_{i+1}+1) + 6l_i l_{i+1} b_{i+1}}{l'_{i+1}(l'_{i+1}-1)(l'_{i+1}+1)}
\end{aligned} \tag{3.11}$$

$$\begin{aligned}
b'_{i+1} &= \frac{l_i(l_i+1)}{l'_{i+1}(l'_{i+1}+1)} b_i + \frac{2}{l'_{i+1}(l'_{i+1}+1)} [2l'_{i+1} - 1 - (2l_i - 1)] * sum_i \\
&+ \frac{l_{i+1}(l_{i+1}+1)}{l'_{i+1}(l'_{i+1}+1)} b_{i+1} + \frac{2}{l'_{i+1}(l'_{i+1}+1)} [2l'_{i+1} - 1 - (2l_{i+1} - 1) - 3l_i] * sum_{i+1} \\
&= \frac{l_i(l_i+1)}{l'_{i+1}(l'_{i+1}+1)} b_i + \frac{4l_{i+1}}{l'_{i+1}(l'_{i+1}+1)} * sum_i + \frac{l_{i+1}(l_{i+1}+1)}{l'_{i+1}(l'_{i+1}+1)} b_{i+1} \\
&\quad - \frac{2l_i}{l'_{i+1}(l'_{i+1}+1)} * sum_{i+1} \\
&= \frac{l_i(l_i+1) * b_i}{l'_{i+1}(l'_{i+1}+1)} + \frac{2l_{i+1} l_i(l_i-1) * a_i}{l'_{i+1}(l'_{i+1}+1)} + \frac{l_{i+1}(l_{i+1}+1) * b_{i+1}}{l'_{i+1}(l'_{i+1}+1)} \\
&\quad - \frac{l_i l_{i+1}(l_{i+1}-1) * a_{i+1}}{l'_{i+1}(l'_{i+1}+1)} + \frac{(4b_i - 2b_{i+1}) * l_i l_{i+1}}{l'_{i+1}(l'_{i+1}+1)} \\
&= \frac{b_i l_i(l_i+1) + 2a_i l_{i+1} l_i(l_i-1) + 4l_i l_{i+1} b_i}{l'_{i+1}(l'_{i+1}+1)} \\
&\quad + \frac{b_{i+1} l_{i+1}(l_{i+1}+1) - a_{i+1} l_i l_{i+1}(l_{i+1}-1) - 2l_i l_{i+1} b_{i+1}}{l'_{i+1}(l'_{i+1}+1)}
\end{aligned} \tag{3.12}$$

β_i Segment Upper Bound in Merge Operation

A long segment \check{C}'_{i+1} that is merged from $\check{C}_i, \check{C}_{i+1}$ by Eq. (3.11), (3.12). Let us denote $\check{C}_i + \check{C}_{i+1}$ as one reconstructed time series. Thus \check{C}'_{i+1} and $\check{C}_i + \check{C}_{i+1}$ have the same length. The upper bound in merging operation is defined as $\beta'_{i+1} = \text{get_max}([1, l_i, l_i + 1, l'_{i+1}], C'_{i+1}, \check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}) * (l'_{i+1} - 1)$. In other words, we choose the max absolute point differences from endpoints in $\{C_i, C_{i+1}, \check{C}_i, \check{C}_{i+1}, \check{C}'_{i+1}\}$. There is a virtual illustration of $\check{C}_i, \check{C}_{i+1}, \check{C}'_{i+1}$ in Fig. 3.6.

3.4.2 Initialization

Initialization algorithm transfers C into initialized \hat{C} ($\hat{C}.size \in [1, \frac{n}{2}]$). *SAPLA* scans C once to find the top N largest Increment Areas (refer to Definition 4.1) as \hat{C} segment endpoints. Let $\varepsilon(\check{C}'_j, \check{C}^e_j)$ denote an increment threshold. Let $\max(\varepsilon(\check{C}'_j, \check{C}^e_j))_{N-1}$ denote the $(N-1)^{th}$ largest Increment Area. In general cases, we could get at least N segments after initialization. When scanning a new point $c'_{i'}$, we will get increment area by Definition 3.4.1. We will compare this increment area with the increment threshold. If the current increment area is bigger than the increment threshold, we will get a new threshold and a new segment. After initialization, *SAPLA* uses split & merge iteration in Section 3.4.3 to get exact N segments representation.

Algorithm 3.4.2 shows the process of initialization aiming to find segment endpoints by $\varepsilon(\check{C}'_i, \check{C}^e_i)$. Computation of $\varepsilon(\check{C}'_i, \check{C}^e_i)$ needs a'_i, b'_i . Because of Eq. (3.2) and Eq. (3.3), Algorithm 3.4.2 scans C once to get a'_i, b'_i ($i \in [0, n)$). If $\varepsilon(\check{C}'_i, \check{C}^e_i) > \max(\varepsilon(\check{C}'_j, \check{C}^e_j))_{N-1}$ ($i > j$), \hat{c}'_i will be added to \hat{C} . $\langle (\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}), \hat{c}_{i+1} \rangle$ is stored in map ω^m from min to max. $\langle \beta_i, \hat{c}_i \rangle$ is stored in map ω^s from max to min.

3.4.3 Split & Merge Iteration

\hat{c}_i and \hat{c}_{i+1} with $\min_{i=0}^{\hat{C}.size-2} \varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ are regarded as candidate merge segments. After initialization, C is represented by initialized \hat{C} . Merge and split operations are applied to \hat{C} for sum upper bound (β) reduction. β_i is proposed to bound maximum deviation (ε_i in Definition 2.1.5) with $O(1)$ time complexity. $\max_{i=0}^{\hat{C}.size-1} \beta_i$ is regarded as a split threshold and β is regarded as an iteration threshold. Because a merge operation involves adjacent segments \hat{c}_i and \hat{c}_{i+1} , $\min_{i=0}^{\hat{C}.size-2} \varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ is proposed as a merge threshold.

Merge operations cannot guarantee a small max deviation. Therefore, we further apply split operations. The splitting operation could be regarded as a reverse operation of the merging operation

Algorithm 3.4.2: Initialization

input : $C : \{c_0, c_1, \dots, c_{n-1}\}; N;$
output: Initialized \hat{C} , $\hat{C}.size \in [1, \frac{n}{2}]$;
 ω^m : Map storing \hat{c}_{i+1} by $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ from min to max;
 $\omega^m.top := \langle \min_{i=0}^{\hat{C}.size-2} \varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}), \hat{c}_{i+1} \rangle$;
 ω^s : Map storing \hat{c}_i by β_i from max to min;
 $\omega^s.top := \langle \max_{j=0}^{\hat{C}.size-1} \beta_j, \hat{c}_j \rangle$;
 η : priority_queue. store $\varepsilon(\check{C}'_i, \check{C}_i^e)$; // $\eta.size \in [0, N)$;
 $\eta.top := \min \varepsilon(\check{C}'_i, \check{C}_i^e)$ in η ;
1 $\hat{c}_i := \langle a_i, b_i, r_i \rangle \leftarrow \langle c_1 - c_0, c_0, 1 \rangle$, $l_i \leftarrow 2$, $i \leftarrow 0$;
2 **while** $r_i < n$ **do**
3 Compute \hat{c}'_i and $\varepsilon(\check{C}'_i, \check{C}_i^e)$ from \hat{c}_i by Eq (3.2), Eq (3.3) and Definition 3.4.1;
4 Compute β_i by Section 3.4.1;
5 **if** $\eta.size < N - 1$ **then**
6 $\eta.push(\varepsilon(\check{C}'_i, \check{C}_i^e))$;
7 $\hat{C}.insert(\hat{c}_i)$;
8 $\omega^s.add\langle \beta_i, \hat{c}_i \rangle$;
9 **if** $i > 0$ **then**
10 Compute $\varepsilon(\check{C}'_i, \check{C}_{i-1} + \check{C}_i)$ by Eq. (3.11)(3.12) and Definition 3.4.4;
11 Compute β'_i by Section 3.4.1;
12 $\omega^m.add(\varepsilon(\check{C}'_i, \check{C}_{i-1} + \check{C}_i), \hat{c}_i)$;
13 $i++$, $r_i+ = 2$, $l_i \leftarrow 2$;
14 **else if** $\varepsilon(\check{C}'_i, \check{C}_i^e) > \eta.top$ **then**
15 Update η by $\varepsilon(\check{C}'_i, \check{C}_i^e)$;
16 $\hat{C}.insert(\hat{c}_i)$;
17 $\omega^s.add\langle \beta_i, \hat{c}_i \rangle$;
18 $\omega^m.add(\varepsilon(\check{C}'_i, \check{C}_{i-1} + \check{C}_i), \hat{c}_i)$;
19 $i++$, $r_i+ = 2$, $l_i \leftarrow 2$;
20 **else**
21 $\hat{c}_i \leftarrow \hat{c}'_i$; // $r'_i = r_i + 1$, $l'_i = l_i + 1$

in Section 3.4.1. In other words, we already have one long segment denoted as \check{C}'_{i+1} . We use two short segments $\check{C}_i + \check{C}_{i+1}$ to instead \check{C}'_{i+1} . It is easy to know $l'_{i+1} = l_i + l_{i+1}$ and $r'_{i+1} = r_{i+1}$. *SAPLA* proposes $\max_{i=0}^{\hat{C}.size-1} \beta_i$ as the segment split threshold. *SAPLA* applies the peak finding technique [17] to find the split point in \hat{c}_i with $\max_{i=0}^{\hat{C}.size-1} \beta_i$. After initialization, we have got β_i by Section 3.4.1. In split & merge iteration, we keep updating β_i in merge operation by Section 3.4.1 and in split operation by Section 3.4.3.

β_i Segment Upper Bound in Splitting Operation

The upper bound in the splitting operation could be regarded as the reverse operation of upper bound computation in merging operation. Thus, it is easy to get $\beta_i = \text{get_max}([1, l_i], C_i, \check{C}'_{i+1}, \check{C}_i) * (l_i - 1)$, $\beta_{i+1} = \text{get_max}([1, l_{i+1}], C_{i+1}, \check{C}'_{i+1}, \check{C}_{i+1}) * (l_{i+1} - 1)$. Note that, the order in segment for \check{C}'_{i+1} should be transformed as $[1 - l_i, \dots, l'_{i+1} - l_i]$; Thus, C_{i+1} , \check{C}_{i+1} and \check{C}'_{i+1} will have the same order value.

In merging operation, upper bound β'_{i+1} is defined in Section 3.4.1, and Fig. 3.6 shows an example of merging operation when \check{C}_i and \check{C}_{i+1} are merged into \check{C}'_{i+1} . In spitting operation, upper bound β_i is defined in Section 3.4.3 and Fig. 3.6 also be regarded as an example of a split operation where \check{C}'_{i+1} is split into \check{C}_i and \check{C}_{i+1} . Theorem 3.4.5 provides the conditions that make $\beta_i \geq \theta_i$.

Lemma 3.4.2. Let C_i denote the i^{th} segment of original time series C . Let \check{C}_i denote the i^{th} segment of reconstructed time series \check{C} from *SAPLA* representation \hat{C} . Let $s_p = \{\sum_{j=r_{i-1}+1}^{r_i} (c_j - \check{c}_j), \text{if } c_j - \check{c}_j > 0\}$ and $s_n = \{\sum_{j=r_{i-1}+1}^{r_i} (c_j - \check{c}_j), \text{if } c_j - \check{c}_j < 0\}$. $s_p + s_n = 0$, like Eq. (3.13) shows,

$$\begin{aligned}
& \sum_{j=r_{i-1}+1}^{r_i} (c_j - \check{c}_j) \\
&= \sum_{j=r_{i-1}+1}^{r_i} [c_j - (a_i * (j - r_{i-1} - 1) + b_i)] \\
&= \sum_{j=r_{i-1}+1}^{r_i} c_j - \left(\frac{l_i(l_i - 1)}{2} a_i + l_i b_i \right) \\
&= \sum_{j=r_{i-1}+1}^{r_i} c_j - \frac{6 \left[\sum_{t=1}^{l_i-1} (t * c_{r_{i-1}+1+t}) - \frac{l_i-1}{2} * \sum_{j=r_{i-1}+1}^{r_i} (c_j) \right]}{l_i + 1} \\
&\quad - \frac{2 \left[(2l_i - 1) \sum_{j=r_{i-1}+1}^{r_i} (c_j) - 3 \sum_{t=1}^{l_i-1} (t * c_{r_{i-1}+1+t}) \right]}{l_i + 1} \\
&= 0
\end{aligned} \tag{3.13}$$

Theorem 3.4.5. *In merging operation, suppose segment max deviation is $\theta'_{i+1} = |c_t - \check{c}_t|$, $t \in [r_{i-1} + 1, r'_{i+1}]$. Excludes max deviation, the sum value of absolute point difference is $\{s = \sum_{j=r_{i-1}+1}^{r'_{i+1}} |c_j - \check{c}_j|, j \neq t\}$. $\theta'_{i+1} \leq s$ because $\text{sum}_i = \sum_{j=r_{i-1}+1}^{r'_{i+1}} (c_j - \check{c}_j) = 0$ as Eq. (3.13) shows. Average point difference is $\frac{s}{l'_{i+1}-1}$. So, if $\text{get_max}([1, l_i, l_i + 1, l'_{i+1}], C'_{i+1}, \check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}) \geq \frac{s}{l'_{i+1}-1}$, we will get $\beta_{i+1} \geq \theta'_{i+1}$. In spitting operation, $\beta_i \geq \theta_i$ has same situation.*

Proof. $\beta_{i+1} = \text{get_max}([1, l_i, l_i + 1, l'_{i+1}], C'_{i+1}, \check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}) * (l'_{i+1} - 1)$. We suppose $\theta'_{i+1} = c_{r_i} - \check{c}'_{r_i}$ in Fig. 3.6. According to Lemma 3.4.2, θ'_{i+1} also has maximum value. That is when only c_{r_i} is located above \check{C}_i , other points are all below \check{C}_i and average value is $\frac{c_{r_i} - \check{c}'_{r_i}}{l'_{i+1} - 1}$. So, the worst case is $\text{max_d}'_{i+1} < \frac{c_{r_i} - \check{c}'_{r_i}}{l'_{i+1} - 1}$. During our experiment, we have not found this extreme case. For proof β_i in split operation is similar with β_{i+1} . We will not discuss in detail. \square

Finding Split Point in \check{C}'_{i+1}

SAPLA regards \hat{c}'_{i+1} with $\max_{i=-1}^{\hat{C}.size-2} \beta'_{i+1}$ as candidate split segment. Split point is regarded as r_i in \hat{c}_i . SAPLA finds r_i with nearly maximum $\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ by peak finding technique [17]. Eq. (3.14) and (3.15) help compute a_i and b_i in \hat{c}_i by a'_{i+1} , b'_{i+1} and a_{i+1} , b_{i+1} . Eq. (3.16) and (3.17) help compute a_{i+1} and b_{i+1} in \hat{c}_{i+1} in the same way. Fig. 3.7 provides an example of finding split point in \check{C}'_{i+1} .

$$\begin{aligned}
a_i &= a'_{i+1} * \frac{l'_{i+1} * (l'^2_{i+1} - 1)}{l_i * (l_i^2 - 1)} + \frac{6 * l_{i+1} * \text{sum}_i}{l_i(l_i^2 - 1)} \\
&\quad - \frac{6 * \text{sum}_{i+1}}{l_i^2 - 1} - \frac{l_{i+1} * (l_{i+1}^2 - 1)}{l_i * (l_i^2 - 1)} * a_{i+1} \\
&= a'_{i+1} \frac{l'_{i+1}(l'^2_{i+1} - 1)}{l_i(l_i^2 - 1)} + \frac{3l_{i+1}a_i}{l_i + 1} - \frac{3l_{i+1}(l_{i+1} - 1)a_{i+1}}{l_i^2 - 1} \\
&\quad + \frac{6l_{i+1}(b_i - b_{i+1})}{l_i^2 - 1} - \frac{l_{i+1}(l_{i+1}^2 - 1)}{l_i(l_i^2 - 1)} a_{i+1} \\
&= a'_{i+1} * \frac{l'_{i+1}(l'_{i+1} - 1)(l'_{i+1} + 1 + 3l_{i+1})}{l_i(l_i^2 - 1)} - \\
&\quad a_{i+1} * \frac{l_{i+1}(l_{i+1} - 1)(3l_i + 4l_{i+1} + 1)}{l_i(l_i^2 - 1)} \\
&\quad + \frac{6l_{i+1}[b'_{i+1}l'_{i+1} - b_{i+1}(l_i + l_{i+1})]}{l_i(l_i^2 - 1)}
\end{aligned} \tag{3.14}$$

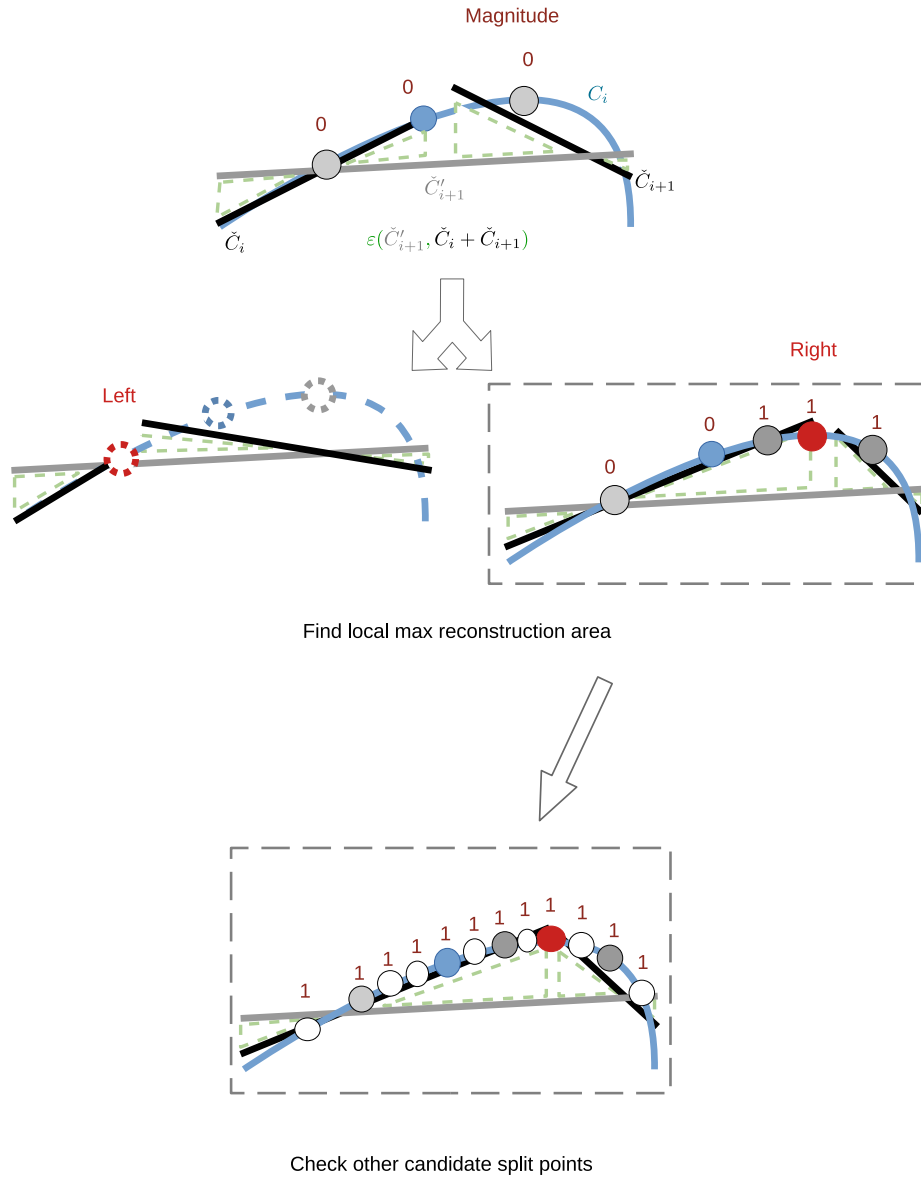


Figure 3.7: An example of finding a split point in \check{C}'_{i+1} . \bullet is the middle point of C_i . \bullet is the candidate split point in step 1), it is the middle point between endpoint and \bullet . 1) SAPLA gets split point \bullet with local max $\epsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ and max magnitude (1). 2) SAPLA will check other candidate points \circ until has bigger $\epsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1})$ or all magnitudes of candidate split points are equal to \bullet magnitude.

$$\begin{aligned}
b_i &= b'_{i+1} * \frac{l'_{i+1} * (l'_{i+1} + 1)}{l_i * (l_i + 1)} + \frac{2 * sum_{i+1}}{l_i + 1} \\
&\quad - \frac{4 * l_{i+1} * sum_i}{l_i(l_i + 1)} - \frac{l_{i+1} * (l_{i+1} + 1)}{l_i * (l_i + 1)} * b_{i+1} \\
&= b'_{i+1} \frac{l'_{i+1}(l'_{i+1} + 1)}{l_i(l_i + 1)} + \frac{l_{i+1}(l_{i+1} - 1)a_{i+1}}{l_i + 1} \\
&\quad - \frac{2l_i(l_i - 1)a_i}{l_i + 1} + \frac{2l_{i+1}(b_{i+1} - 2b_i)}{l_i + 1} - \frac{l_{i+1}(l_{i+1} + 1)}{l_i(l_i + 1)} b_{i+1} \\
&= b'_{i+1} * \frac{l'_{i+1}(l'_{i+1} + 1 - 4l_{i+1})}{l_i(l_i + 1)} + b_{i+1}l_{i+1} * \frac{2l'_{i+1} + l_{i+1} - 1}{l_i(l_i + 1)} \\
&\quad + a_{i+1} * \frac{(l'_{i+1} + l_{i+1})l_{i+1}(l_{i+1} - 1)}{l_i(l_i + 1)} - a'_{i+1} * \frac{2l_{i+1}l'_{i+1}(l'_{i+1} - 1)}{l_i(l_i + 1)}
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
a_{i+1} &= a'_{i+1} * \frac{l'_{i+1} * (l'^2_{i+1} - 1)}{l_{i+1} * (l_{i+1}^2 - 1)} + \frac{6 * sum_i}{l_{i+1}^2 - 1} \\
&\quad - \frac{6 * l_i * sum_{i+1}}{l_{i+1} * (l_{i+1}^2 - 1)} - \frac{l_i * (l_i^2 - 1)}{l_{i+1} * (l_{i+1}^2 - 1)} * a_i \\
&= a'_{i+1} \frac{l'_{i+1}(l'_{i+1} - 1)(l'_{i+1} + 1 - 3l_i)}{l_{i+1}(l_{i+1}^2 - 1)} + \\
&\quad a_i \frac{l_i(l_i - 1)(2l'_{i+1} + l_{i+1} - 1)}{l_{i+1}(l_{i+1}^2 - 1)} + \frac{6l_i l'_{i+1}(b_i - b'_{i+1})}{l_{i+1}(l_{i+1}^2 - 1)}
\end{aligned} \tag{3.16}$$

$$\begin{aligned}
b_{i+1} &= b'_{i+1} * \frac{l'_{i+1} * (l'_{i+1} + 1)}{l_{i+1} * (l_{i+1} + 1)} + \frac{2 * l_i * sum_{i+1}}{l_{i+1} * (l_{i+1} + 1)} \\
&\quad - \frac{4 * sum_i}{l_{i+1} + 1} - \frac{l_i * (l_i + 1)}{l_{i+1} * (l_{i+1} + 1)} * b_i \\
&= a'_{i+1} \frac{l_i l'_{i+1}(l'_{i+1} - 1)}{l_{i+1}(l_{i+1} + 1)} + b'_{i+1} \frac{l'_{i+1}(l'_{i+1} + 1 + 2l_i)}{l_{i+1}(l_{i+1} + 1)} \\
&\quad - \frac{a_i l_i(l_i - 1)(l'_{i+1} + l_{i+1})}{l_{i+1}(l_{i+1} + 1)} - \frac{b_i l_i(3l'_{i+1} + l_{i+1} + 1)}{l_{i+1}(l_{i+1} + 1)}
\end{aligned} \tag{3.17}$$

Algorithm 3.4.3 shows how split & merge iteration works. We have got $\omega^m.top$ and $\omega^s.top$ in Algorithm 3.4.2. When $\hat{C}.size > N$, SAPLA applies merge operation for $\omega^m.top$ to reduce $\hat{C}.size$. Eq. (3.11), (3.12) help to reduce computation time.

When $\hat{C}.size < N$, SAPLA applies split operation for $\omega^s.top$ to increase $\hat{C}.size$. Eq. (3.14), (3.15), (3.16), (3.17) help to reduce computation time.

When $\hat{C}.size = N$, SAPLA computes β_j, β_{j+1} from $\omega^s.top$ and β_{i+1} from $\min\{\omega^m.top, \varepsilon(\check{C}'_j, \check{C}'_{j-1} +$

$\check{c}_j)$, $\varepsilon(\check{c}'_{j+2}, \check{c}_{j+1} + \check{c}_{j+2})$. Thus, we will get β^{sm} from the above split-merge computation. Then SAPLA computes β_{y+1} from $\omega^m.top$ and β_t, β_{t+1} from $\omega^s.top$. We will get β^{ms} from the above merge-split computation. Finally, if $\beta \leq \min\{\beta^{ms}, \beta^{sm}\}$, iteration will be transferred to segment endpoint movement iteration. If $\beta > \min\{\beta^{ms}, \beta^{sm}\}$, $\{\beta, \hat{C}, \omega^s, \omega^m\}$ will be updated by the above computation and Algorithm 3.4.3 will continue iteration.

Algorithm 3.4.3: Split & Merge Iteration

input : $C : \{c_0, c_1, \dots, c_{n-1}\}$; $\hat{C}, \omega^m, \omega^s$ from Algorithm 3.4.2;
output: $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$ with reduced β ;

- 1 **while** $\hat{C}.size > N$ **do**
- 2 $\hat{c}'_{i+1} := \omega^m.top$; Update \hat{C} by \hat{c}'_{i+1} , ω^s by β_{i+1} , ω^m by $\varepsilon(\check{c}'_{i+1}, \check{c}_i + \check{c}_{i+1})$; //Eq. (3.11)
 (3.12), Section 3.4.1; Definition 3.4.4.
- 3 **while** $\hat{C}.size < N$ **do**
- 4 $\hat{c}'_{j+1} := \omega^s.top$; Update \hat{C} by \hat{c}_j, \hat{c}_{j+1} ; ω^s by β_j, β_{j+1} ; ω^m by
 $\varepsilon(\check{c}'_{j+1}, \check{c}_j + \check{c}_{j+1})$; //Section 3.4.3; Section 3.4.3.
- 5 $\beta^{sm} \leftarrow \beta^{ms} \leftarrow 0$; // all segment are labeled un-split and un-merged
- 6 **while** $\beta \geq \min\{\beta^{sm}, \beta^{ms}\}$ **do**
- 7 **if** $\omega^s.top$ has been split **then** get next segment.
- 8 **if** $\omega^m.top$ has been merged **then** get next segment.
- 9 Compute $\beta_j, \beta_{j+1}, \hat{c}_j, \hat{c}_{j+1}$ from $\omega^s.top$;
- 10 Compute $\beta_{i+1}, \hat{c}'_{i+1}$ from $\min\{\omega^m.top, \varepsilon(\check{c}'_j, \check{c}_{j-1} + \check{c}_j), \varepsilon(\check{c}'_{j+2}, \check{c}_{j+1} + \check{c}_{j+2})\}$; //
 $O(1)$.
- 11 Compute $\beta_{y+1}, \hat{c}'_{y+1}$ from $\omega^m.top$; // $O(1)$
- 12 Compute $\beta_t, \beta_{t+1}, \hat{c}_t, \hat{c}_{t+1}$ from $\omega^s.top$;
- 13 Compute β^{sm} by $\{\beta, \beta_j, \beta_{j+1}, \beta_{i+1}\}$;
- 14 Compute β^{ms} by $\{\beta, \beta_t, \beta_{t+1}, \beta_{y+1}\}$;
- 15 **if** $\beta^{sm} < \beta$ or $\beta^{ms} < \beta$ **then**
- 16 **if** $\beta^{sm} < \beta^{ms}$ **then** Update \hat{C} by $\hat{c}_j, \hat{c}_{j+1}, \hat{c}'_{i+1}$;
- 17 **else** Update \hat{C} by $\hat{c}_t, \hat{c}_{t+1}, \hat{c}'_{y+1}$;
- 18 //label split or merged
- 19 Update $\omega^s, \omega^m; \beta \leftarrow \min\{\beta^{sm}, \beta^{ms}\}$;

3.4.4 Segment Endpoint Movement Iteration

In the segment endpoint movement iteration, SAPLA moves left and right endpoints of \check{C}_i with $\max_{i=0}^{N-1} \beta_i$ for β reduction. Fig. 3.8 provides an example of \check{C}_i endpoints movement. There are 4 cases: 1) \check{C}_i increases right endpoints. 2) \check{C}_i decreases right endpoints. 3) \check{C}_i increases left endpoints. 4) \check{C}_i decreases left endpoints. SAPLA computes updated β of each movement and chooses minimal β as final movement.

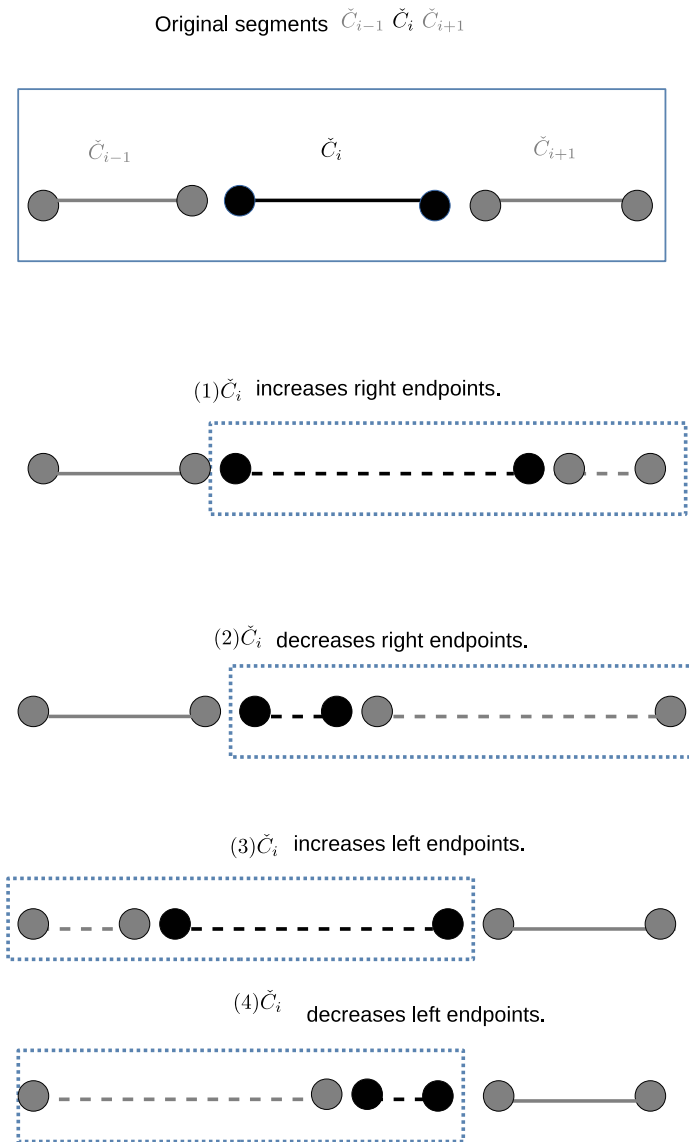


Figure 3.8: Example of segment endpoint movement for \check{C}_i . \check{C}_i tries to move endpoints for β reduction. There are four cases. The grey dot \bullet is the endpoint in $\check{C}_{i-1}, \check{C}_{i+1}$. The black dot \bullet is the endpoint in \check{C}_i .

β_i Segment Upper Bound in Endpoint Movement

One case is that \check{C}_i increases right endpoint, we could get β_i by Section 3.4.1. Because β_i computation for decreasing right endpoint, increasing left endpoint and decreasing left endpoint are similar with β_i computation in Section 3.4.1. We will not discuss the other 3 cases in detail. Eq. (3.2), Eq. (3.3), (3.18), (3.19), (3.20), (3.21), (3.22), (3.23) help to reduce computation time.

Algorithm 3.4.4 shows how segment endpoint movement iteration works. \check{C}_i with $\max_{i=0}^{N-1} \beta_i$ increases and decreases its left and right endpoints during iteration. Thus, we will get 4 updated c , which are called $\beta^a, \beta^b, \beta^c, \beta^d$ by β_i . Algorithm 3.4.5 shows how to compute $\hat{c}'_i, \hat{c}'_{i+1}, \beta_i, \beta_{i+1}$ and updated β .

1) β^a is from $\{\beta_i, \beta_{i+1}\}$ when \check{C}_i increases right endpoints and \check{C}_{i+1} decreases left endpoints. 2) β^b is from $\{\beta_i, \beta_{i+1}\}$ when \check{C}_i decreases right endpoints and \check{C}_{i+1} increases left endpoints. 3) β^c is from $\{\beta_{i-1}, \beta_i\}$ when \check{C}_i increases left endpoints and \check{C}_{i-1} decreases right endpoints. 4) β^d is from $\{\beta_{i-1}, \beta_i\}$ when \check{C}_i decreases left endpoints and \check{C}_{i-1} increases right endpoints. Each movement will continue until β cannot be reduced. SAPLA pops out computed \hat{c}_i and repeats above process for β reduction.

Algorithm 3.4.4: Segment Endpoint Movement Iteration

```

input :  $C$ ; After split & merge iteration  $\hat{C}$ ;
output:  $\hat{C} = \{\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{N-1}\}$  with reduced  $\beta$ ;
1  $\eta :=$  priority_queue;  $\eta.top := \hat{c}_i$  with  $\max_{i=0}^{N-1} \beta_i$ ;
2  $\beta^a \leftarrow \beta^b \leftarrow \beta^c \leftarrow \beta^d \leftarrow \beta$ ;
3 while  $\beta \geq \min\{\beta^a, \beta^b, \beta^c, \beta^d\}$  and  $\eta \neq \emptyset$  do
4    $\beta^a \leftarrow$  increase_right( $\beta, \hat{c}_i, \hat{c}_{i+1}$ ); //Algorithm 3.4.5
5    $\beta^b \leftarrow$  decrease_right( $\beta, \hat{c}_i, \hat{c}_{i+1}$ ); /Algorithm 3.4.6
6    $\beta^c \leftarrow$  decrease_right( $\beta, \hat{c}_{i-1}, \hat{c}_i$ );
7    $\beta^d \leftarrow$  increase_right( $\beta, \hat{c}_{i-1}, \hat{c}_i$ );
8   if  $\beta > \min\{\beta^a, \beta^b, \beta^c, \beta^d\}$  then
9     if  $\min(\beta^a, \beta^b) < \min(\beta^c, \beta^d)$  then
10       $\hat{c}_i \leftarrow \hat{c}'_i; \hat{c}_{i+1} \leftarrow \hat{c}'_{i+1}$ ;
11    else  $\hat{c}_{i-1} \leftarrow \hat{c}'_{i-1}; \hat{c}_i \leftarrow \hat{c}'_i$ ;
12     $\beta \leftarrow \min\{\beta^a, \beta^b, \beta^c, \beta^d\}$ ;  $\eta.pop$ ;

```

Algorithm 3.4.5: Coefficients for Endpoint Movement

Input: $\beta, \hat{c}_i, \hat{c}_{i+1}$
Output: Updated β

```

1  $\beta' \leftarrow \beta;$ 
2 Function increase_right( $\beta, \hat{c}_i, \hat{c}_{i+1}$ ):
3   while  $\beta' \leq \beta$  and  $l'_{i+1} \geq 2$  do
4      $\beta \leftarrow \beta';$ 
5      $r'_i \leftarrow r_i + 1;$ 
6     Computes  $\hat{c}'_i$  by Eq. (3.2),Eq. (3.3)
7     Computes  $\hat{c}'_{i+1}$  by Eq. (3.22) and Eq. (3.23);// $O(1)$ 
8     Computes  $\beta_i, \beta_{i+1}$  by Section 3.4.4;// $O(1)$ 
9     Updates  $\beta'$  by  $\beta_i, \beta_{i+1}$ 
10  return  $\beta$ 

```

Algorithm 3.4.6: Coefficients for Endpoint Movement

Input: $\beta, \hat{c}_i, \hat{c}_{i+1}$
Output: Updated β

```

1  $\beta' \leftarrow \beta;$ 
2 Function decrease_right( $\beta, \hat{c}_i, \hat{c}_{i+1}$ ):
3   while  $\beta' \leq \beta$  and  $l'_i \geq 2$  do
4      $\beta \leftarrow \beta';$ 
5      $r'_i \leftarrow r_i - 1;$ 
6     Computes  $\hat{c}'_i$  by Eq. (3.18), Eq. (3.19);
7     Computes  $\hat{c}'_{i+1}$  by Eq. (3.20) and Eq. (3.21) ;// $O(1)$ 
8     Computes  $\beta_i, \beta_{i+1}$  by Section 3.4.4;// $O(1)$ 
9     Updates  $\beta'$  by  $\beta_i, \beta_{i+1}$ 
10  return  $\beta$ 

```

$$\begin{aligned}
a'_i &= \frac{l_i + 1}{l'_i - 1} a_i + \frac{6 \text{sum}_i}{l_i l'_i (l'_i - 1)} + \frac{6 l_i c_{r_i}}{l_i l'_i (l'_i - 1)} \\
&= \frac{l_i + 1}{l'_i - 1} a_i + \frac{3(l_i - 1)}{l'_i (l'_i - 1)} a_i + \frac{6(b_i - c_{r_i})}{l'_i (l'_i - 1)} \\
&= \frac{l'_i (l_i + 1) a_i + 3(l_i - 1)}{l'_i (l'_i - 1)} a_i + \frac{6(b_i - c_{r_i})}{l'_i (l'_i - 1)} \\
&= \frac{(l_i + 4) a_i}{l_i - 2} + \frac{6(b_i - c_{r_i})}{(l_i - 1)(l_i - 2)}
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
b'_i &= \frac{b_i(l_i + 1)}{l'_i} \\
&\quad - \frac{4(\text{sum}_i - c_{r_i}) + 2(1 - l'_i)c_{r_i}}{l_i l'_i} \\
&= \frac{b_i(l_i + 1) - 4b_i}{l'_i} - 2a_i + \frac{2c_{r_i}}{l'_i} \\
&= \frac{(l_i - 3)b_i}{l_i - 1} - 2a_i + \frac{2c_{r_i}}{l_i - 1}
\end{aligned} \tag{3.19}$$

$$\begin{aligned}
a'_i &= \frac{a_i * (l_i - 1)}{l_i + 2} + \frac{6 * \text{sum}_i}{l'_i (l_i + 1)(l_i + 2)} \\
&= \frac{a_i * (l_i - 1)}{l_i + 2} + \frac{6 * (a_i * \frac{l_i - 1}{2} + b_i - c_{r_{i-1}})}{(l_i + 1)(l_i + 2)} \\
&= \frac{a_i(l_i - 1)(l_i + 4) + 6(b_i - c_{r_{i-1}})}{(l_i + 1)(l_i + 2)}
\end{aligned} \tag{3.20}$$

$$b'_i = \frac{2(2l_i + 1)c_{r_{i-1}} + l_i(l_i - 1)(b_i - a_i)}{(l_i + 1)(l_i + 2)} \tag{3.21}$$

$$a'_i = a_i + \frac{6(c_{r_{i-1}+1} - b_i)}{(l_i - 1)(l_i - 2)} \tag{3.22}$$

$$b'_i = a_i + \frac{(l_i + 3)b_i - 4c_{r_{i-1}+1}}{l_i - 1} \tag{3.23}$$

3.4.5 Time Complexity Analysis.

One advantage of *SAPLA* is that it can adjust iteration times by iteration threshold β . The worst time complexity is $O(N(n - 2N) + 2n \log n) = O(n(N + \log n))$. $O(n(N + \log n))$ is the worst case of the split & merge iteration. We think the initialization helps to avoid the worst case. Our experiment [1] shows *SAPLA* is much faster than *APLA*, *CHEBY*. *SAPLA* is faster than *APCA* for some datasets.

For the initialization part, the worst time complexity is $O(n \log N)$. The worst case is when \hat{c} is constructed ($l = 2$), *SAPLA* will apply $\hat{C}.\text{insert}(\hat{c})$ and update $\max(\varepsilon(\check{C}', \check{C}^\ell))_{N-1}$. $\max(\varepsilon(\check{C}', \check{C}^\ell))_{N-1}$ will be updated $\frac{n}{2}$ times and cost $O(\log N)$ during each update.

For the split & merge iteration part, when $\hat{C}.\text{size} > N$, the worst case is $\hat{C}.\text{size} = \frac{n}{2}$. After merge operation, sorts $\min_{i=0}^{\frac{n}{2}-2}(\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}))$, $\max_{j=0}^{\frac{n}{2}-1} \beta_j$ costs $O(2 \log \frac{n}{2})$ and $\hat{C}.\text{size}$ minus one. There are $\frac{n}{2} - N$ merge times. So, for merge loop, the worst time complexity is $O(\sum_{i=0}^{\frac{n}{2}-N} 2 \log(\frac{n}{2} - i)) = O(\sum_{i=N}^{\frac{n}{2}} \log i) \rightarrow O(n \log n)$.

When $\hat{C}.\text{size} < N$, the worst time complexity is $O(Nn)$. In split operation, the worst time complexity of finding split point is $O(n - 2\hat{C}.\text{size})$. After split operation, we need to sort β_i from big to small and reconstruction area (Definition 3.4.4) from small to big. Thus, the time complexity for sorting operation is $O(2 \log \hat{C}.\text{size})$. The worst case in split operation is when $\hat{C}.\text{size} = 1$, we need $N - 1$ loop times to get N segments. So, the time complexity is $O(n(N - 1) - N(N + 1) + 2 \sum_{i=1}^N \log i) = O(Nn)$.

When $\hat{C}.\text{size} = N$ and new $\beta < \text{old } \beta$, we need to sort $\max_{j=-1}^{N-1} \beta'_{j+1}$ and $\min_{i=0}^{N-1}(\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}))$, their time complexity is $O(2 \log N)$. So, the time complexity in each while loop is $O(n - 2N + 2 \log N)$. In each while loop, each segment has $\frac{1}{N}$ probability to be split except candidate merge segments and candidate split segment are same. For example, after split operation, $\varepsilon(\check{C}'_{j+1}, \check{C}_j + \check{C}_{j+1})$ is smaller than $\min_{i=0}^{N-1}(\varepsilon(\check{C}'_{i+1}, \check{C}_i + \check{C}_{i+1}))$, loop will stop. We do not apply split operation and merge operation on same segment in each while loop. When one segment is split, we will not split it again because we want to leave the split chance to other segments. We do the same strategy on merge operation. Thus, the while loop times are $\in [1, N]$, and the worst time complexity is $O(N(n - 2N) + 2N \log(N)) = O(Nn)$. The whole time complexity is $O(n(N + \log n))$.

For segment endpoint movements, the worst time complexity is $O(N(n - 2N)) = O(Nn)$. For the worst case, candidate \hat{c}_i with $\max_{i=0}^{N-1} \beta_i$ has $l_i = n - 2N$. Because each \check{C}_i has the longest $l_i = n - 2N$ movements, which means \check{C}_i is reduced to $l_i = 2$. Then \check{C}_{i+1} moves $n - 2N$ and so on. So, time complexity is $O(N(n - 2N))$. For segment endpoint movements, the worst case is each \check{C}_i

has the longest $l_i = n - 2N$ movements, thus time complexity is $O(N(n - 2N)) = O(Nn)$.

3.5 Experimental Evaluation

This section introduces the experimental evaluation using max deviation (θ) and dimensionality reduction time. We have implemented *SAPLA*, *APLA*, *APCA*, *PLA*, *PAA*, *CHEBY*, *PAALM*, *SAX*, R-tree, DBCH-tree by C++ [1]. Their summaries are shown in Table 2.1. The processor is Intel(R) Core(TM) i5-7600 CPU @ 3.50 GHz. RAM is 8 GB. We evaluated datasets by VS2019 in Windows 10 system. We randomly evaluated five query time series and summarized the results.

We evaluated all homogeneous datasets in [18] that the time series length is bigger than 1024 (20 datasets) in Table 3.1. The parameter is $\{12 \leq M \leq 24, M+ = 6\}$, $\{2 \leq K \leq 64, K* = 2\}$. Due to the space limitation, the comparisons of each parameter in all datasets are shown in our technical report [1].

3.5.1 Datasets in this thesis

In this section, we discuss the real datasets that are evaluated in our thesis. We use UCRArchive2018 [18] datasets, which were released in Fall 2018. UCRArchive2018 [18] claims that thousands of people have downloaded the UCR archive, and the UCR archive has been referenced above hundreds of times.

Each dataset has two parts, a TRAIN part and a TEST part. For example, the *HandOutlines* dataset has *HandOutlines_TEST.tsv* and *HandOutlines_TRAIN.tsv*. These files have different sizes. *TRAIN* file usually has more size than *TEST*. Because these files use *ASCII* format, we read datasets by C++ language. We use *ifstream* class to perform input / output operations.

Time series is a sequence taken at successive equally spaced points in time. Time series are regarded as a high dimensional data type. Each row has one time series. The first column is a class label, and we do not use this label in this thesis. The rest of the columns are time series values. We skip the first column when reading original time series.

In order to evaluate time series dimensionality reduction methods, we select datasets with the time series is longer than 1024. These datasets are shown in Table 3.1. The first column is the order of datasets in this thesis. The second column is the name of each dataset. The third column is an abbreviation of the dataset name. The fourth column is the type of each dataset.

Order	Name	Abbreviations	Type
1	<i>HandOutlines</i>	(<i>HO</i>)	Image
2	<i>HouseTwenty</i>	(<i>HT</i>)	Device
3	<i>PigAirwayPressure</i>	(<i>PAP</i>)	Hemodynamics
4	<i>PigArtPressure</i>	(<i>PAPS</i>)	Hemodynamics
5	<i>PigCVP</i>	(<i>CVP</i>)	Hemodynamics
6	<i>InlineSkate</i>	(<i>IS</i>)	Motion
7	<i>EthanolLevel</i>	(<i>EL</i>)	Spectro
8	<i>CinCECGTorso</i>	(<i>CT</i>)	Sensor
9	<i>SemgHandGenderCh2</i>	(<i>SHG</i>)	Spectrum
10	<i>SemgHandMovementCh2</i>	(<i>SHM</i>)	Spectrum
11	<i>SemgHandSubjectCh2</i>	(<i>SHS</i>)	Spectrum
12	<i>ACSF1</i>	(<i>ACS</i>)	Device
13	<i>EOGHorizontalSignal</i>	(<i>EHS</i>)	EOG
14	<i>EOGverticalSignal</i>	(<i>EVS</i>)	EOG
15	<i>Haptics</i>	(<i>H</i>)	Motion
16	<i>Mallat</i>	(<i>M</i>)	Simulated
17	<i>Phoneme</i>	(<i>PM</i>)	Sensor
18	<i>StarLightCurves</i>	(<i>SLC</i>)	Sensor
19	<i>MixedShapesRegularTrain</i>	(<i>MSR</i>)	Image
20	<i>MixedShapesSmallTrain</i>	(<i>MST</i>)	Image

Table 3.1: 20 Homogeneous Datasets [18]

Example of Single Dataset

The reason for evaluating up to twenty datasets is that the time series in each dataset has a big difference. Figure 3.9 shows the visual comparison of one original time series in the first ten homogeneous datasets. Figure 3.10 shows the visual comparison of one original time series in the last ten homogeneous datasets. Each sub-figure shows one original time series. There are one hundred time series in each dataset.

Before reducing original time series, we use z-score normalization to avoid outlier issues. An original time series $C = \{c_0, c_1, \dots, c_{n-1}\}$, where n is time series length. $c_t, t \in [0, n-1]$ is the time series point value. Let μ denote the average value of the original time series. Eq. 3.24 shows how to compute the average value of original time series C . Let σ denote standard deviation. Eq. 3.25 shows how to compute the standard deviation of original time series C . Eq. 3.26 shows how to compute the z-score value of original time series C .

$$\mu = \frac{\sum_{t=0}^{n-1} c_t}{n} \quad (3.24)$$

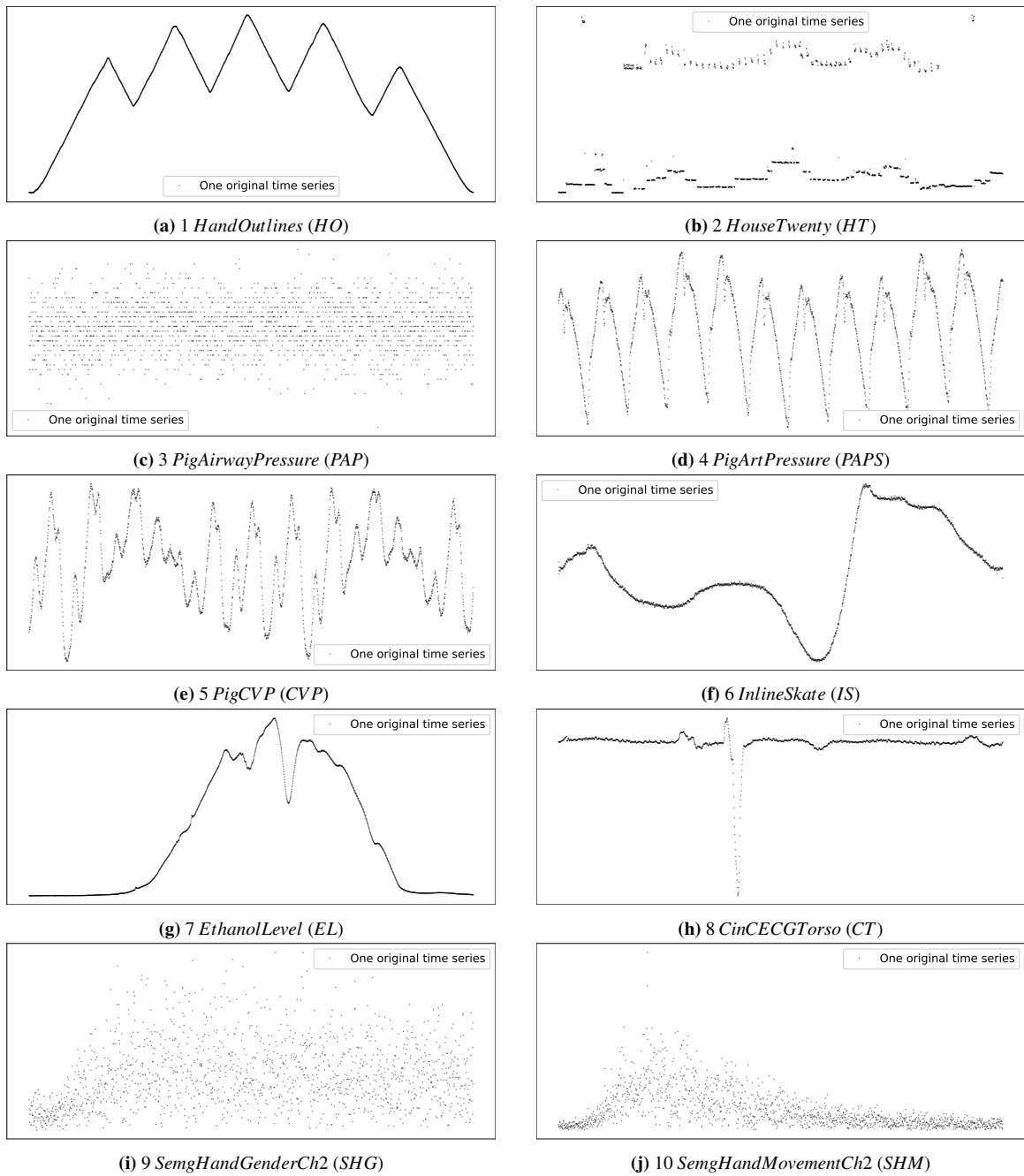


Figure 3.9: Visual comparison of the original time series in the first ten homogeneous datasets. Each sub-figure shows one original time series. The black dot is one point in original time series.

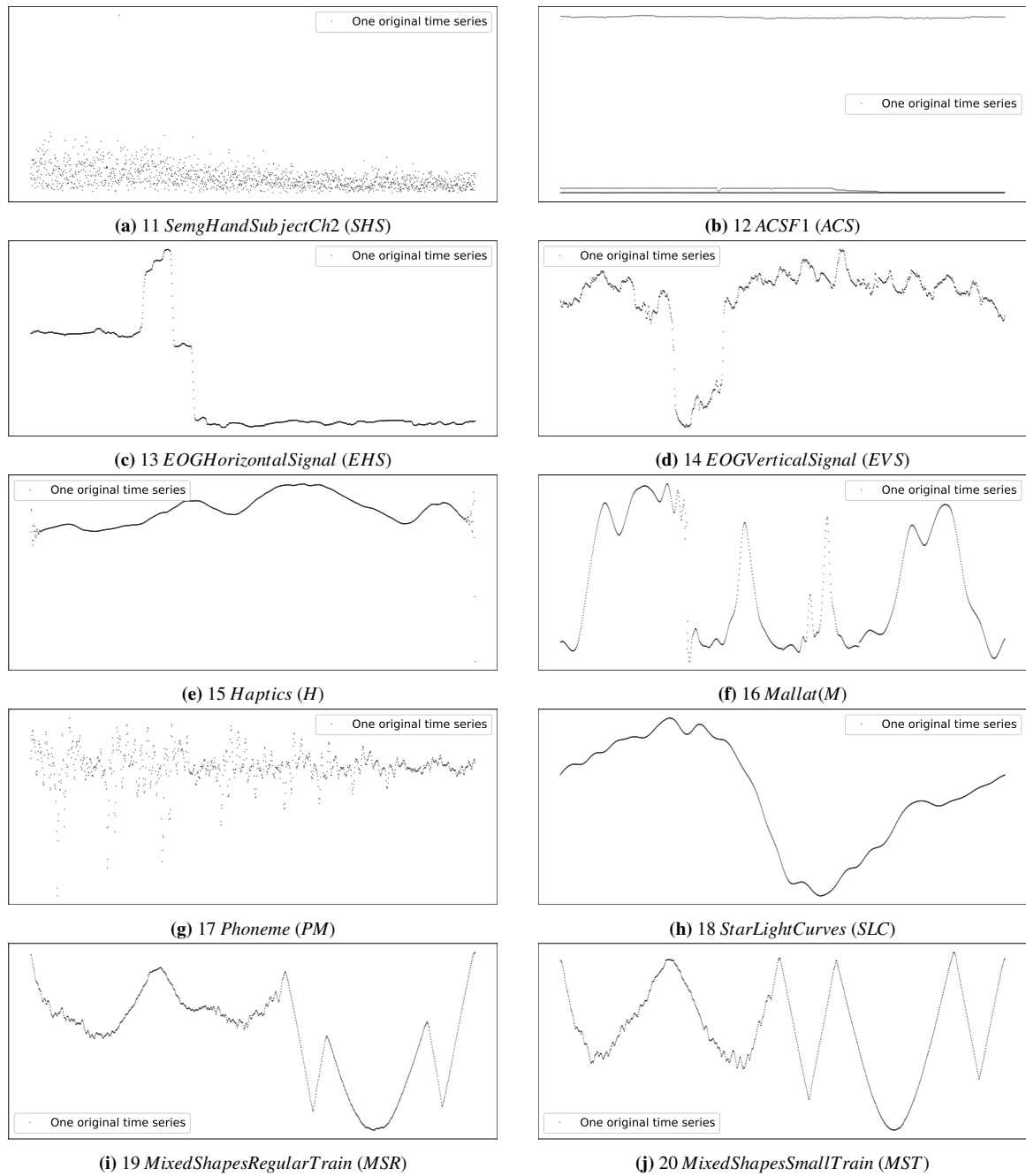


Figure 3.10: Visual comparison of the original time series in the last ten homogeneous datasets. Each sub-figure shows one original time series. The black dot is one point in original time series.

$$\sigma = \sqrt{\frac{\sum_{t=0}^{n-1} (c_t - \mu)^2}{n}} \quad (3.25)$$

$$c_t^z = \frac{c_t - \mu}{\sigma} \quad (3.26)$$

3.5.2 Comparison on Max Deviation (θ)

Max deviation is used to evaluate the tightness between the original time series and the reconstructed time series from representation coefficients. Table 3.2 shows the max deviation comparison of seven dimensionality reduction methods. The baseline dimensionality reduction method is *APLA*. *APLA* uses the dynamic programming to get optimal *APLA* representation. The max deviation of each dimensionality reduction method divides the *APLA* max deviation for each dataset. When the value in Table 3.2 is smaller than one, this dimensionality reduction method is better than *APLA*. Otherwise, it is worse than *APLA*. We could find *SAPLA* is better than *APLA* in five datasets, they are *HandOutlines*, *Mallat*, *Phoneme*, *MixedShapesRegularTrain* and *MixedShapesSmallTrain*. *APCA* is better than *APLA* in three datasets, they are *SemgHandGenderCh2*, *SemgHandSubjectCh2*, *ACSF1*. Because *PAALM* [61] does not consider max deviation, it has the biggest max deviation. However, the pruning power comparison in later sections will show the importance of small max deviation. The experiment result of *PAALM* indicates small max deviation is important in *k*-NN search. Table 3.3 shows the ranks comparison of each dimensionality reduction method in each dataset. The bottom is the average rank of each dimensionality reduction method. For example, the *SAPLA* ranks one in *HandOutlines* dataset, which means *SAPLA* has the best max deviation in this dataset. We could find *APLA* ranks one in eleven datasets. The average ranks of these dimensionality reduction methods are *APLA* (1.45), *SAPLA* (1.95), *PLA* (3.35), *APCA* (3.6), *CHEBY* (4.9), *PAALM* (6.95).

Figure 3.11 shows the scatter plot of our proposed dimensionality reduction method *SAPLA* and baseline dimensionality reduction method *APLA*. We could find that the most black points fall near the red colour diagonal line. There are four datasets *Phoneme*, *ACSF1*, *PigArtPressure* and *HouseTwenty*, which are far away from the red colour diagonal line. Figure 3.12 is a critical difference diagram for max deviation. The best ranks are to the left. The methods on the left side are better than the right side methods. Because we need to compare multiple dimensionality reduction methods in multiple datasets and compare our proposed dimensionality reduction method *SAPLA*

with other dimensionality reduction methods, the post-hoc analysis is Bonferroni-Dunn Tests. The predefined threshold value α is 0.05. The number of datasets is twenty. The average Rank is *SAPLA* : 1.95, *APLA* : 1.45, *APCA* : 3.60, *PLA* : 3.35, *PAA* : 5.80, *CHEBY* : 4.90, *PAALM* : 6.95. The critical difference value is 1.802097074. The null-hypothesis is that all dimensionality reduction methods perform equally well and the performance differences are random. If the difference of average ranks between two dimensionality reduction methods is smaller than the critical difference value, their performance difference is not significant (horizontal line). We could find three adaptive-length dimensionality reduction methods *SAPLA*, *APLA*, *APCA* and one equal-length dimensionality reduction method *PLA* are not significantly different. We use the Wilcoxon signed ranks test for comparison of paired dimensionality reduction methods between our proposed dimensionality reduction method *SAPLA* and other dimensionality reduction methods. The Wilcoxon signed ranks test is a non-parametric statistical hypothesis test. We do not use the t-test because the maximum deviation differences of paired dimensionality reduction methods are not normal distributions. The null-hypothesis is that the paired reduction methods perform equally well, and the performance differences are random. We define the predefined threshold value α as 0.05. The p-values of Wilcoxon signed ranks test are (*SAPLA* vs *APLA*: $p = 1.923370 \times 10^{-2} < 0.05$), (*SAPLA* vs *APCA*: $p = 1.923370 \times 10^{-2} < 0.05$), (*SAPLA* vs *PLA*: $p = 1.907349 \times 10^{-6} < 0.05$), (*SAPLA* vs *PAA*: $p = 1.907349 \times 10^{-6} < 0.05$), (*SAPLA* vs *CHEBY*: $p = 1.907349 \times 10^{-6} < 0.05$), (*SAPLA* vs *PAALM*: $p = 1.907349 \times 10^{-6} < 0.05$). we would reject the null hypothesis.

We can conclude that under the threshold 0.05, the adaptive-length segment dimensionality reduction method *APLA* has better max deviation than our proposed adaptive-length segment dimensionality reduction method *SAPLA*. *SAPLA* has better max deviation than one adaptive-length segment dimensionality reduction method *APCA* and one equal-length segment dimensionality reduction method *PLA*. The max deviation of our proposed adaptive-length segment dimensionality reduction method *SAPLA* is much better than three equal-length segment dimensionality reduction methods *CHEBY*, *PAA* and *PAALM*.

3.5.3 Comparison on Dimensionality Reduction Time

Equal-length segment dimensionality reduction methods are much faster than adaptive-length segment dimensionality reduction methods because they do not need to consider the length of each segment. However, analysis of other experiment parameters indicates adaptive-length segment dimensionality reduction methods could use fewer segments to get better max deviation and pruning

MaxDeviation

	SAPLA	APLA	APCA	PLA	PAA	Chebyshev	PAALM
HandOutlines	0.77	1.00	6.32	1.76	7.68	1.85	34.49
HouseTwenty	1.46	1.00	2.45	2.10	3.07	3.44	3.89
PigAirwayPressure	1.07	1.00	2.06	1.64	3.52	2.88	8.46
PigArtPressure	1.28	1.00	1.42	1.43	2.50	2.12	2.94
PigCVP	1.04	1.00	1.39	1.51	2.62	2.43	3.37
InlineSkate	1.12	1.00	3.12	1.48	4.02	1.51	11.34
EthanolLevel	1.01	1.00	3.32	1.45	3.76	1.56	23.70
CinCEGTorso	1.09	1.00	2.38	1.80	2.83	3.59	3.98
SemgHandGenderCh2	1.08	1.00	0.98	1.80	2.88	2.85	3.03
SemgHandMovementCh2	1.12	1.00	1.00	1.69	2.70	2.67	2.89
SemgHandSubjectCh2	1.15	1.00	0.95	2.00	3.16	3.13	3.28
ACSF1	1.25	1.00	0.58	2.02	3.68	3.79	3.69
EOGHorizontalSignal	1.25	1.00	2.23	1.44	2.68	2.65	8.27
EOGVerticalSignal	1.41	1.00	2.19	1.64	2.76	2.96	6.84
Haptics	1.10	1.00	1.81	1.40	2.63	1.46	5.65
Mallat	0.99	1.00	1.59	1.25	2.24	1.82	3.47
Phoneme	0.95	1.00	1.13	1.39	2.48	2.48	2.55
StarLightCurves	1.01	1.00	2.84	1.52	3.56	2.30	10.38
MixedShapesRegularTrain	0.87	1.00	1.78	1.22	2.64	1.53	4.75
MixedShapesSmallTrain	0.87	1.00	1.78	1.22	2.64	1.53	4.75

Table 3.2: Max deviation comparison on 20 homogeneous datasets. The baseline dimensionality reduction method is *APLA*.

Max Deviation Average Rank							
	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM
HandOutlines	1.0	2.0	5.0	3.0	6.0	4.0	7.0
HouseTwenty	2.0	1.0	4.0	3.0	5.0	6.0	7.0
PigAirwayPressure	2.0	1.0	4.0	3.0	6.0	5.0	7.0
PigArtPressure	2.0	1.0	3.0	4.0	6.0	5.0	7.0
PigCVP	2.0	1.0	3.0	4.0	6.0	5.0	7.0
InlineSkate	2.0	1.0	5.0	3.0	6.0	4.0	7.0
EthanolLevel	2.0	1.0	5.0	3.0	6.0	4.0	7.0
CinCECGTorso	2.0	1.0	4.0	3.0	5.0	6.0	7.0
SemgHandGenderCh2	3.0	2.0	1.0	4.0	6.0	5.0	7.0
SemgHandMovementCh2	3.0	2.0	1.0	4.0	6.0	5.0	7.0
SemgHandSubjectCh2	3.0	2.0	1.0	4.0	6.0	5.0	7.0
ACSF1	3.0	2.0	1.0	4.0	5.0	7.0	6.0
EOGHorizontalSignal	2.0	1.0	4.0	3.0	6.0	5.0	7.0
EOGVerticalSignal	2.0	1.0	4.0	3.0	5.0	6.0	7.0
Haptics	2.0	1.0	5.0	3.0	6.0	4.0	7.0
Mallat	1.0	2.0	4.0	3.0	6.0	5.0	7.0
Phoneme	1.0	2.0	3.0	4.0	6.0	5.0	7.0
StarLightCurves	2.0	1.0	5.0	3.0	6.0	4.0	7.0
MixedShapesRegularTrain	1.0	2.0	5.0	3.0	6.0	4.0	7.0
MixedShapesSmallTrain	1.0	2.0	5.0	3.0	6.0	4.0	7.0
Average Rank	1.95	1.45	3.6	3.35	5.8	4.9	6.95

Table 3.3: Max deviation comparison on 20 homogeneous datasets. Ranks of these dimensionality reduction methods in each dataset. The bottom line is the average rank of each method.

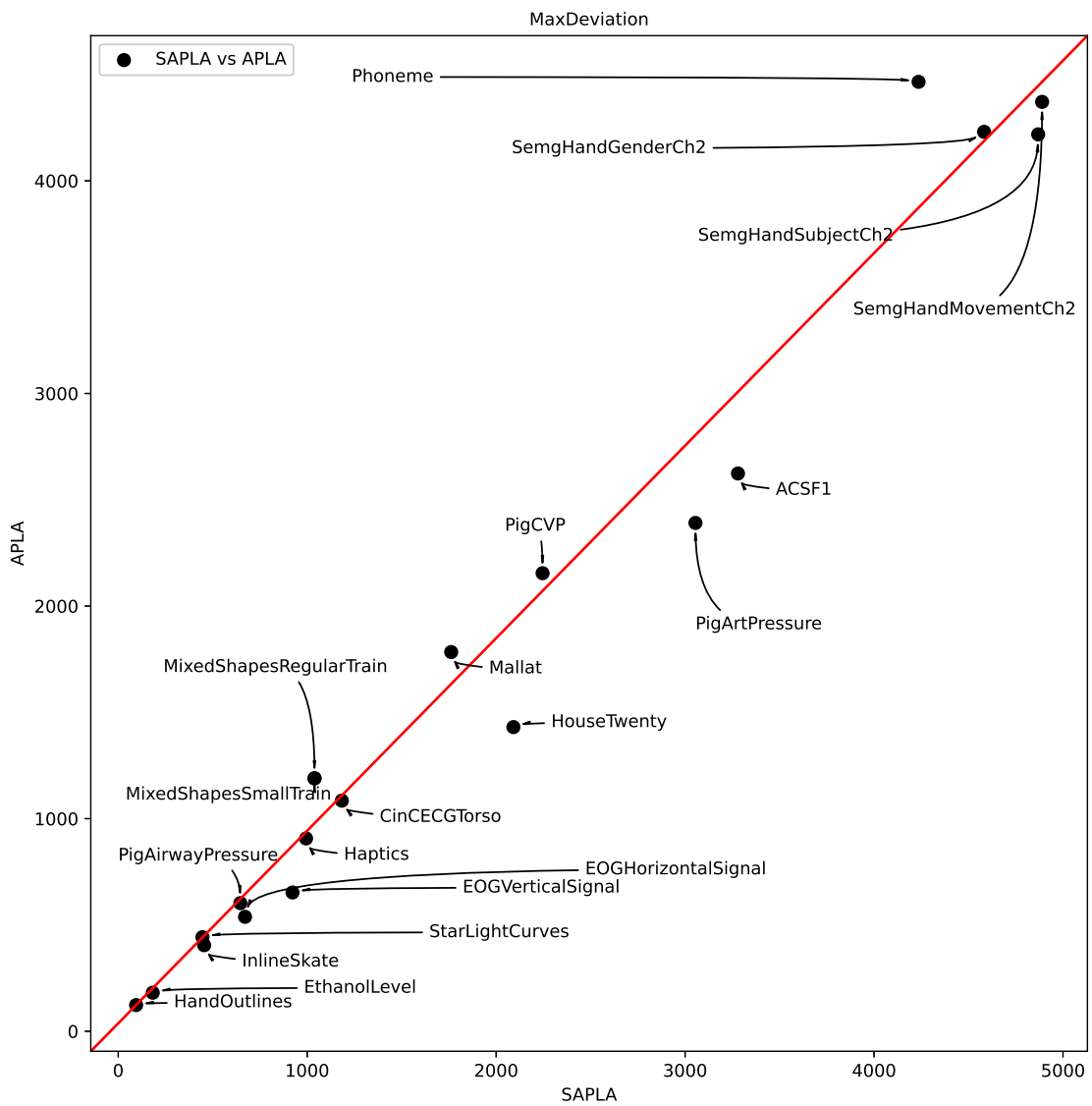


Figure 3.11: Max deviation comparison between *SAPLA* and *APLA* on 20 homogeneous datasets.

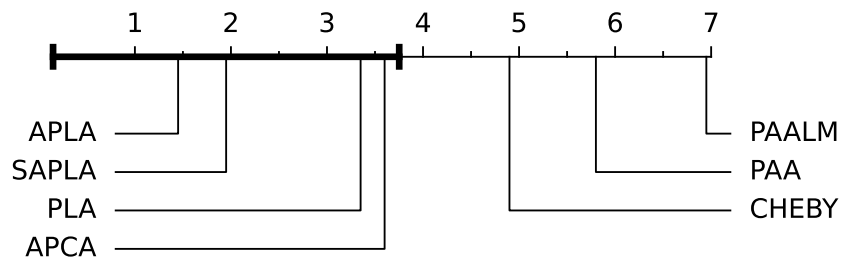


Figure 3.12: The critical difference diagram of max deviation comparison on 20 homogeneous datasets.

power than equal-length segment dimensionality reduction methods. Table 3.4 shows that *SAPLA* is faster than *APLA* up to n times (n is the length of the original time series). The baseline dimensionality reduction method is *APLA*. The dimensionality reduction time of each dimensionality reduction method is divided by the *APLA* dimensionality reduction time deviation for each dataset. When the value in Table 3.4 is smaller than one, it means this dimensionality reduction time is faster than *APLA*. Otherwise, it is slower than *APLA*. We could find *SAPLA* is better than *APLA* about one thousand times. Because the original time series length is 1024 in our experiment. We can conclude that *SAPLA* is faster than *APLA* up to n times.

Dimensionality Reduction Time (s)

	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM	SAX
HandOutlines	6.06e-04	1.00e+00	1.50e-04	1.13e-05	1.39e-05	6.20e-04	1.45e-05	6.22e-05
HouseTwenty	7.16e-04	1.00e+00	5.61e-04	8.58e-06	1.84e-05	5.79e-04	1.78e-05	5.96e-05
PigAirwayPressure	5.88e-04	1.00e+00	3.71e-04	8.80e-06	1.57e-05	5.69e-04	1.78e-05	6.28e-05
PigArtPressure	6.66e-04	1.00e+00	1.54e-04	8.62e-06	1.49e-05	5.96e-04	1.59e-05	5.73e-05
PigCVP	7.70e-04	1.00e+00	1.72e-04	8.64e-06	1.70e-05	6.25e-04	1.62e-05	5.80e-05
InlineSkate	4.66e-04	1.00e+00	1.65e-04	9.66e-06	1.71e-05	5.57e-04	1.88e-05	5.49e-05
EthanolLevel	9.86e-04	1.00e+00	1.80e-04	9.84e-06	1.87e-05	5.92e-04	1.84e-05	6.59e-05
CinCECGTorso	7.87e-04	1.00e+00	2.57e-04	8.83e-06	1.80e-05	5.34e-04	2.10e-05	6.24e-05
SemgHandGenderCh2	7.61e-04	1.00e+00	2.81e-03	9.33e-06	1.93e-05	5.61e-04	2.03e-05	6.52e-05
SemgHandMovementCh2	7.73e-04	1.00e+00	2.43e-03	9.55e-06	2.02e-05	5.94e-04	2.18e-05	6.82e-05
SemgHandSubjectCh2	6.95e-04	1.00e+00	2.81e-03	1.03e-05	2.05e-05	5.75e-04	2.06e-05	6.78e-05
ACSF1	6.40e-04	1.00e+00	2.85e-03	9.37e-06	1.54e-05	5.29e-04	1.56e-05	6.28e-05
EOGHorizontalSignal	5.45e-04	1.00e+00	1.83e-04	1.08e-05	1.78e-05	5.60e-04	1.86e-05	6.40e-05
EOGVerticalSignal	5.58e-04	1.00e+00	2.00e-04	9.58e-06	1.78e-05	5.41e-04	1.86e-05	6.33e-05
Haptics	5.54e-04	1.00e+00	1.60e-04	9.51e-06	1.63e-05	5.54e-04	1.74e-05	1.25e-04
Mallat	8.01e-04	1.00e+00	1.59e-04	1.08e-05	1.65e-05	5.70e-04	1.79e-05	6.54e-05
Phoneme	6.95e-04	1.00e+00	4.14e-04	9.25e-06	1.91e-05	5.68e-04	1.92e-05	6.40e-05
StarLightCurves	9.00e-04	1.00e+00	1.56e-04	9.48e-06	1.50e-05	5.61e-04	1.63e-05	6.51e-05
MixedShapesRegularTrain	7.11e-04	1.00e+00	1.58e-04	9.48e-06	1.64e-05	5.74e-04	1.71e-05	6.32e-05
MixedShapesSmallTrain	7.30e-04	1.00e+00	1.62e-04	9.71e-06	1.69e-05	5.79e-04	1.77e-05	6.79e-05

Table 3.4: Dimensionality reduction time comparison on 20 homogeneous datasets. The baseline dimensionality reduction method is *APLA*. The time unit is second.

Chapter 4

Lower Bound Distance Measure & Index Structure for *SAPLA*

4.1 Overview

This chapter proposes a lower bound distance measure between two time series to guarantee no false dismissals and tightness for adaptive-length segment dimensionality reduction methods, called $Dist_{PAR}$. Lower bound lemma can guarantee no-false-dismissals in k -NN search. Tightness of lower bound distance measure could help to improve k -NN performance. We will prove the lower bounding lemma and the tightness of $Dist_{PAR}$ for adaptive-length segment dimensionality reduction methods. The proposed dimensionality reduction techniques map two original time series into two low dimensional *SAPLA* spaces, the $Dist_{PAR}$ between them is a lower bound of the Euclidean distance between these two original time series. We proposed a Distance Based Covering with Convex Hull (DBCH-tree) for indexing and implemented the important node splitting and branch picking algorithms using the proposed lower bounding distance in the DBCH-tree construction.

4.1.1 Motivation

APCA [40] proposes two lower bounding distance measures that make adaptive-length segment dimensionality reduction methods indexable. One keeps lower bounding lemma, called $Dist_{LB}$, and another has tight Euclidean distance approximation but non-lower bounding, called $Dist_{AE}$. We propose $Dist_{PAR}$, which has guaranteed lower bounding lemma and tightness.

R-tree [31] splits the node by finding a minimum area waste and picks a branch with a mini-

mum area increase. However, homogeneous time series datasets are usually from the same sensors, such as stock price, ECG datasets, pressure and temperature. We find that *MBR* of homogeneous time series could cause overlap problems. This thesis proposes DBCH-tree for solving the overlap problem caused by *MBR* [40, 31]. The introduction of the *MBR* is in Appendix.

4.2 Lower Bounding Distance Measure $Dist_{PAR}$

We propose a lower bounding distance measure for adaptive-length segment dimensionality reduction methods (*SAPLA*, *APLA* [48], *APCA* [40]), denoted as $Dist_{PAR}$. $Dist_{PAR}$ can guarantee a lower bound of the Euclidean distance and a tight approximation of the Euclidean distance. Let \hat{q}_i, \hat{c}_i denote the i^{th} segment *SAPLA* representations in \hat{Q}, \hat{C} , suppose they have the same right endpoint and segment length denoted as l_i . Let \check{q}_j, \check{c}_j denote the reconstructed point in \hat{q}_i, \hat{c}_i by a linear function $a * j + b$, their Euclidean distance square is shown in Eq.(4.1).

$$Dist_S(\hat{q}_i, \hat{c}_i) = \sum_{j=0}^{l_i-1} (\check{q}_j - \check{c}_j)^2 = \frac{l_i(l_i-1)(2l_i-1)}{6} (\hat{q}_{a_i} - \hat{c}_{a_i})^2 + l_i(l_i-1)(\hat{q}_{a_i} - \hat{c}_{a_i})(\hat{q}_{b_i} - \hat{c}_{b_i}) + l_i(\hat{q}_{b_i} - \hat{c}_{b_i})^2 \quad (4.1)$$

Definition 4.2.1. ($Dist_{PAR}$) There are two *SAPLA* representations \hat{Q} and \hat{C} . Let \hat{Q}_R denote all segment right points r_i in \hat{Q} . Let \hat{C}_R denote all segment right points r_i in \hat{C} . We define $R = \hat{Q}_R \cup \hat{C}_R$. A partition process is carried out similarly to the split operation in Section 3.4.3. We regard the right endpoint of the shorter segment as the split point of the long segment. We could get new shorter segment representation coefficients a, b by Eq. (3.14)-(3.17). After partition, we could get partitioned \hat{Q}^P and \hat{C}^P that they have the same segment right endpoints $\hat{Q}_R^P = \hat{C}_R^P$. Thus, $Dist_{PAR}(\hat{Q}^P, \hat{C}^P)$ is defined in Eq. (4.2).

$$Dist_{PAR}(\hat{Q}^P, \hat{C}^P) = \sqrt{\sum_{i=0}^{R.size-1} Dist_S(\hat{q}_i^P, \hat{c}_i^P)} \quad (4.2)$$

Fig. 4.1 shows an example of $Dist_{PAR}$. The Euclidean distance of two original time series is 17. Two original time series of length 10 are reduced to 2 dimensions by adaptive-length segment dimensionality reduction method *SAPLA*. Fig. 4.1b shows two reconstructed time series from

SAPLA representation coefficients. $Dist_{PAR}$ partitioned these representations by Definition 4.2.1. $Dist_{PAR} = 14$ is a very tight approximation of the Euclidean distance. $Dist_{AE}[13] = 20$ does not lower bound the Euclidean distance. $Dist_{LB}[13] = 11$ is a less tight approximation of the Euclidean distance. We could find that $Dist_{LB} < Dist_{PAR} < Dist_{AE}$, which means $Dist_{PAR}$ is lower bound the Euclidean distance and tighter than $Dist_{LB}$. $Dist_{AE}$ is bigger than Euclidean distance, which breaks the lower bounding lemma.

The proof of $Dist_{PAR}$ lower bound the Euclidean distance is shown in Section 4.3. The proof of $Dist_{PAR}$ is a tight approximation of the Euclidean distance is shown in Section 4.4. The algorithm of $Dist_{PAR}$ is shown in Algorithm 4.2.1. We already know that $Dist_{AE}$ has $O(n)$ time complexity and that $Dist_{LB}$ also needs $O(n)$ time complexity for “projecting” new segment right endpoints [13]. Because of Eq.(3.14)(3.15)(3.16)(3.17), the worst time complexity of $Dist_{PAR}$ is smaller than $O(n)$.

4.3 Lower Bounding Lemma for $Dist_{PAR}$

Let Q and C denote two original time series. Let \hat{Q} and \hat{C} denote two *SAPLA* representations. Let \hat{Q}^p and \hat{C}^p denote the partitioned *SAPLA* representations of \hat{Q} and \hat{C} . In order to guarantee no false dismissal, $Dist_{PAR}(\hat{Q}, \hat{C})$ between two partitioned *SAPLA* representations \hat{Q}^p and \hat{C}^p should be smaller than the Euclidean distance $Dist(Q, C)$ between two time series Q and C . Let N' denote the partitioned dimension.

It is obvious that $Dist(Q, C)$ and $Dist_{PAR}(\hat{Q}, \hat{C})$ are summation of the distance of all segments. \hat{Q}^p and \hat{C}^p have same right endpoints. Thus, it is sufficient to prove one segment that lower bound distance from partitioned *SAPLA* is smaller than or equal to Euclidean distance of the same segments. For the first segment $\hat{q}_0 = \langle \hat{q}_a, \hat{q}_b, \hat{q}_r \rangle$, $\hat{c}_0 = \langle \hat{c}_a, \hat{c}_b, \hat{c}_r \rangle$, their right endpoints are equal ($\hat{q}_r = \hat{c}_r$). So, the proof of $Dist(Q_0, C_0) \geq Dist_{PAR}(\hat{Q}_0, \hat{C}_0)$ is same with $Dist(Q_0, C_0) \geq Dist_{PLA}(\hat{Q}_0, \hat{C}_0)$ (proved in [15]). Because the entire distance is summation of all segments, we can guarantee $Dist_{PAR}(\hat{Q}, \hat{C}) \leq Dist(Q, C)$.

4.4 Proof of $Dist_{PAR}$ Tightness

Let \hat{Q}^p and \hat{C}^p denote the partitioned *SAPLA* representations of \hat{Q} and \hat{C} . In order to prove the tightness of $Dist_{PAR}$, we need to guarantee $Dist_{LB} \leq Dist_{PAR}$.

\hat{Q}^p and \hat{C}^p have same right endpoints r_i . Let \hat{C}_R^p denote all the partitioned right endpoints

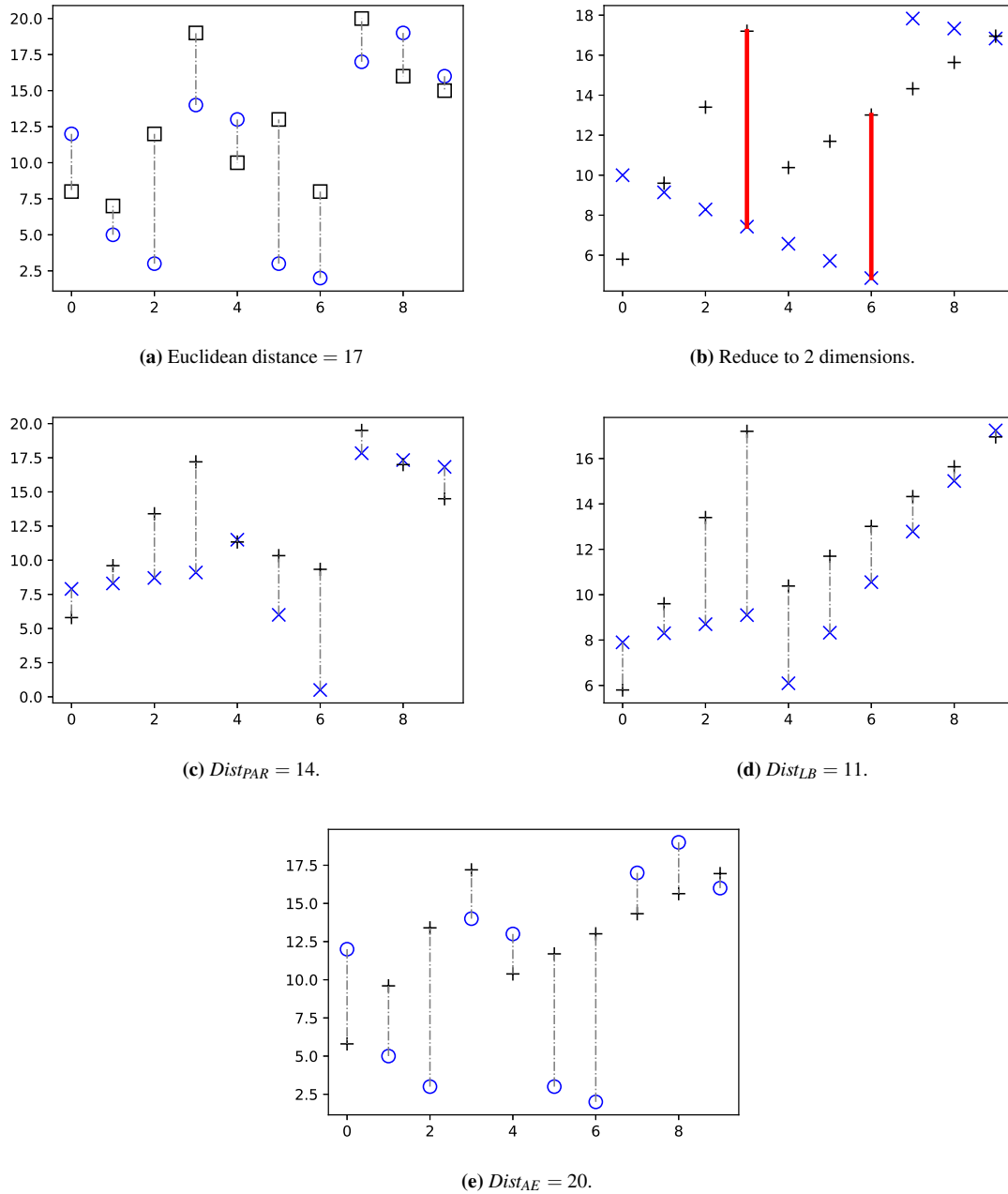


Figure 4.1: A visual comparison of lower bounding distance measures for adaptive-length segment representation. The blue color circle \circ and the black color box \square are original time series points. \times and $+$ are reconstructed time series points from representation coefficients. Fig.4.1a is the Euclidean distance between two original time series. Fig.4.1b is two reconstructed time series from the *SAPLA* dimensionality reduction method ($N = 2$). Red line is the position to partition for Fig.4.1c.

Algorithm 4.2.1: $Dist_{PAR}$ Lower Bounding Distance Measure

input : Two SAPLA representations coefficients denoted as \hat{Q}, \hat{C} from Q, C .
output: $D :=$ lower bounding distance of $\hat{Q} \hat{C}$.

- 1 $x, y, z, w, e :=$ temp segment;
- 2 $x \leftarrow \hat{q}_0, y \leftarrow \hat{c}_0$;
- 3 $D \leftarrow 0; i :=$ id in $\hat{C}; i \leftarrow 0$;
- 4 $v :=$ collection of segment id;
- 5 $v_f :=$ the first id in v ;
- 6 $v_e :=$ the last id;
- 7 $l :=$ segment length;
- 8 $r :=$ segment right endpoint;
- 9 **while** $x_r \leq \hat{q}_{r_{N-1}}$ or $y_r \leq \hat{c}_{r_{N-1}}$ **do**
- 10 **if** $x_r = y_r$ **then**
- 11 $D += Dist(x, y)$ by Eq.(4.1);
- 12 $x++, y++$;
- 13 **else**
- 14 **if** $x_r > y_r$ **then**
- 15 collect i from \hat{C} in v when $\hat{c}_{r_i} < x_r, i++$;
- 16 $tr \leftarrow x_r$;
- 17 **while** $v \neq \emptyset$ **do**
- 18 **if** $\hat{c}[v_f].l > \hat{c}[v_e].l$ **then**
- 19 $z_r \leftarrow \hat{c}[v_e - 1]_r$; // here is $\hat{c}[v_e]_r = x_r$;
- 20 $w_r \leftarrow x_r$; (Another case $\hat{c}[v_e]_r < x_r$ is similar, we put longer part of x in next iteration. Detail implementation is in [1]).
- 21 Compute a, b of z, w by x and Eq.(3.16)(3.17)
- 22 **if** $w_r = tr$ **then** $e \leftarrow w$
- 23 **else** $D += Dist(w, \hat{c}[v_e])$; $v_e --$;
- 24 $x \leftarrow z$;
- 25 **else** do similar way like above by Eq.(3.14)(3.15)
- 26 $x \leftarrow e$;
- 27 $y \leftarrow \hat{c}_i$;
- 28 **else** do similar way like above

in \hat{C}^p . Let \hat{Q}_R denote all segment right endpoints r_i in *SAPLA* representation \hat{Q} . Let \hat{C}_R denote all segment right endpoints r_i in *SAPLA* representation \hat{C} . $Dist_{LB}$ [40] lower bounding distance measure converts the original time series Q into a *SAPLA* representation \hat{Q}^{LB} that has the same segment right endpoints r_i with *SAPLA* representation \hat{C} . It is obvious that the *SAPLA* representation \hat{Q}_R^{LB} and \hat{C}_R have the same segment right endpoints, and their segment right endpoints are all included in partitioned *SAPLA* representation, which means $\hat{Q}_R^{LB} = \hat{C}_R \subseteq \hat{C}_R^p$. Thus, we can conclude that any segment \hat{c}_i in *SAPLA* representation \hat{C} could be computed by one segment or merged (Eq. (3.11) (3.12)) from several segments in partitioned *SAPLA* representation \hat{C}^p . Meanwhile, let $l_i = r_i - r_{i-1}$ denote the segment length. $\exists m \in [0, k]$ satisfies $\sum_{j=m}^k \hat{C}_{l_j}^p = \hat{C}_{l_i}$. It is sufficient to prove one segment that $Dist_{LB}(\hat{q}_i, \hat{c}_i)$ is smaller than or equal to the $Dist_{PAR}$ of one segment or several segments. If the segment \hat{c}_i has the same segment right endpoint as \hat{c}_k^p and their segment length are equal, their distance will be equal. If there are more than two segments in partitioned *SAPLA* representation \hat{C}^p that their summation of segments length are equal to segment length of \hat{c}_i in *SAPLA* representation \hat{C} , these segments can be merged into two segments \hat{c}_{k-1}^p and \hat{c}_k^p . The above conclusions can also work on partitioned *SAPLA* representation \hat{Q}^p and projected *SAPLA* representation \hat{Q}^{LB} . Thus, we could get $Dist_{LB} \leq Dist_{PAR}$.

We will prove the tightness of $Dist_{PAR}$ on one segment. Let \hat{q} and \hat{c} denote the first segment in $Dist_{LB}$ distance approximation. Suppose there are two segments \hat{c}_0, \hat{c}_1 in $Dist_{PAR}$ distance approximation that their summation of segment length is equal to the length of segment \hat{c} in *SAPLA* representation \hat{C} , denoted as $l = l_0 + l_1$. $r = r_1$. We could regard the \hat{c} as the *SAPLA* representation of reconstructed time series from \hat{c}_0 plus \hat{c}_1 . Their reconstructed time series are denoted as $\check{C}_0 + \check{C}_1$. If $\check{C}_0 + \check{C}_1$ and $\check{Q}_0 + \check{Q}_1$ are regarded as two original time series, \hat{q} and \hat{c} are regarded as their *SAPLA* representation, we will get $Dist(\check{C}_0 + \check{C}_1, \check{Q}_0 + \check{Q}_1) \geq Dist_{PAR}(\hat{q}, \hat{c})$ (proved in Section 4.3). According to Section 4.3, $Dist(\check{C}_0, \check{Q}_0) \geq Dist_{PAR}(\hat{q}_0, \hat{c}_0)$ and $Dist(\check{C}_1, \check{Q}_1) \geq Dist_{PAR}(\hat{q}_1, \hat{c}_1)$. Let Q and C denote the original time series in first segment of $Dist_{LB}$ distance approximation. Because we already prove $Dist_{LB} \leq Dist_{PAR}$ above, we will have $(Dist(C, Q))^2 = (Dist(C_0, Q_0))^2 + (Dist(C_1, Q_1))^2 \geq (Dist_{PAR}(\hat{q}_0, \hat{c}_0))^2 + (Dist_{PAR}(\hat{q}_1, \hat{c}_1))^2 \geq (Dist_{LB}(\hat{q}, \hat{c}))^2$. We complete the proof of tightness on the first segment. Because the entire distance is summation of those segments, we can conclude $Dist_{LB}(\hat{Q}, \hat{C}) \leq Dist_{PAR}(\hat{Q}, \hat{C}) \leq Dist(Q, C)$.

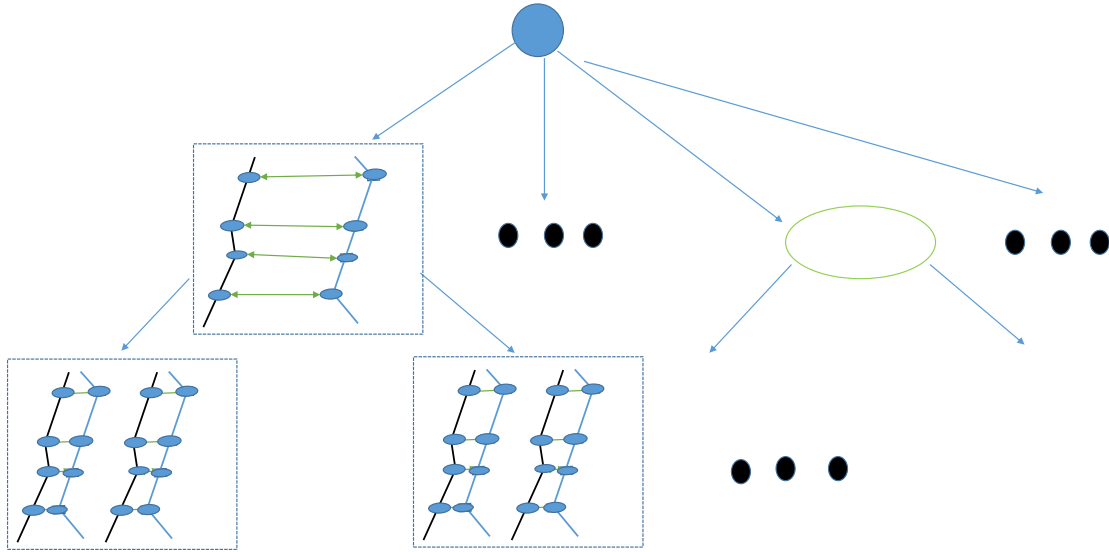


Figure 4.2: Illustration of hierarchical DBCH-tree.

4.5 DBCH-tree

This thesis proposes a height-balanced index structure for covering a set of original time series or representation coefficients by distance. This index structure referred to as Distance Based Covering with Convex Hull (DBCH-tree), could efficiently improve indexing. DBCH-tree could build a tighter index structure than *MBR*-based R-tree [13] [40]. For a set of n -length time series $S = \{C[1], C[2], \dots, C[T]\}$, where $C[i] = \{c_{i0}, c_{i1}, \dots, c_{i(n-1)}\}$. In this thesis, we represent each time series by *SAPLA* dimensionality reduction techniques. We could get a set of N segment number *SAPLA* representations $\hat{S}[] = \{\hat{C}[1], \hat{C}[2], \dots, \hat{C}[T]\}$, where $\hat{C}[i] = \{\hat{c}_{i0}, \hat{c}_{i1}, \dots, \hat{c}_{i(N-1)}\}$. Let V denote the DBCH structure of \hat{S} that $(V.u, V.l) = \operatorname{argmax}_{x,y \in \hat{S}[]} \operatorname{Dist}(x,y)$. Let $V.u$ and $V.l$ denote one pair representations with the maximum lower bounding distance. Figure 4.2 shows what a DBCH structure looks like. Algorithm 4.5.1 shows how to build a DBCH structure. The difference between DBCH-tree and R-tree is that R-tree uses *MBR* to contain entries. DBCH-tree uses the convex hull like Figure 4.2 shows to contain entries. R-tree splits a node or picks a branch based on waste area. DBCH-tree splits a node or picks a branch based on distance.

Algorithm 4.5.1: A DBCH Structure for Time Series

```

input : One time series representation denoted as  $\hat{E}'$ .
output: A DBCH structure.
1 Node := the inserted node.
2 Node +=  $\hat{E}'$ ;
3  $Dist_{max} \leftarrow -1$ ;
4 for  $i = 0; i < Node.size; i++$  do
5   ┌ Apply partition on  $\hat{E}_i$  and  $\hat{E}_{Node.size-1}$ 
6 for  $i = 0; i < Node.size; i++$  do
7   ┌ for  $j = i+1; j < Node.size; j++$  do
8     ┌ //Lower bounding distance measures could be  $Dist_{PAR}, Dist_{PAA}, Dist_{AE}, Dist_{LB},$ 
9       ┌  $Dist_{Cheby}$  and  $Dist_{SAX}$ .
10      ┌ Compute lower bounding distance  $Dist$  between  $\hat{E}_i$  and  $\hat{E}_j$ .
11      ┌ if  $Dist_{max} < Dist$  then
12        ┌  $Dist_{max} \leftarrow Dist$ ;
13        ┌  $Node.l \leftarrow \hat{E}_i$ ;
14        ┌  $Node.u \leftarrow \hat{E}_j$ ;

```

4.6 Node Splitting and Branch Picking

Fig. 4.3 shows how homogeneous time series overlap each other. The most popular node splitting algorithm in R-tree is the quadratic method. Node splitting algorithm in R-tree select two entries as seeds that will cause the biggest waste area if we put them into *MBR* [31] [40]. Like Fig. 4.3a shows, node splitting in R-tree attempts to find a small-area split. R-tree scans each candidate entry and computes the combined *MBR* with each seed. R-tree assigns the candidate entry into the appropriate group that cause less waste area until all entries are assigned. If the number of assigned entries in one group is fewer than the user-defined minimum number of entries in one node, the rest entries are put into that group directly. The branch picking algorithm picks a branch with a minimum area increase. We propose improved node splitting and branch picking algorithms for time series. We split node and pick branch by lower bounding distance instead of *MBR* area. We can avoid serious overlap problem in node splitting and branch picking process.

As the improvement of node splitting in DBCH-tree, the main difference in comparison with R-tree is we focus on lower bounding distance measure, not waste area. When one node needs to split, we first choose the pair with a maximum lower bounding distance and denote them as seed₁ and seed₂. Then, we compute the lower bounding distance between the rest entries and these two seeds. If they are close to seed₁, we put them in node₁. Otherwise, we put them in

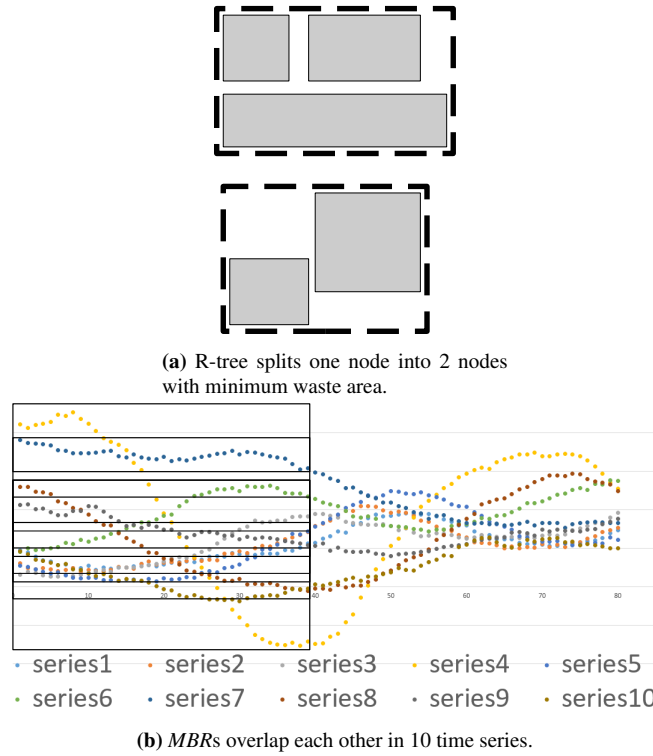


Figure 4.3: This example (10 time series) shows how homogeneous time series *MBRs* overlap each other.

$node_2$. Branch picking algorithm does similar work that chooses a branch with a minimum distance increase. Algorithm 4.6.1 shows how to split node by distance and Algorithm 4.6.2 shows how to pick a branch by distance.

R-tree uses a minimum bounding rectangle (*MBR*) [31] [40] to contain the entry of points. DBCH-tree uses a minimum distance convex hull to contain the entry of points. Given a query time series, we can construct a DBCH-tree for representation points. We can use this DBCH-tree to get K nearest neighbors (k -NN) queries efficiently. Let R denote one DBCH structure. Let C denote one time series. The distance between time series and one DBCH structure is shown in Algorithm 4.6.3. *GEMINI* k -NN algorithm [65] [34] [40] uses a small-distance-first manner with query point Q over DBCH-tree. This algorithm uses a priority queue to navigate nodes in the Tree from small distance to big distance. We use lower bounding distance measures, such as $Dist_{PAR}$ proposed in Section 4.2.

Algorithm 4.6.1: Node Splitting by Distance Measures.

```

input : One new entry  $E'$ .
output: Two new nodes.
1  $Node :=$  the inserted node.
2  $Node_0, Node_1 :=$  two new nodes.
3  $MaxFill :=$  The maximum number of the entries in one node.
4  $MinFill :=$  The minimum number of the entries in one node.
5  $seed_0, seed_1 :=$  seeds for node splitting // Pick Seeds.
6  $Node += E'$ ;
7  $Dist_{max} \leftarrow -1$ ;
8 for  $i = 0; i < Node.size; i++$  do
9   for  $j = i+1; j < Node.size; j++$  do
10    //Lower bounding distance measures could be  $Dist_{PAR}, Dist_{PAA}, Dist_{AE}, Dist_{LB},$ 
11     $Dist_{Cheby}$  and  $Dist_{SAX}$ .
12    compute lower bounding distance  $Dist$  between  $E_i$  and  $E_j$ .
13    if  $Dist_{max} < Dist$  then
14       $Dist_{max} \leftarrow Dist$ ;
15       $seed_0 \leftarrow i$ ;
16       $seed_1 \leftarrow j$ ;
17 for  $i = 0; i < Node.size; i++$  do
18   if  $E_i$  is not seed and  $E_i$  has not been assigned then
19     if  $E_i$  is close to  $seed_0$  then
20       if  $Node_0.size + MinFill > MaxFill$  then
21          $Node_1 += E_i$ ;
22          $Node_1.size ++$ ;
23       else
24          $Node_0 += E_i$ ;
25          $Node_0.size ++$ ;
26     if  $E_i$  is close to  $seed_1$  then
27       if  $Node_1.size + MinFill > MaxFill$  then
28          $Node_0 += E_i$ ;
29          $Node_0.size ++$ ;
30       else
31          $Node_1 += E_i$ ;
32          $Node_1.size ++$ ;

```

Algorithm 4.6.2: Branch Picking by Distance Measures.

```

input : One new entry  $E'$ .
output: The branch to pick.
1  $first_{time} \leftarrow true$ ; // get the initial branch
2  $difference_{best} \leftarrow -1$ ;
3 for  $i = 0; i < Node.size; i++$  do
4   Combine  $E'$  with  $branch_i$ ;
5   //Lower bounding distance measures could be  $Dist_{PAR}, Dist_{PAA}, Dist_{AE}, Dist_{LB},$ 
    $Dist_{Cheby}$  and  $Dist_{SAX}$ .
6   compute the combined maximum lower bounding distance  $Dist_{max}$  after combination;
7    $difference_{current} = Dist_{max} - Dist_{current}$ ;
8   if  $difference_{current} < difference_{best}$  or  $first_{time}$  then
9      $branch_{best} \leftarrow i$ ;
10     $distance_{best} \leftarrow Dist_{max}$ ;
11     $difference_{best} \leftarrow difference_{current}$ ;
12     $first_{time} \leftarrow false$ ;
13  else if  $difference_{current} == difference_{best}$  and  $Dist_{max} < distance_{best}$  then
14     $branch_{best} \leftarrow i$ ;
15     $distance_{best} \leftarrow Dist_{max}$ ;

```

Algorithm 4.6.3: Distance Computation Between One Time Series and A DBCH Structure.

```

input : One time series  $C$ .
         One DBCH structure  $R$ .
output:  $Dist_{MBR}$ 
1 if  $Dist_{PAR}(C, R.u) > Dist_{PAR}(R.l, R.u)$  then
2   if  $Dist_{PAR}(C, R.l) > Dist_{PAR}(C, R.u)$  then
3      $Dist_{MBR} \leftarrow Dist_{PAR}(C, R.u)$ 
4   else if  $Dist_{PAR}(C, R.l) \leq Dist_{PAR}(C, R.u)$  then
5      $Dist_{MBR} \leftarrow Dist_{PAR}(C, R.l)$ 
6 else if  $Dist_{PAR}(C, R.l) > Dist_{PAR}(R.l, R.u)$  then
7   if  $Dist_{PAR}(C, R.l) < Dist_{PAR}(C, R.u)$  then
8      $Dist_{MBR} \leftarrow Dist_{PAR}(C, R.l)$ 
9   else if  $Dist_{PAR}(C, R.l) \geq Dist_{PAR}(C, R.u)$  then
10     $Dist_{MBR} \leftarrow Dist_{PAR}(C, R.u)$ 
11 else
12    $Dist_{MBR} \leftarrow 0$ 

```

4.7 Experimental Evaluation

This section will evaluate the tightness of lower bound, pruning power (ρ) and k -NN time. We have implemented $Dist_{PAR}$, $Dist_{AE}$, $Dist_{LB}$, $Dist_{PAA}$, $Dist_{PLA}$, $Dist_{CHEBY}$, and $Dist_{SAX}$, R-tree, DBCH-tree by C++ [1]. Their summaries are shown in Table 2.7. The processor is Intel(R) Core(TM) i5-7600 CPU @ 3.50 GHz. RAM is 8 GB. We evaluated datasets by VS2019 in Windows 10 system. We evaluated all homogeneous datasets in [18] that time series length is bigger than 1024 (20 datasets), they are in Table 3.1. The parameter is $\{12 \leq M \leq 24, M+ = 6\}$, $\{2 \leq K \leq 64, K* = 2\}$. We evaluate the tightness of lower bound, the equation could be got by Eq. (4.3). We also evaluate the indexing performance by testing pruning power (ρ). ρ could be got by Eq. (4.4), which can avoid implementation bias.

4.7.1 Comparison on Tightness of Lower Bound Distance

According to the lower bounding lemma, the lower bound distance between two representations should always be smaller than the Euclidean distance between their original time series. We want to get the tightest lower bounds, and it helps us to find the K nearest neighbours which have the small Euclidean distances. The tightness of lower bound computation is shown in Eq. 4.3.

$$Tightness\ of\ Lower\ Bound = \frac{Lower\ Bound\ Distance(\hat{C}, \hat{D})}{Dist_{euc}(C, D)} \quad (4.3)$$

Table 4.1 shows the average tightness of lower bound distance for a pair of time series. Each dataset has one hundred time series. The representation coefficient number has three types. We divide $3 * 99 = 297$ for each lower bound distance measurement in each dataset. Because the lower bound distance measurement should follow the lower bounding lemma, the average tightness of lower bound should be smaller than one in Table 4.1. The light blue color rectangles in Table 4.1 show that $Dist_{AE}$ does not follow the lower bounding lemma. The lower bounding distance measurement will cause false dismissals in k -NN search if it cannot follow the lower bounding lemma. So, we focus on comparing the tightness of lower bound between $Dist_{PAR}$ and $Dist_{LB}$. Table 4.1 shows $Dist_{PAR}$ is tighter than $Dist_{LB}$ in each dataset. Table 4.2 shows the average rank of the lower bounding distance measurements $Dist_{PAR}$, $Dist_{AE}$ and $Dist_{LB}$ in each dataset. The bottom row is the average rank of each lower bounding distance measurement. Though $Dist_{AE}$ ranks number one, it does not follow the lower bounding lemma. We could find $Dist_{PAR}$ is tighter than $Dist_{AE}$ in two datasets *EOGHorizontalSignal* and *EOGVerticalSignal*. We could find $Dist_{PAR}$ ranks better

than $Dist_{LB}$ in each dataset.

Figure 4.4 shows the scatter plot of our proposed lower bounding distance measurement $Dist_{PAR}$ and baseline lower bounding distance measurement $Dist_{LB}$. We could find that all black points fall below the light red colour diagonal line which means $Dist_{PAR}$ is tighter than $Dist_{AE}$ in each dataset. Figure 4.5 is a critical difference diagram for the tightness of lower bound. The best ranks are to the left. The methods on the left side are better than the right side methods. Because we need to compare multiple lower bound distance measurements in multiple datasets and our proposed lower bound distance measurement $Dist_{PAR}$ with other lower bound distance measurements, the post-hoc analysis is Bonferroni-Dunn Test. The baseline method name is $Dist_{PAR}$. The predefined threshold value α is 0.05. The number of datasets is twenty. The tightness average rank is $Dist_{PAR} : 1.9$, $Dist_{AE} : 1.1$, $Dist_{LB} : 3.0$. The critical difference value is 0.7086664236437339. The null-hypothesis is that all lower bound distance measurements perform equally well, and the performance differences are random. If the difference in average ranks between two lower bound distance measurements is smaller than the critical difference value, their performance difference is not significant (horizontal line). We could find that these lower bound distance measurements are significantly different.

We use the Wilcoxon signed ranks test for comparison of paired lower bound distance measurements between our proposed lower bound distance measurement $Dist_{PAR}$ and other lower bound distance measurements. The Wilcoxon signed ranks test is a non-parametric statistical hypothesis test. We do not use the t-test because the tightness of lower bound differences of paired lower bound distance measurements is not a normal distribution. The null-hypothesis is that the paired lower bound distance measurements perform differently, and the performance differences are not random. We define the predefined threshold value α as 0.05. The p-values of Wilcoxon signed ranks test are ($Dist_{PAR}$ vs $Dist_{AE}$, $p = 1.335144 \times 10^{-5} < 0.05$), ($Dist_{PAR}$ vs $Dist_{LB}$, $p = 1.907349 \times 10^{-6} < 0.05$). we would reject the null hypothesis.

Tightness of Lower Bound

	DistPAR	DistAE	DistLB
HandOutlines	0.9863501683501684	1.0223097643097643	0.9357609427609427
HouseTwenty	0.8386397306397306	0.9338114478114478	0.7374141414141414
PigAirwayPressure	0.972922558922559	0.9743838383838384	0.9336902356902357
PigArtPressure	0.734989898989899	0.9296397306397307	0.5724814814814815
PigCVP	0.8273569023569023	0.9038350168350169	0.7126835016835017
InlineSkate	0.9934478114478114	0.997026936026936	0.9795218855218856
EthanolLevel	0.9488484848484848	0.982986531986532	0.831043771043771
CinCECGTorso	0.9753804713804713	0.9840606060606061	0.8409595959595959
SemgHandGenderCh2	0.4534747474747474	0.7971111111111111	0.28014511784511786
SemgHandMovementCh2	0.4824983164983165	0.7788080808080808	0.29883030303030306
SemgHandSubjectCh2	0.29163063973063974	0.7732154882154882	0.22817171717171716
ACSF1	0.27699730639730635	14.116161616161616	0.25282659932659934
EOGHorizontalSignal	0.9929057239057238	0.9921582491582491	0.9844343434343434
EOGVerticalSignal	0.9901851851851852	0.9877542087542087	0.9806464646464647
Haptics	0.9217542087542089	0.9632289562289562	0.7977609427609428
Mallat	0.8015555555555556	1.5113198653198654	0.6896161616161616
Phoneme	0.43743771043771046	0.799043771043771	0.35943434343434344
StarLightCurves	0.9627205387205388	0.9819057239057238	0.9213501683501684
MixedShapesRegularTrain	0.906902356902357	0.9466767676767677	0.7880404040404041
MixedShapesSmallTrain	0.9297575757575757	0.9622659932659933	0.8232962962962963

Table 4.1: Average tightness of lower bound comparison on the twenty homogeneous datasets. The light blue rectangle is the lower bound distance bigger than one, breaking the lower bounding lemma.

Tightness of Lower Bound Average Rank			
	DistPAR	DistAE	DistLB
HandOutlines	2.0	1.0	3.0
HouseTwenty	2.0	1.0	3.0
PigAirwayPressure	2.0	1.0	3.0
PigArtPressure	2.0	1.0	3.0
PigCVP	2.0	1.0	3.0
InlineSkate	2.0	1.0	3.0
EthanolLevel	2.0	1.0	3.0
CinCECGTorso	2.0	1.0	3.0
SemgHandGenderCh2	2.0	1.0	3.0
SemgHandMovementCh2	2.0	1.0	3.0
SemgHandSubjectCh2	2.0	1.0	3.0
ACSF1	2.0	1.0	3.0
EOGHorizontalSignal	1.0	2.0	3.0
EOGVerticalSignal	1.0	2.0	3.0
Haptics	2.0	1.0	3.0
Mallat	2.0	1.0	3.0
Phoneme	2.0	1.0	3.0
StarLightCurves	2.0	1.0	3.0
MixedShapesRegularTrain	2.0	1.0	3.0
MixedShapesSmallTrain	2.0	1.0	3.0
Average Rank	1.9	1.1	3.0

Table 4.2: Tightness of lower bound comparison on the twenty homogeneous datasets. $Dist_{AE}$ does not follow the lower bounding lemma.

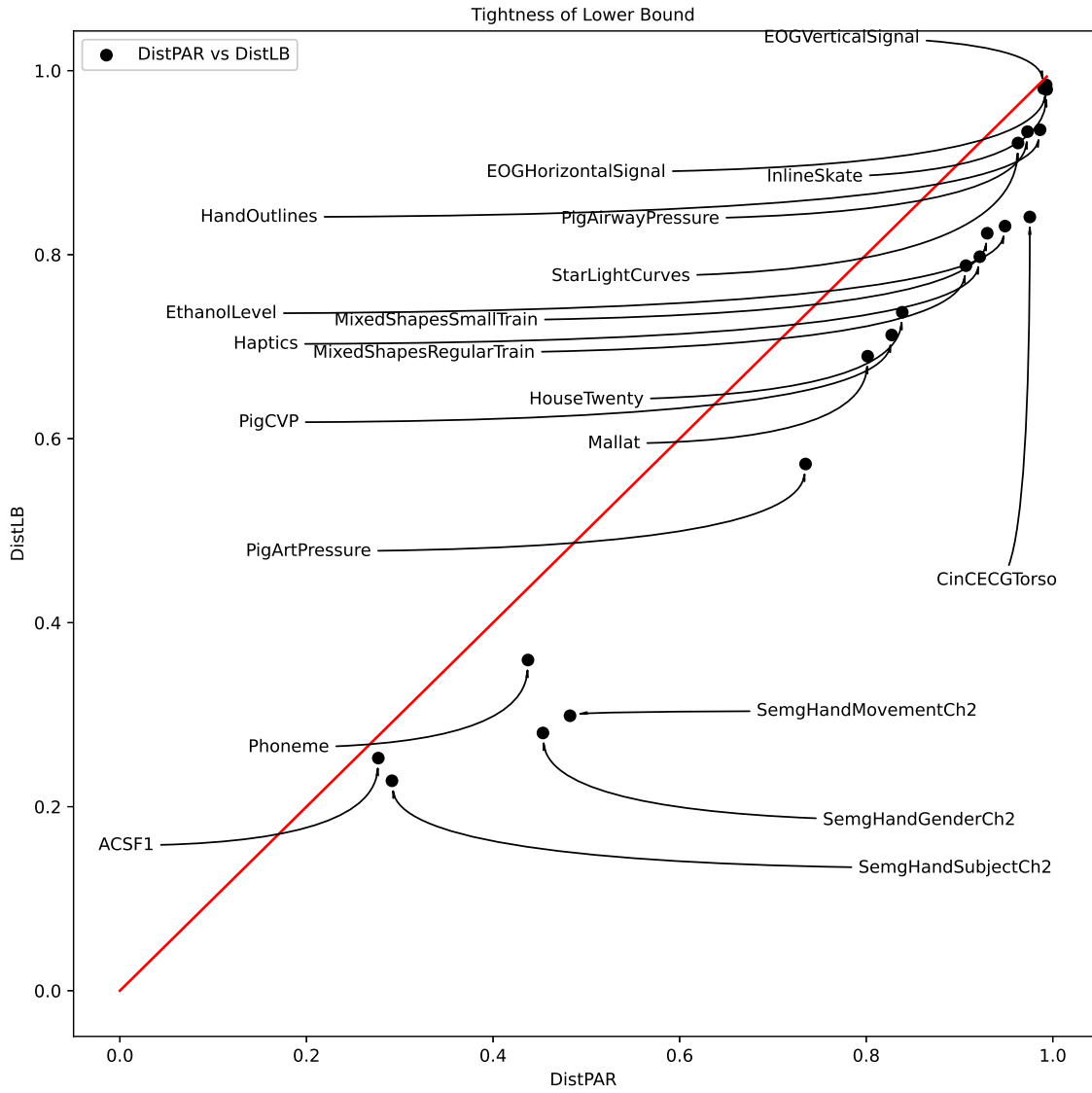


Figure 4.4: Scatter plot between $Dist_{PAR}$ and $Dist_{LB}$ on 20 homogeneous datasets. The dimensionality reduction method is *SAPLA*.

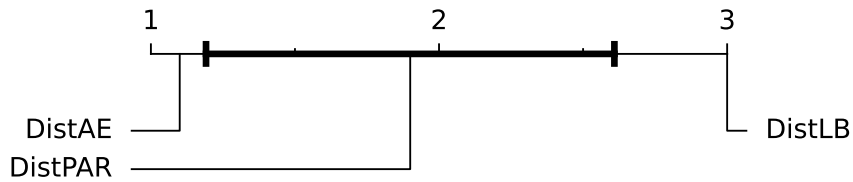


Figure 4.5: The critical difference diagram of tightness of lower bound comparison on 20 homogeneous datasets. The dimensionality reduction method is *SAPLA*.

4.7.2 Comparison on Pruning Power (ρ)

Pruning power is the fraction of the database that has to be measured. When we do a similarity search, we hope all operations are limited in memory. We do not want to extract all the original time series $C = \{c_0, c_1, \dots, c_{n-1}\}$ from the database that is stored in disk. The optimal dimensionality reduction method, lower bounding distance measurement and index structure will only need to extract k nearest original time series from the disk. Therefore, we introduce the pruning power to evaluate each method.

We compare the pruning power ρ of eight dimensionality reduction methods to avoid implementation bias. The experiment results of the pruning power are free of the possibility of implementation bias. According to Eq. 4.4, the smaller of pruning power, the better performance of this method. The value of the pruning power for any transformation depends on the datasets and is independent of implementation bias, including programming language, hardware configuration, page size and spatial access method.

We evaluated the pruning power of *SAPLA*, *PAA*, *PLA*, *APCA*, *APLA*, *SAX*, *PAALM* and *CHEBY* in Section 4.7.2. *CHEBY* k-NN algorithm is different from other dimensionality reduction methods. And *CHEBY* k-NN algorithm involves repeated *I/O* access, thus pruning power of *CHEBY* sometimes bigger than one.

$$\rho = \frac{\text{the number of time series which have to be measured}}{\text{all time series number}} \quad (4.4)$$

Table 4.3 and Table 4.4 show the pruning power of DBCH-tree and R-tree for these eight dimensionality reduction methods. Pruning power is used to evaluate the performance of k -NN search. There are 20 homogeneous datasets. The baseline index structure is R-tree. The red color rows are the dimensionality reduction methods that use DBCH-tree to do k -NN search for each dataset, and the white color rows are the dimensionality reduction methods that use R-tree to do k -NN search for each dataset. The index structure has smaller value in Table 4.3 and Table 4.4 means it needs to extract fewer original time series from the hard disk. Table 4.3 and Table 4.4 show the superiority of *SAPLA* and *APLA* in the DBCH-tree. We have proved that $Dist_{PAR}$ proposed in Section 4.4 is tighter than $Dist_{LB}$. DBCH-tree with $Dist_{PAR}$ has smaller pruning power than R-tree with $Dist_{LB}$ for adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA*. Equal-length segment dimensionality reduction methods *PLA*, *PAA* and *CHEBY* have almost the same pruning in R-tree and DBCH-tree because their lower bounding distance measure is the same in R-tree and

DBCH-tree, which means their lower bounding lemma is not changed. Because *PAALM* and *SAX* do not consider max deviation, their lower bounding distance measurements $Dist_{PAR}$ cannot tightly lower bound the Euclidean distance between the original time series. Our improved node splitting and branch picking algorithms (Section 4.6) in DBCH-tree cannot pick similar entries in one node.

Table 4.5 shows the pruning power ranks comparison of each dimensionality reduction method in DBCH-tree and R-tree index structures for each dataset. The red color columns use DBCH-tree in k -NN search. The white color columns use R-tree in k -NN search. The bottom row is the average rank of each dimensionality reduction method in DBCH-tree and R-tree index structures. We could find DBCH-tree strongly affects adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA*. For example, when *SAPLA* transfers the index structure from R-tree to DBCH-tree in k -NN search, the average rank of *SAPLA* changes from 7.425 to 1.575. The average rank of *APLA* changes from 8.725 to 1.825. The average rank of *APCA* changes from 10.5 to 4.95. The pruning power average ranks of these dimensionality reduction methods transfer the index structure from R-tree to DBCH-tree in k -NN search are *SAPLA* (7.425 \rightarrow 1.575), *APLA* (8.725 \rightarrow 1.825), *APCA* (10.5 \rightarrow 4.95), *PLA* (7.35 \rightarrow 6.05), *PAA* (8.3 \rightarrow 7.0), *CHEBY* (14.0 \rightarrow 14.0), *PAALM* (13.05 \rightarrow 13.9), *SAX* (9.05 \rightarrow 8.3). Because *CHEBY* dimensionality reduction method directly uses Chebyshev coefficients as the entries in index structure. DBCH-tree has no effect on Chebyshev coefficients. *CHEBY* dimensionality reduction method has the same pruning power in R-tree and DBCH-tree. We could find the pruning power of three adaptive-length segment dimensionality reduction methods and three equal-length segment dimensionality reduction methods becoming better when using DBCH-tree to replace R-tree.

	Index	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM	SAX
HandOutlines	R-tree	20.52	21.05	36.69	20.23	20.89	39.13	71.13	57.23
	DBCH-tree	19.21	19.44	25.15	19.97	20.62	39.13	75.76	55.40
HouseTwenty	R-tree	68.74	74.83	75.51	73.21	74.02	95.94	88.78	69.56
	DBCH-tree	41.23	44.06	46.73	70.28	71.25	95.94	90.00	64.01
PigAirwayPressure	R-tree	36.68	37.64	41.83	35.11	35.57	56.09	58.79	42.97
	DBCH-tree	33.91	33.91	34.65	34.63	34.93	56.09	69.46	42.26
PigArtPressure	R-tree	56.07	57.62	57.87	43.01	43.41	73.82	84.15	35.03
	DBCH-tree	30.98	31.21	34.00	41.41	41.87	73.82	85.38	33.92
PigCVP	R-tree	71.30	71.36	76.46	64.88	66.08	94.39	89.98	52.93
	DBCH-tree	39.29	39.99	45.57	62.81	64.05	94.39	90.00	50.73
InlineSkate	R-tree	20.45	20.82	27.12	19.60	20.62	38.18	62.25	34.74
	DBCH-tree	19.17	19.22	20.78	19.51	20.43	38.18	83.57	33.26
EthanolLevel	R-tree	27.96	25.87	40.39	34.14	34.16	47.73	74.15	70.51
	DBCH-tree	19.86	19.84	23.02	32.13	31.18	47.73	83.94	76.34
CinCECGTorso	R-tree	42.74	43.28	48.10	52.43	53.56	89.42	71.66	62.27
	DBCH-tree	23.31	22.64	24.68	50.88	52.92	89.42	83.48	63.08
SemgHandGenderCh2	R-tree	90.00	90.00	90.00	90.00	90.00	108.90	90.00	89.78
	DBCH-tree	85.30	85.92	90.00	90.00	90.00	108.90	90.00	89.66
SemgHandMovementCh2	R-tree	90.00	90.00	90.00	90.00	90.00	108.90	90.00	89.95
	DBCH-tree	85.26	86.38	89.91	90.00	90.00	108.90	90.00	89.95

Table 4.3: Pruning power comparison on the first ten homogeneous datasets. The white color rectangle \square is original R-tree. The red color rectangle \blacksquare is DBCH-tree.

	Index	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM	SAX
SemgHandSubjectCh2	R-tree	90.00	90.00	90.00	90.00	90.00	108.90	90.00	89.57
	DBCH-tree	88.00	88.66	90.00	90.00	90.00	108.90	90.00	89.57
ACSF1	R-tree	44.60	44.86	50.85	48.41	45.23	69.32	25.41	20.45
	DBCH-tree	36.15	33.35	48.10	48.18	47.90	69.32	25.44	19.92
EOGHorizontalSignal	R-tree	20.55	20.93	24.42	20.88	21.54	41.48	52.75	33.36
	DBCH-tree	19.24	19.33	19.77	20.66	20.86	41.48	79.42	32.86
EOGVerticalSignal	R-tree	20.47	21.13	23.37	21.29	22.06	43.11	56.21	33.51
	DBCH-tree	19.34	19.38	19.54	20.73	21.46	43.11	78.46	32.88
Haptics	R-tree	32.66	39.34	42.79	25.88	27.68	46.22	76.63	51.59
	DBCH-tree	21.68	24.37	26.32	24.66	26.19	46.22	87.29	48.10
Mallat	R-tree	49.06	50.41	59.91	43.23	43.95	66.67	82.91	64.56
	DBCH-tree	29.03	25.58	40.08	40.61	41.29	66.67	87.46	62.90
Phoneme	R-tree	90.00	90.00	90.00	90.00	89.99	108.90	90.00	84.86
	DBCH-tree	81.16	81.41	83.36	90.00	89.97	108.90	90.00	83.82
StarLightCurves	R-tree	26.01	26.01	31.99	23.11	24.33	43.97	59.50	37.84
	DBCH-tree	19.96	19.91	21.43	22.46	23.44	43.97	77.33	37.49
MixedShapesRegularTrain	R-tree	37.71	37.82	47.76	27.68	29.11	52.43	80.08	42.41
	DBCH-tree	23.95	23.32	26.59	26.38	27.87	52.43	88.77	40.65
MixedShapesSmallTrain	R-tree	37.71	37.82	47.76	27.68	29.11	52.43	80.08	42.41
	DBCH-tree	23.95	23.32	26.59	26.38	27.87	52.43	88.77	40.65

Table 4.4: Pruning power comparison on the last ten homogeneous datasets. The white color rectangle \square is original R-tree. The red color rectangle \blacksquare is DBCH-tree.

Prune Power Average Rank

	SAPLA	SAPLA*	APLA	APLA*	APCA	APCA*	PLA	PLA*	PAA	PAA*	CHEBY	CHEBY*	PAALM	PAALM*	SAX	SAX*
HandOutlines	5.0	1.0	8.0	2.0	10.0	9.0	4.0	3.0	7.0	6.0	11.5	11.5	15.0	16.0	14.0	13.0
HouseTwenty	5.0	1.0	11.0	2.0	12.0	3.0	9.0	7.0	10.0	8.0	15.5	15.5	13.0	14.0	6.0	4.0
PigAirwayPressure	8.0	1.5	9.0	1.5	10.0	4.0	6.0	3.0	7.0	5.0	13.5	13.5	15.0	16.0	12.0	11.0
PigArtPressure	10.0	1.0	11.0	2.0	12.0	4.0	8.0	6.0	9.0	7.0	13.5	13.5	15.0	16.0	5.0	3.0
PigCVP	10.0	1.0	11.0	2.0	12.0	3.0	8.0	6.0	9.0	7.0	15.5	15.5	13.0	14.0	5.0	4.0
InlineSkate	6.0	1.0	9.0	2.0	10.0	8.0	4.0	3.0	7.0	5.0	13.5	13.5	15.0	16.0	12.0	11.0
EthanollLevel	5.0	2.0	4.0	1.0	10.0	3.0	8.0	7.0	9.0	6.0	11.5	11.5	14.0	16.0	13.0	15.0
CinCEGTorso	4.0	2.0	5.0	1.0	6.0	3.0	8.0	7.0	10.0	9.0	15.5	15.5	13.0	14.0	11.0	12.0
SemgHandGenderCh2	9.5	1.0	9.5	2.0	9.5	9.5	9.5	9.5	9.5	9.5	15.5	15.5	9.5	9.5	4.0	3.0
SemgHandMovementCh2	10.0	1.0	10.0	2.0	10.0	3.0	10.0	10.0	10.0	10.0	15.5	15.5	10.0	10.0	4.5	4.5
SemgHandSubjectCh2	9.5	1.0	9.5	2.0	9.5	9.5	9.5	9.5	9.5	9.5	15.5	15.5	9.5	9.5	3.5	3.5
ACSF1	7.0	6.0	8.0	5.0	14.0	11.0	13.0	12.0	9.0	10.0	15.5	15.5	3.0	4.0	2.0	1.0
EOGHorizontalSignal	4.0	1.0	8.0	2.0	10.0	3.0	7.0	5.0	9.0	6.0	13.5	13.5	15.0	16.0	12.0	11.0
EOGVerticalSignal	4.0	1.0	6.0	2.0	10.0	3.0	7.0	5.0	9.0	8.0	13.5	13.5	15.0	16.0	12.0	11.0
Haptics	8.0	1.0	9.0	2.0	10.0	6.0	4.0	3.0	7.0	5.0	11.5	11.5	15.0	16.0	14.0	13.0
Mallat	8.0	2.0	9.0	1.0	10.0	3.0	6.0	4.0	7.0	5.0	13.5	13.5	15.0	16.0	12.0	11.0
Phoneme	11.0	1.0	11.0	2.0	11.0	3.0	11.0	11.0	7.0	6.0	15.5	15.5	11.0	11.0	5.0	4.0
StarLightCurves	8.5	2.0	8.5	1.0	10.0	3.0	5.0	4.0	7.0	6.0	13.5	13.5	15.0	16.0	12.0	11.0
MixedShapesRegularTrain	8.0	2.0	9.0	1.0	12.0	4.0	5.0	3.0	7.0	6.0	13.5	13.5	15.0	16.0	11.0	10.0
MixedShapesSmallTrain	8.0	2.0	9.0	1.0	12.0	4.0	5.0	3.0	7.0	6.0	13.5	13.5	15.0	16.0	11.0	10.0
Average Rank	7.425	1.575	8.725	1.825	10.5	4.95	7.35	6.05	8.3	7.0	14.0	14.0	13.05	13.9	9.05	8.3

Table 4.5: Pruning power comparison on the 20 homogeneous datasets.

Figure 4.6 shows the pruning power scatter plot of our proposed index structure DBCH-tree and the baseline index structure R-tree. We could find that all black points fall above the red colour diagonal line. This situation means the pruning power becomes better in k -NN search when we use DBCH-tree to replace R-tree in all datasets.

Figure 4.7 is a pruning power critical difference diagram for adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA* use DBCH-tree and R-tree in k -NN search. The “*” means the adaptive-length segment dimensionality reduction method uses DBCH-tree in k -NN search. The adaptive-length segment dimensionality reduction method without “*” uses R-tree index structure in k -NN search. The best ranks are to the left. The adaptive-length segment dimensionality reduction methods use DBCH-tree index structure on the left side to have better pruning power than the right side adaptive-length segment dimensionality reduction methods use R-tree index structure. Because we need to compare multiple dimensionality reduction methods in multiple datasets and our proposed adaptive-length segment dimensionality reduction method *SAPLA* uses our proposed DBCH-tree index structure with other dimensionality reduction methods using DBCH-tree or R-tree, the post-hoc analysis is Bonferroni-Dunn Test. The baseline method name is *SAPLA**. The predefined threshold value α is 0.05. The number of datasets is twenty. The pruning power average rank of these adaptive-length segment dimensionality reduction methods is *SAPLA** : 1.35, *APLA** : 1.60, *APCA** : 3.35, *SAPLA* : 4.00, *APLA* : 4.70, *APCA* : 5.70. The critical difference value is 1.5239821521264612. The null-hypothesis is that our proposed adaptive-length segment dimensionality reduction method *SAPLA* uses our proposed DBCH-tree index structure performs equally well in k -NN search with other adaptive-length segment dimensionality reduction methods using DBCH-tree or R-tree index structure and the performance differences are random. If the difference in average ranks between two dimensionality reduction methods is smaller than the critical difference value, their performance difference is not significant (horizontal line). We could find the pruning power of two adaptive-length dimensionality reduction methods *SAPLA* and *APLA* are not significantly different when using DBCH-tree index structure in k -NN search.

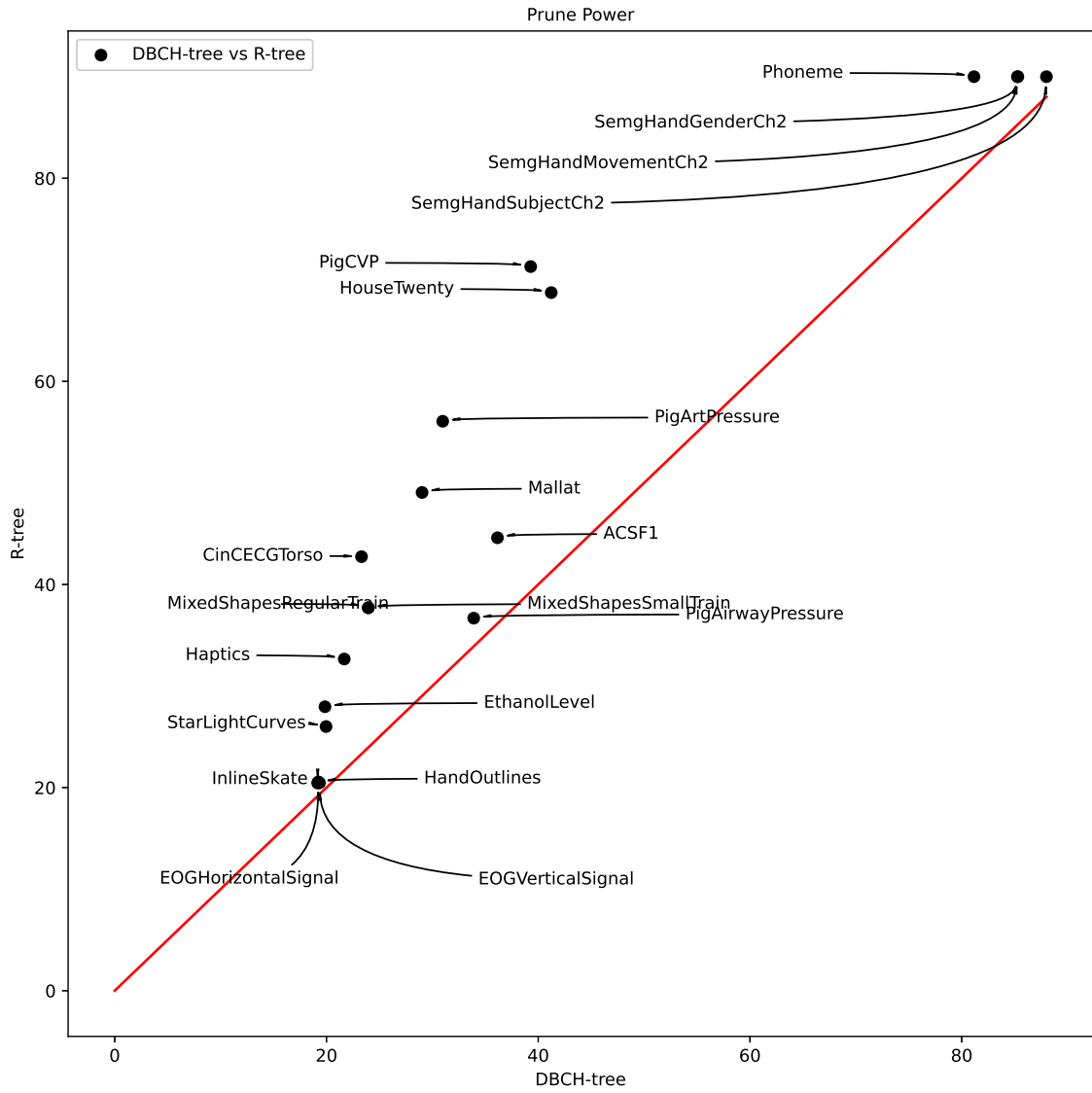


Figure 4.6: Scatter plot of pruning power between DBCH-tree and R-tree on 20 homogeneous datasets. The dimensionality reduction method is *SAPLA*.

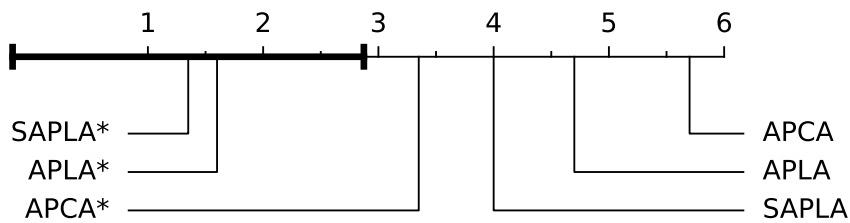


Figure 4.7: Pruning power comparison between DBCH-tree and R-tree for adaptive-length segment dimensionality reduction methods on 20 homogeneous datasets.

We use the Wilcoxon signed ranks test for paired comparison between our proposed index structure DBCH-tree and R-tree in each dimensionality reduction method. The Wilcoxon signed ranks test is a non-parametric statistical hypothesis test. We do not use the t-test, because the pruning power differences of DBCH-tree and R-tree in each dimensionality reduction method are not normal distributions. The null-hypothesis is that the DBCH-tree and R-tree in each dimensionality reduction method perform equally well, and the performance differences are random. We define the predefined threshold value α as 0.05. The p-values of Wilcoxon signed ranks test are (*SAPLA* vs *SAPLA**: $p = 8.844915 \times 10^{-5} < 0.05$), (*APLA* vs *APLA**: $p = 8.832386 \times 10^{-5} < 0.05$), (*APCA* vs *APCA**: $p = 1.960934 \times 10^{-4} < 0.05$), (*PLA* vs *PLA**: $p = 4.36809 \times 10^{-4} < 0.05$), (*PAA* vs *PAA**: $p = 3.589969 \times 10^{-3} < 0.05$), (*PAALM* vs *PAALM**: $p = 4.36809 \times 10^{-4} < 0.05$), (*SAX* vs *SAX**: $p = 8.411029 \times 10^{-3} < 0.05$). we would reject the null hypothesis.

We can conclude that under the threshold 0.05, our proposed adaptive-length segment dimensionality reduction method *SAPLA* has better pruning power than the adaptive-length segment dimensionality reduction method *APLA* in the DBCH-tree index structure and R-tree index structure. The pruning power of our proposed adaptive-length segment dimensionality reduction method *SAPLA* is much better than the adaptive-length segment dimensionality reduction method *APCA*. Three adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA* and three equal-length segment dimensionality reduction methods in the DBCH-tree index structure have better pruning power than in the R-tree index structure.

4.7.3 Comparison on k -NN Time

Table 4.6 and Table 4.7 show the k -NN time of DBCH-tree and R-tree for these eight dimensionality reduction methods. There are 20 homogeneous datasets. The baseline index structure is R-tree. The red color rows are the dimensionality reduction methods that use DBCH-tree to do k -NN search for each dataset, and the white color rows are the dimensionality reduction methods that use R-tree to do k -NN search for each dataset. The index structure has smaller value in Table 4.6 and Table 4.7 means it has fast k -NN search. Table 4.6 and Table 4.7 show *SAPLA* and *APLA* have similar k -NN search time in DBCH-tree and R-tree index structures. The experiments in Section 4.7.2 show DBCH-tree could improve the pruning power of *SAPLA* and *APLA* in k -NN search. Equal-length segment dimensionality reduction methods *PAA*, *PAALM* and *SAX* speed up k -NN search when using DBCH-tree to replace R-tree.

Table 4.8 shows the k -NN search time ranks comparison of each dimensionality reduction

method in DBCH-tree and R-tree index structures for each dataset. The red color columns use DBCH-tree in k -NN search. The white color columns use R-tree in k -NN search. The bottom row is the average rank of each dimensionality reduction method in DBCH-tree and R-tree index structures. We could find DBCH-tree has a strong effect on pruning power of adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA* with similar k -NN search time. DBCH-tree improves the rank of equal-length segment dimensionality reduction methods *PAA*, *PAALM* and *SAX*. The k -NN search time average ranks of these dimensionality reduction methods transfer the index structure from R-tree to DBCH-tree in k -NN search are *SAPLA* (10.8 \rightarrow 11.1), *APLA* (11.25 \rightarrow 10.8), *APCA* (13.55 \rightarrow 8.35), *PLA* (1.0 \rightarrow 2.05), *PAA* (13.2 \rightarrow 3.9), *CHEBY* (7.05 \rightarrow 6.35), *PAALM* (14.3 \rightarrow 5.05), *SAX* (14.1 \rightarrow 3.15). Because *PLA* dimensionality reduction method proposes a novel distance computation equation between the query time series and the internal node of the index structure. DBCH-tree index structure has a slower time than the R-tree index structure. However, DBCH-tree index structure has better pruning power for *PLA* than the R-tree index structure. We could find the k -NN search time of two adaptive-length segment dimensionality reduction methods and four equal-length segment dimensionality reduction methods becoming better when using DBCH-tree to replace R-tree.

Figure 4.8 shows the scatter plot of our proposed index structure DBCH-tree and the baseline index structure R-tree. There are twenty datasets in this scatter plot. Each black color represents a dataset. We could find that seven black points fall below a red colour diagonal line. This situation means the k -NN search time becomes slower in k -NN search when we use DBCH-tree index structure to replace R-tree in seven datasets. However, the DBCH-tree index structure helps to speed up k -NN search time in thirteen datasets.

Figure 4.9 is a k -NN search time critical difference diagram for adaptive-length segment dimensionality reduction methods *SAPLA*, *APLA* and *APCA* use DBCH-tree and R-tree in k -NN search. The “*” means the adaptive-length segment dimensionality reduction method uses DBCH-tree in k -NN search. The adaptive-length segment dimensionality reduction method without “*” uses R-tree index structure in k -NN search. The best ranks are to the left. Because we need to compare multiple dimensionality reduction methods in multiple datasets and compare our proposed adaptive-length segment dimensionality reduction method *SAPLA* uses our proposed DBCH-tree index structure with other dimensionality reduction methods use DBCH-tree or R-tree, the post-hoc analysis is Bonferroni-Dunn Test. The baseline method name is *SAPLA**. The predefined threshold value α is 0.05. The number of datasets is twenty. The k -NN search time average rank of these adaptive-

length segment dimensionality reduction methods is $SAPLA^*$: 3.55, $APLA^*$: 3.25, $APCA^*$: 1.60, $SAPLA$: 3.55, $APLA$: 3.90, $APCA$: 5.15. The critical difference value is 1.5239821521264612. The null-hypothesis is that our proposed adaptive-length segment dimensionality reduction method $SAPLA$ uses our proposed DBCH-tree index structure performs equally well in k -NN search with other adaptive-length segment dimensionality reduction methods using DBCH-tree or R-tree index structure. If the difference of average ranks between two dimensionality reduction methods is smaller than the critical difference value, their performance difference is not significant (horizontal line). We could find two adaptive-length dimensionality reduction methods $SAPLA$ and $APLA$ are not significantly different when using DBCH-tree index structure in k -NN search.

	Index	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM	SAX
HandOutlines	R-tree	0.26	0.27	1.28	0.02	0.42	0.09	0.46	0.47
	DBCH-tree	0.27	0.26	0.06	0.03	0.04	0.08	0.07	0.04
HouseTwenty	R-tree	0.82	0.87	0.98	0.03	0.84	0.11	0.88	0.91
	DBCH-tree	0.31	0.30	0.25	0.05	0.06	0.10	0.07	0.05
PigAirwayPressure	R-tree	0.72	2.97	2.08	0.02	0.69	0.09	0.73	0.72
	DBCH-tree	0.34	0.35	0.14	0.03	0.04	0.08	0.06	0.04
PigArtPressure	R-tree	0.22	0.24	0.33	0.02	0.41	0.09	0.43	0.42
	DBCH-tree	0.32	0.33	0.11	0.04	0.05	0.09	0.06	0.04
PigCVP	R-tree	0.28	0.28	0.40	0.03	0.45	0.09	0.56	2.52
	DBCH-tree	0.33	0.34	0.15	0.04	0.05	0.10	0.07	0.05
InlineSkate	R-tree	0.41	0.41	0.77	0.01	0.70	0.08	0.72	0.73
	DBCH-tree	0.29	0.29	0.11	0.03	0.04	0.07	0.07	0.04
EthanolLevel	R-tree	0.46	0.46	0.57	0.02	0.70	0.08	0.71	0.72
	DBCH-tree	0.15	0.16	0.06	0.03	0.04	0.07	0.07	0.04
CinCECGTorso	R-tree	0.40	0.37	0.52	0.03	3.28	0.10	0.68	0.67
	DBCH-tree	0.25	0.23	0.13	0.04	0.05	0.09	0.07	0.05
SemgHandGenderCh2	R-tree	0.19	0.19	0.21	0.03	0.30	0.10	0.29	0.28
	DBCH-tree	0.34	0.31	0.28	0.06	0.07	0.10	0.07	0.06
SemgHandMovementCh2	R-tree	0.19	0.19	0.20	0.03	0.27	0.10	0.49	0.28
	DBCH-tree	0.35	0.32	0.27	0.05	0.07	0.09	0.07	0.06

Table 4.6: k -NN search time comparison on the first ten homogeneous datasets. The \square is original R-tree. The \blacksquare is DBCH-tree.

	Index	SAPLA	APLA	APCA	PLA	PAA	CHEBY	PAALM	SAX
SemgHandSubjectCh2	R-tree	0.18	0.18	0.18	0.03	0.26	0.10	0.27	0.26
	DBCH-tree	0.35	0.31	0.29	0.06	0.07	0.10	0.07	0.06
ACSF1	R-tree	0.35	0.36	0.44	0.02	0.37	0.08	0.38	0.39
	DBCH-tree	0.29	0.35	0.18	0.05	0.06	0.08	0.05	0.03
EOGHorizontalSignal	R-tree	0.66	1.11	0.90	0.01	0.77	0.08	0.80	0.80
	DBCH-tree	0.23	0.22	0.10	0.03	0.04	0.07	0.07	0.04
EOGVerticalSignal	R-tree	0.67	0.62	0.93	0.01	0.88	0.08	0.84	0.84
	DBCH-tree	0.20	0.20	0.11	0.03	0.03	0.07	0.06	0.03
Haptics	R-tree	0.27	0.31	0.53	0.02	0.62	0.08	0.65	0.66
	DBCH-tree	0.23	0.22	0.07	0.03	0.04	0.07	0.07	0.04
Mallat	R-tree	0.24	0.23	0.34	0.02	0.38	0.09	0.39	0.39
	DBCH-tree	0.19	0.18	0.08	0.04	0.05	0.08	0.07	0.04
Phoneme	R-tree	0.22	0.21	0.25	0.03	0.32	0.10	0.31	0.33
	DBCH-tree	0.34	0.38	0.34	0.06	0.07	0.10	0.06	0.06
StarLightCurves	R-tree	1.52	0.40	0.68	0.01	0.61	0.08	0.65	0.60
	DBCH-tree	0.29	0.28	0.09	0.03	0.04	0.07	0.06	0.03
MixedShapesRegularTrain	R-tree	0.30	0.31	0.59	0.02	0.59	0.09	0.64	0.61
	DBCH-tree	0.29	0.27	0.09	0.03	0.04	0.08	0.07	0.04
MixedShapesSmallTrain	R-tree	0.31	0.33	3.06	0.02	0.59	0.08	0.63	0.61
	DBCH-tree	0.29	0.27	0.09	0.03	0.04	0.08	0.07	0.04

Table 4.7: k -NN search time comparison on the last ten homogeneous datasets. The \square is original R-tree. The \blacksquare is DBCH-tree.

KNN Time (s) Average Rank

	SAPLA	SAPLA*	APLA	APLA*	APCA	APCA*	PLA	PLA*	PAA	PAA*	CHEBY	CHEBY*	PAALM	PAALM*	SAX	SAX*
HandOutlines	9.0	12.0	11.0	10.0	16.0	5.0	1.0	2.0	13.0	3.0	8.0	7.0	14.0	6.0	15.0	4.0
HouseTwenty	11.0	10.0	13.0	9.0	16.0	8.0	1.0	2.0	12.0	4.0	7.0	6.0	14.0	5.0	15.0	3.0
PigAirwayPressure	13.0	9.0	16.0	10.0	15.0	8.0	1.0	2.0	11.0	4.0	7.0	6.0	14.0	5.0	12.0	3.0
PigArtPressure	9.0	11.0	10.0	13.0	12.0	8.0	1.0	2.0	14.0	4.0	6.0	7.0	16.0	5.0	15.0	3.0
PigCVP	10.0	11.0	9.0	12.0	13.0	8.0	1.0	2.0	14.0	4.0	6.0	7.0	15.0	5.0	16.0	3.0
InlineSkate	12.0	9.0	11.0	10.0	16.0	8.0	1.0	2.0	13.0	4.0	7.0	6.0	14.0	5.0	15.0	3.0
EthanollLevel	12.0	9.0	11.0	10.0	13.0	5.0	1.0	2.0	14.0	3.0	8.0	7.0	15.0	6.0	16.0	4.0
CinCEGTorso	12.0	10.0	11.0	9.0	13.0	8.0	1.0	2.0	16.0	4.0	7.0	6.0	15.0	5.0	14.0	3.0
SemgHandGenderCh2	8.0	16.0	9.0	15.0	10.0	11.0	1.0	2.0	14.0	4.0	7.0	6.0	13.0	5.0	12.0	3.0
SemgHandMovementCh2	8.0	15.0	9.0	14.0	10.0	11.0	1.0	2.0	12.0	4.0	7.0	6.0	16.0	5.0	13.0	3.0
SemgHandSubjectCh2	8.0	16.0	10.0	15.0	9.0	14.0	1.0	2.0	12.0	4.0	7.0	6.0	13.0	5.0	11.0	3.0
ACSF1	11.0	9.0	12.0	10.0	16.0	8.0	1.0	3.0	13.0	5.0	6.0	7.0	14.0	4.0	15.0	2.0
EOGHorizontalSignal	11.0	10.0	16.0	9.0	15.0	8.0	1.0	2.0	12.0	3.0	7.0	6.0	13.0	5.0	14.0	4.0
EOGVerticalSignal	12.0	10.0	11.0	9.0	16.0	8.0	1.0	2.0	15.0	4.0	7.0	6.0	13.0	5.0	14.0	3.0
Haptics	11.0	10.0	12.0	9.0	13.0	5.0	1.0	2.0	14.0	3.0	8.0	7.0	15.0	6.0	16.0	4.0
Mallat	12.0	10.0	11.0	9.0	13.0	6.0	1.0	2.0	14.0	4.0	8.0	7.0	16.0	5.0	15.0	3.0
Phoneme	9.0	15.0	8.0	16.0	10.0	14.0	1.0	2.0	12.0	5.0	7.0	6.0	11.0	4.0	13.0	3.0
StarLightCurves	16.0	10.0	11.0	9.0	15.0	8.0	1.0	2.0	13.0	4.0	7.0	6.0	14.0	5.0	12.0	3.0
MixedShapesRegularTrain	11.0	10.0	12.0	9.0	14.0	8.0	1.0	2.0	13.0	4.0	7.0	6.0	16.0	5.0	15.0	3.0
MixedShapesSmallTrain	11.0	10.0	12.0	9.0	16.0	8.0	1.0	2.0	13.0	4.0	7.0	6.0	15.0	5.0	14.0	3.0
Average Rank	10.8	11.1	11.25	10.8	13.55	8.35	1.0	2.05	13.2	3.9	7.05	6.35	14.3	5.05	14.1	3.15

Table 4.8: k -NN time average rank comparison on the 20 homogeneous datasets.

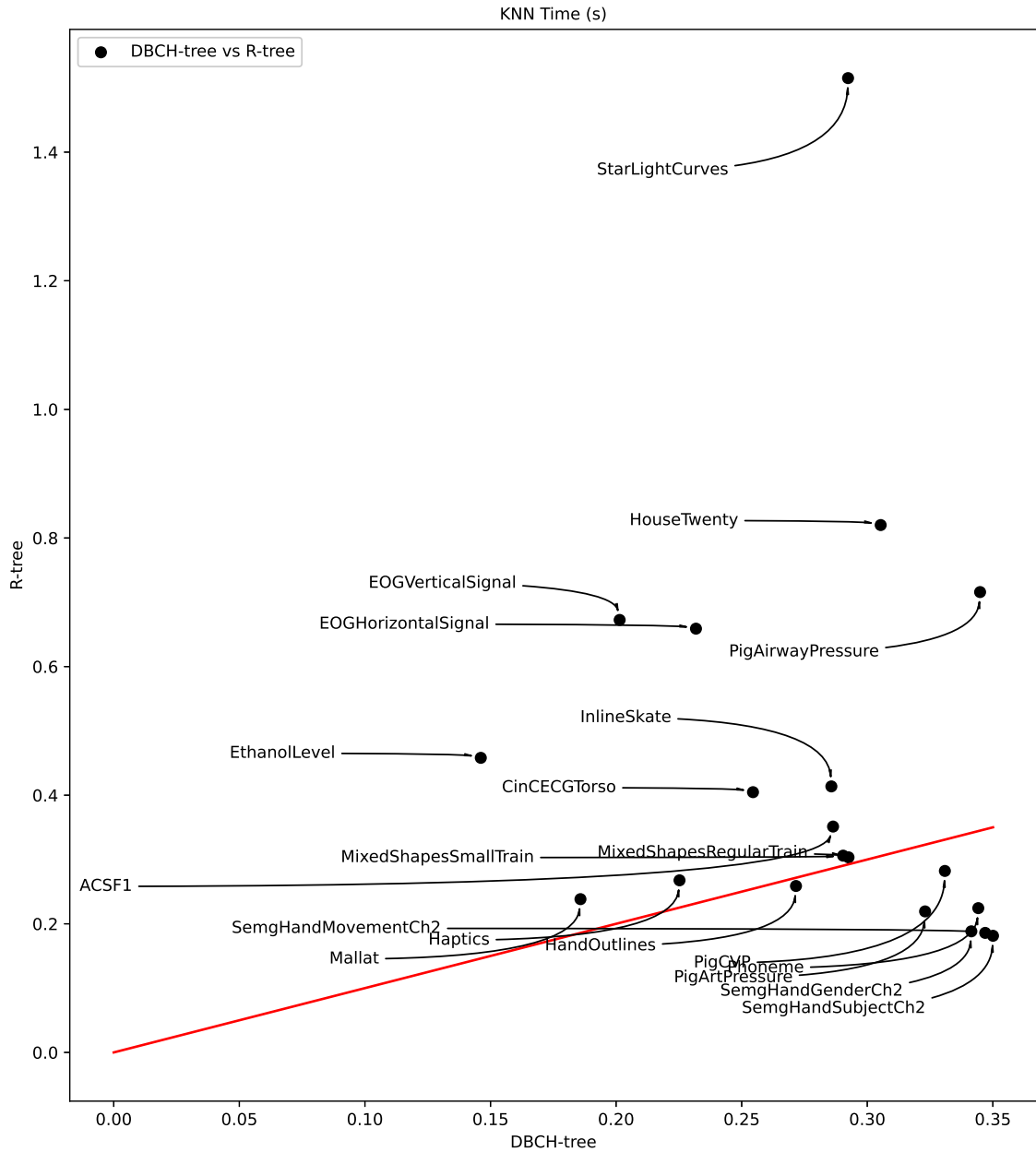


Figure 4.8: Scatter plot of k -NN time between DBCH-tree and R-tree on 20 homogeneous datasets. The dimensionality reduction method is *SAPLA*.

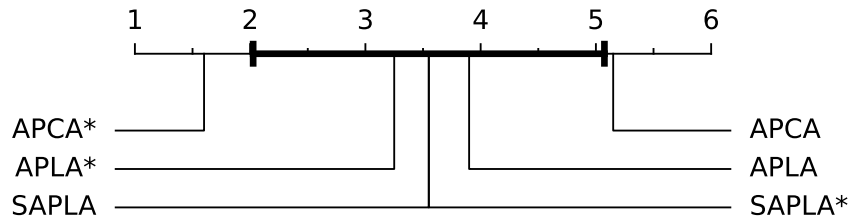


Figure 4.9: k -NN time comparison between DBCH-tree and R-tree for adaptive-length segment dimensionality reduction methods on 20 homogeneous datasets.

We use the Wilcoxon signed ranks test for paired k -NN search time comparison between our proposed index structure DBCH-tree and R-tree in each dimensionality reduction method. The Wilcoxon signed ranks test is a non-parametric statistical hypothesis test. We do not use the t-test because the pruning power differences of DBCH-tree and R-tree in each dimensionality reduction method are not normal distributions. The null-hypothesis is that the DBCH-tree and R-tree in each dimensionality reduction method performs equally well, and the performance differences are random. We define the predefined threshold value α as 0.05. The p-values of Wilcoxon signed ranks test are ($SAPLA$ vs $SAPLA^*$: $p = 1.230927 \times 10^{-1} > 0.05$), ($APLA$ vs $APLA^*$: $p = 1.230927 \times 10^{-1} > 0.05$), ($APCA$ vs $APCA^*$: $p = 8.201599 \times 10^{-5} < 0.05$), (PLA vs PLA^* : $p = 1.907349 \times 10^{-6} < 0.05$), (PAA vs PAA^* : $p = 1.907349 \times 10^{-6} < 0.05$), ($CHEBY$ vs $CHEBY^*$: $p = 6.389618 \times 10^{-3} < 0.05$), ($PAALM$ vs $PAALM^*$: $p = 1.907349 \times 10^{-6} < 0.05$), (SAX vs SAX^* : $p = 1.907349 \times 10^{-6} < 0.05$). we would reject the null hypothesis for six dimensionality reduction methods. The p-values of two adaptive-length segment dimensionality reduction methods are bigger than 0.05. They do not reject the null hypothesis.

Chapter 5

Conclusion

5.1 Summaries

Dimensionality reduction techniques and time series similarity search are widely studied and important field that provide the basis for many higher level data mining tasks. A general framework called Generic Multimedia Indexing Method (*GEMINI*) [23] converts time series into a lower dimensional point. It uses a lower bound of the Euclidean distance to guarantee no-false-dismissals while filtering through the index. Under the *GEMINI* structure, this thesis proposes one adaptive-length segment dimensionality reduction method *SAPLA*, one lower bounding measure $Dist_{PAR}$ for adaptive-length segment dimensionality reduction methods, and a DBCH-tree structure that uses distance based node splitting and branch picking algorithms. For whole sequence similarity matching, R-tree index, DBCH-tree index and k -NN algorithms are implemented. We evaluated several classic dimensionality reduction methods by real datasets.

We implement and analyze the existing basic dimensionality reduction techniques for time series data. For x-axis dimensionality reduction, we analyze the equal-length segment dimensionality reduction method *PLA* [15], *PAA* [39], *CHEBY* and *PAALM* [61]. We also analyze adaptive-length segment dimensionality reduction methods *APCA* [40] and *APLA* [48]. We point out the limitations of the existing dimensionality reduction techniques and evaluate the dimensionality reduction techniques on real-life datasets. We also conduct preliminary research and make some improvements to the existing dimensionality reduction techniques. Our experimental results on several datasets compare and verify the efficiency and effectiveness of the existing techniques in Chapter 2, including several dimensionality reduction techniques, one index building method, and two k -nearest neighbours (k -NN) search methods. *APLA* [48] combines the virtues of *APCA* [40] and *PLA* [15] for the

smaller max deviation. Because *APLA* has the guaranteed error bounds (scan each point to get max deviation) in the reduction process, *APLA* has $O(Nn^2)$ time complexity (n is the original time series length, and N denotes the segment number after the reducing process). Our experiment shows that the dimensionality reduction time of *APLA* is much slower than other methods. We propose *SAPLA* that decreases dimensionality reduction runtime while keeping retrieval performance compared to *APLA*.

We propose an adaptive-length dimensionality reduction method (*SAPLA*) in Section 3.4. *SAPLA* focuses on finding segment endpoints to reduce the sum upper bound of segment max deviation. *SAPLA* prunes redundant computation for dimensionality reduction time reduction ($O(n(N + \log n))$), which is much faster than *APLA* [48] ($O(Nn^2)$) about n times in Section 3.5.3. Our experiment shows *SAPLA* sacrifices little max deviation as presented in Section 3.5.2. *SAPLA* uses a , b of a linear function and adaptive-length segment to represent time series for small max deviation with fewer segments. *SAPLA* is structured into three logical phases (initialisation, split & merge iteration, segment endpoint iteration) and then we analyse its runtime complexity which is significantly lower $O(n * (N + \log n))$ compared to *APLA* time complexity $O(n^2 * N)$. Further more, *SAPLA* is faster than *APLA* about n times with little max deviation loss in our experiment. In the initialisation stage, users set a segment number N . Time series C is initialized into \hat{C} . In the second stage, split & merge iteration tries to reduce the sum upper bound by splitting a segment with the maximum upper bound into two segments and merging two adjacent segments with the minimum reconstruction area. The segment endpoint movement iteration helps to reduce the sum upper bound. Finally, we will get *SAPLA* representation $\hat{C} = \{ \langle a_0, b_0, r_0 \rangle, \dots, \langle a_{N-1}, b_{N-1}, r_{N-1} \rangle \}$.

SAPLA is in comparison with other seven dimensionality reduction methods (*PLA* [15], *APCA* [40], *PLA* [15], *CHEBY* [9], *APLA* [48], *PAALM* [61], *SAX* [60]) through 20 ($n \geq 1024$) real homogeneous datasets [18]. *SAPLA* shows small max deviation, pruning power (ρ), fast dimensionality reduction time and indexing time.

This thesis proposes a lower bounding distance measures $Dist_{PAR}$ for *SAPLA*, *APLA* and *APCA*. $Dist_{PAR}$ has a guaranteed lower bounding lemma, and it is tighter than $Dist_{LB}$. $Dist_{AE}$ has a tight Euclidean distance approximation but non-lower bounding. $Dist_{PAR}$ combines virtues of $Dist_{LB}$ [40] and $Dist_{AE}$ [40]. This thesis also explains how to use adaptive-length dimensionality reduction techniques in R-Tree to retrieve k -NN time series queries efficiently. *MBR* of homogeneous time series could cause serious overlap problem in R-tree. Thus, we propose a DBCH-tree that splits the node and picks the branch by lower bounding distance instead of waste area. DBCH-tree could

efficiently improve indexing in our experiment.

In the experimental evaluation, this thesis compares eight time series dimensionality reduction techniques on 20 datasets from the UCR2018 archive. The dominant use case is similarity search using an R-tree and a DBCH-tree. *SAPLA* shows similar max deviation, pruning power, accuracy and k -NN time compared to *APLA*. However, *SAPLA*'s dimensionality reduction time is faster compared to *APLA*, which accompanies the theoretical runtime complexities.

Our proposed research will bring innovative analysis techniques (such as clustering, matching, filtering and visualization) to high-dimensional spatiotemporal trajectories management, including motif discovery, data characteristics understanding, hypotheses validation, and private data publication. State failure detection will provide an early warning to our future city.

5.2 Limitations and Future Work

This thesis is motivated by proposing an essential dimensionality reduction method for higher-level data mining tasks like classification, motif discovery and anomaly detection. The limitation is that there is no research on higher-level data mining tasks like calculating the matrix profile for motif discovery. The proposed adaptive-length segment dimensionality reduction method applies a conditional upper bound max deviation reduction, not the unconditional upper bound. The distance measure is Euclidean distance, not Dynamic Time Warping (*DTW*) distance. The similarity search is whole sequence matching, not sub-sequence matching. The DBCH-tree constructs the index structure based on distance. If the entry does not consider the distance between each other or cannot get accurate distance measures, DBCH-tree will put dissimilarity entries together and damage the similarity search process.

- The proposed dimensionality reduction technique in this thesis provides a conditional upper bound for max deviation reduction. A guaranteed upper bound for max deviation could help to get minimum max deviation, reconstruction error between original time series and reconstructed time series by representation coefficients and how to achieve global minimum reconstruction error should be investigated further.
- Use Dynamic Time Warping (*DTW*) distance measure instead of Euclidean distance. *DTW* is widely applied in similarity search, clustering, classification, anomaly detection and so on. *DTW* has $O(n^2)$ time complexity.

- Apply higher-level data mining tasks such as classification, motif discovery, sub-sequence matching, or anomaly detection. The sub-sequence matching is a generalization of whole matching. Our proposed adaptive-length segment dimensionality reduction methods and index structure could be extended to sub-sequence matching in the future.
- Propose an extended *SAX* dimensionality reduction method based on the *y*-axis dimension, which can improve the similarity performance and has a small dimensionality reduction time.

Appendices

Appendix A

Detail Experiment Report for Each Dataset

There are 20 homogeneous datasets.

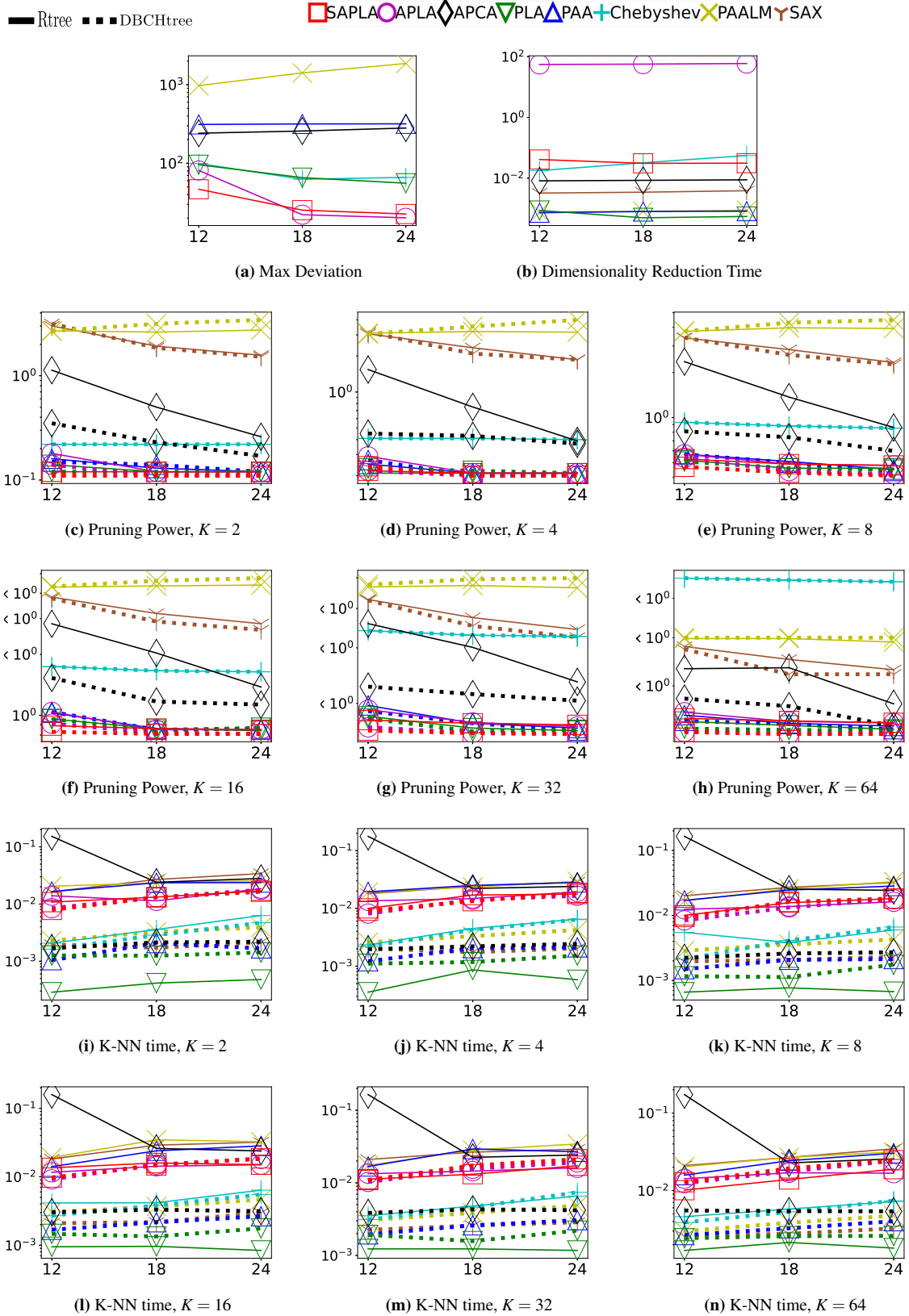


Figure A.1: Dataset 1HO, HandOutlines. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

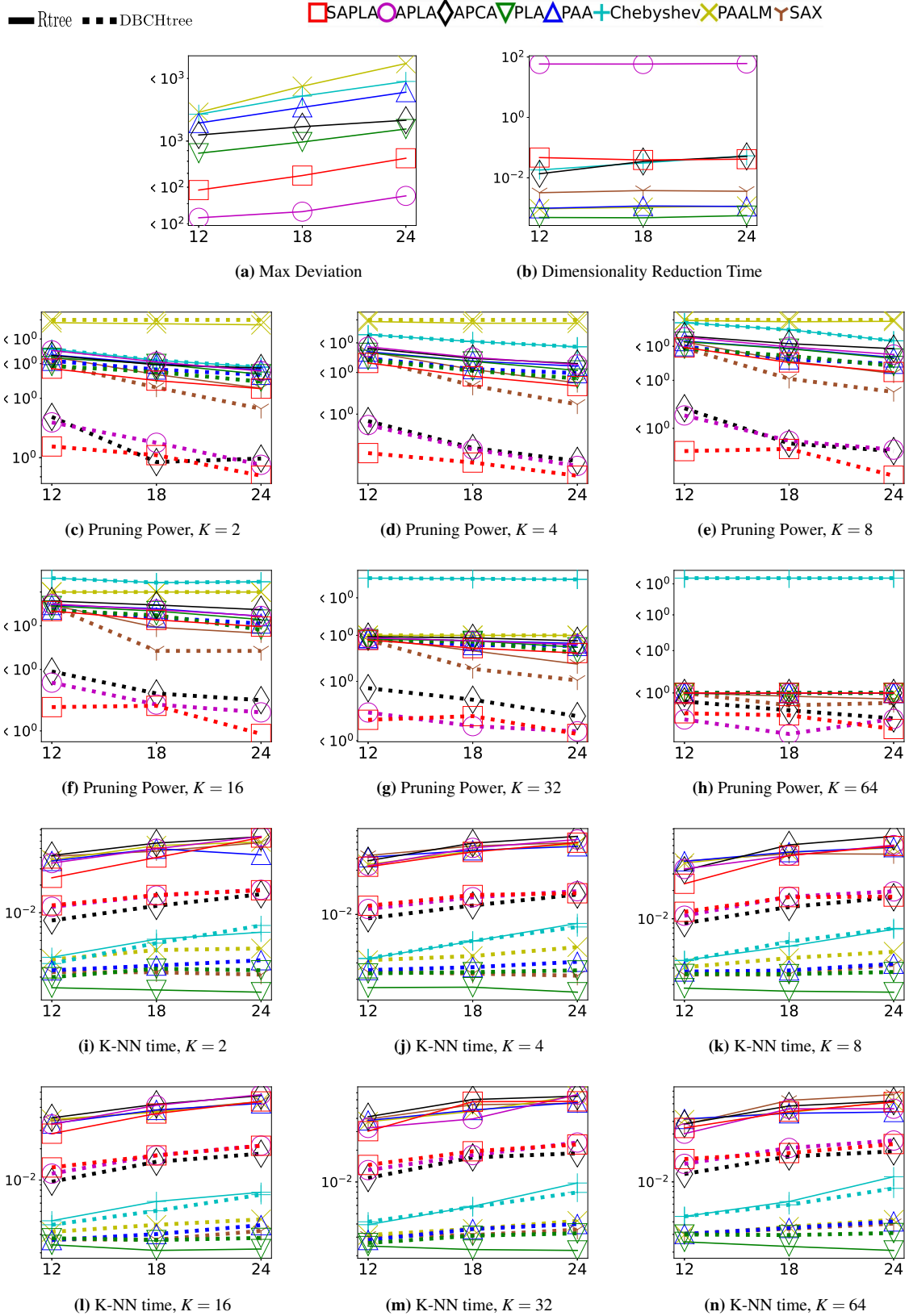


Figure A.2: Dataset 2HT, HouseTwenty. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

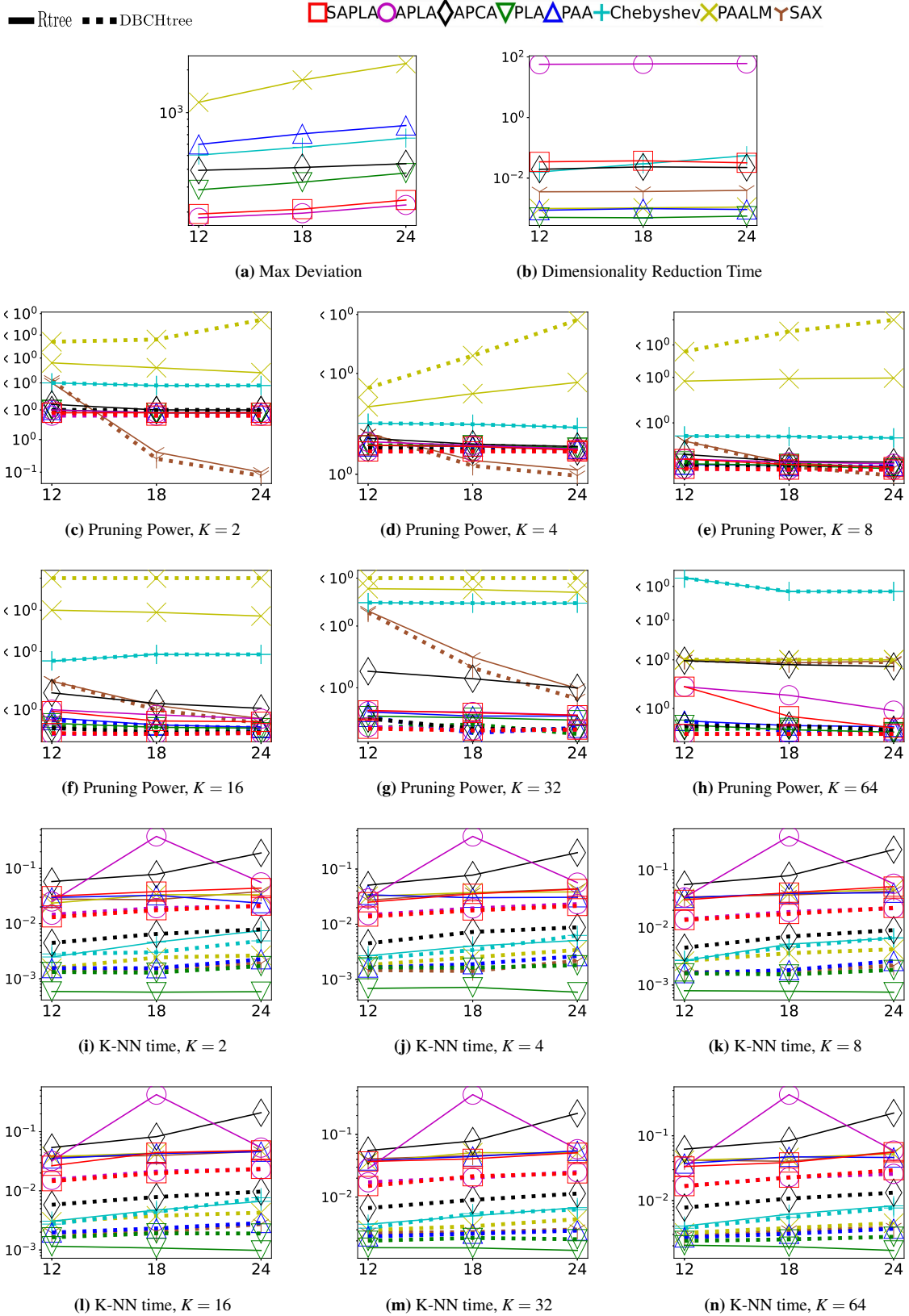


Figure A.3: Dataset 3PAP, PigAirwayPressure. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

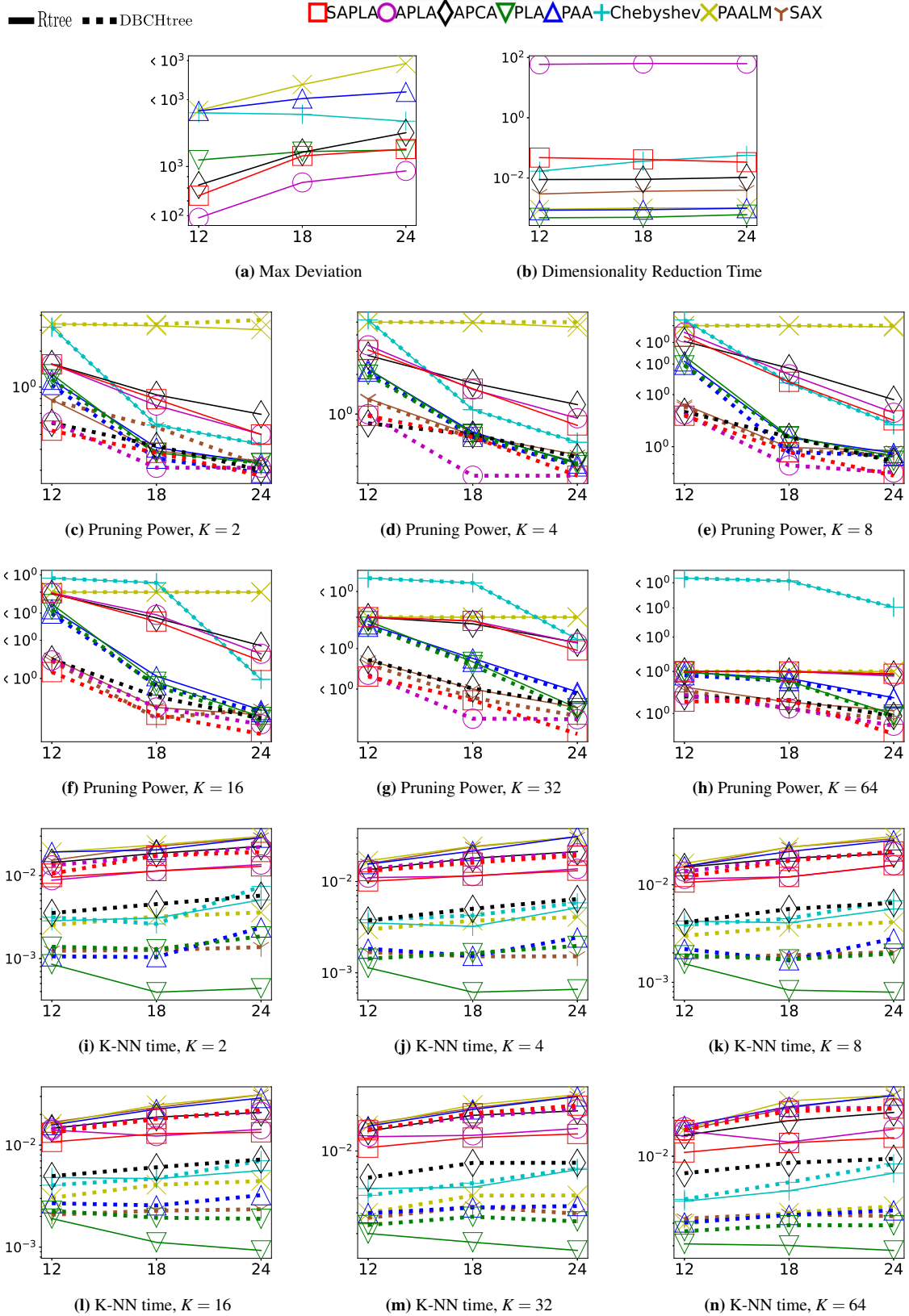


Figure A.4: Dataset 4PAPS, PigArtPressure. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

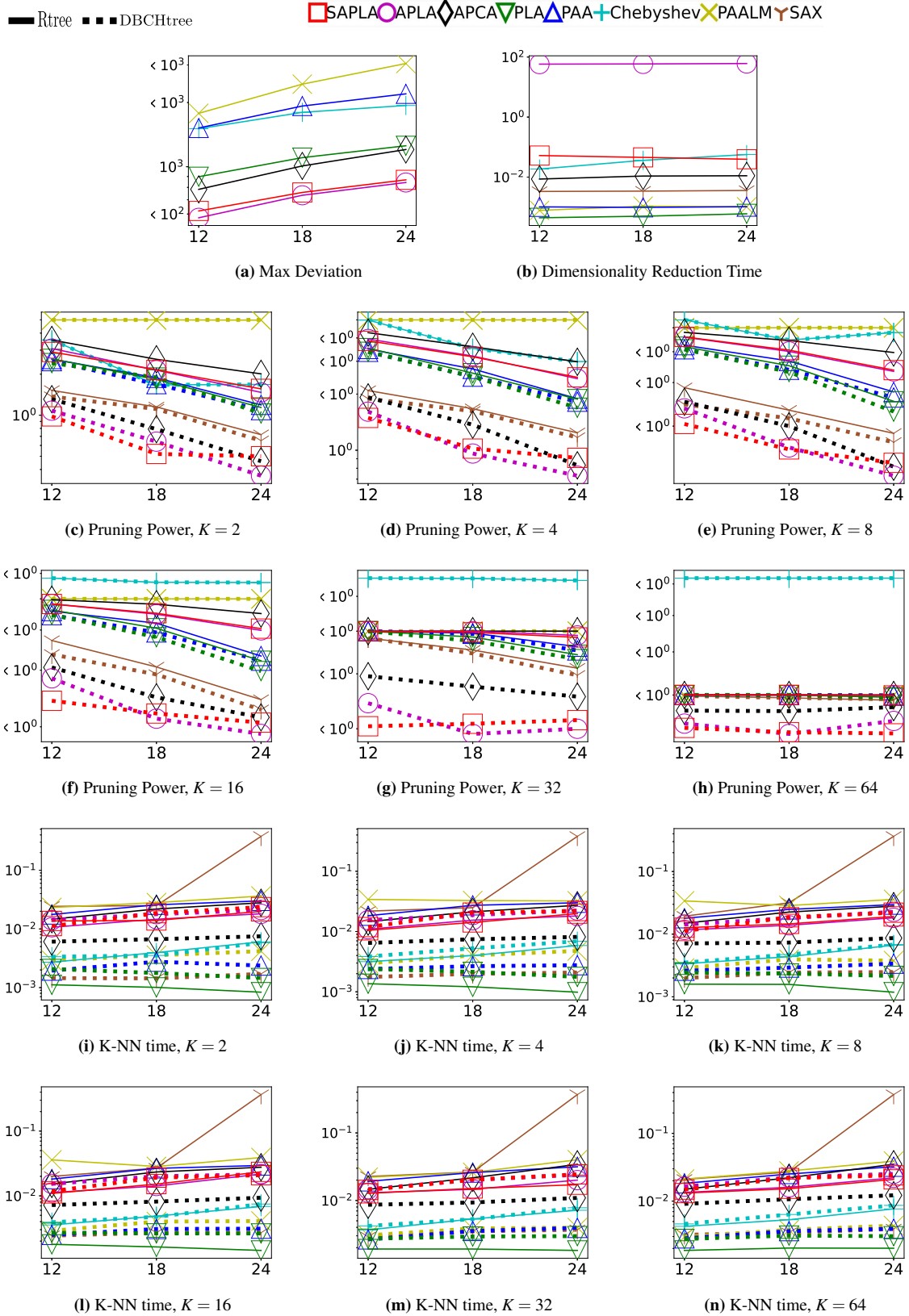


Figure A.5: Dataset 5CVP, PigCVP. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

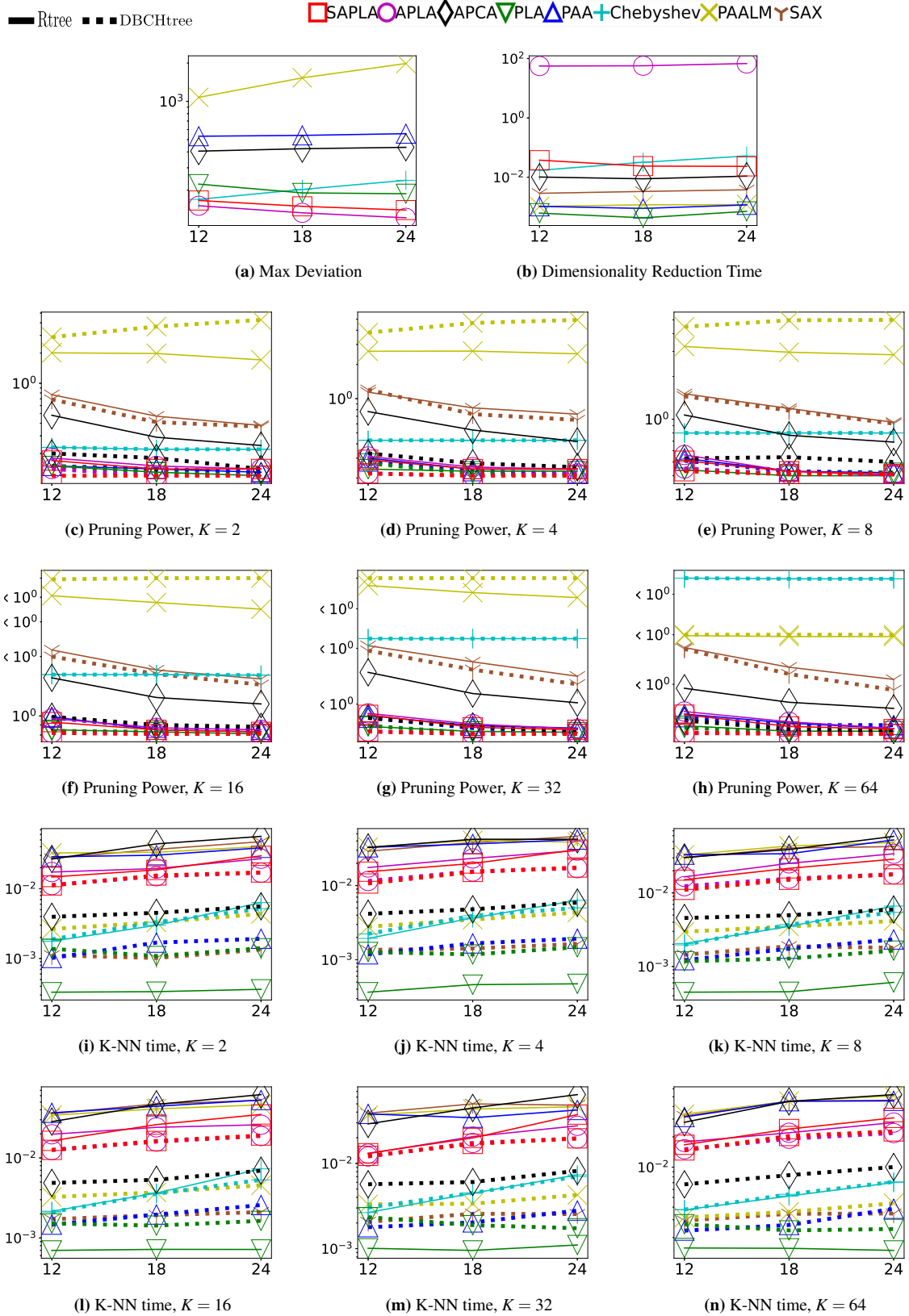


Figure A.6: Dataset *6IS*, *InlineSkate*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

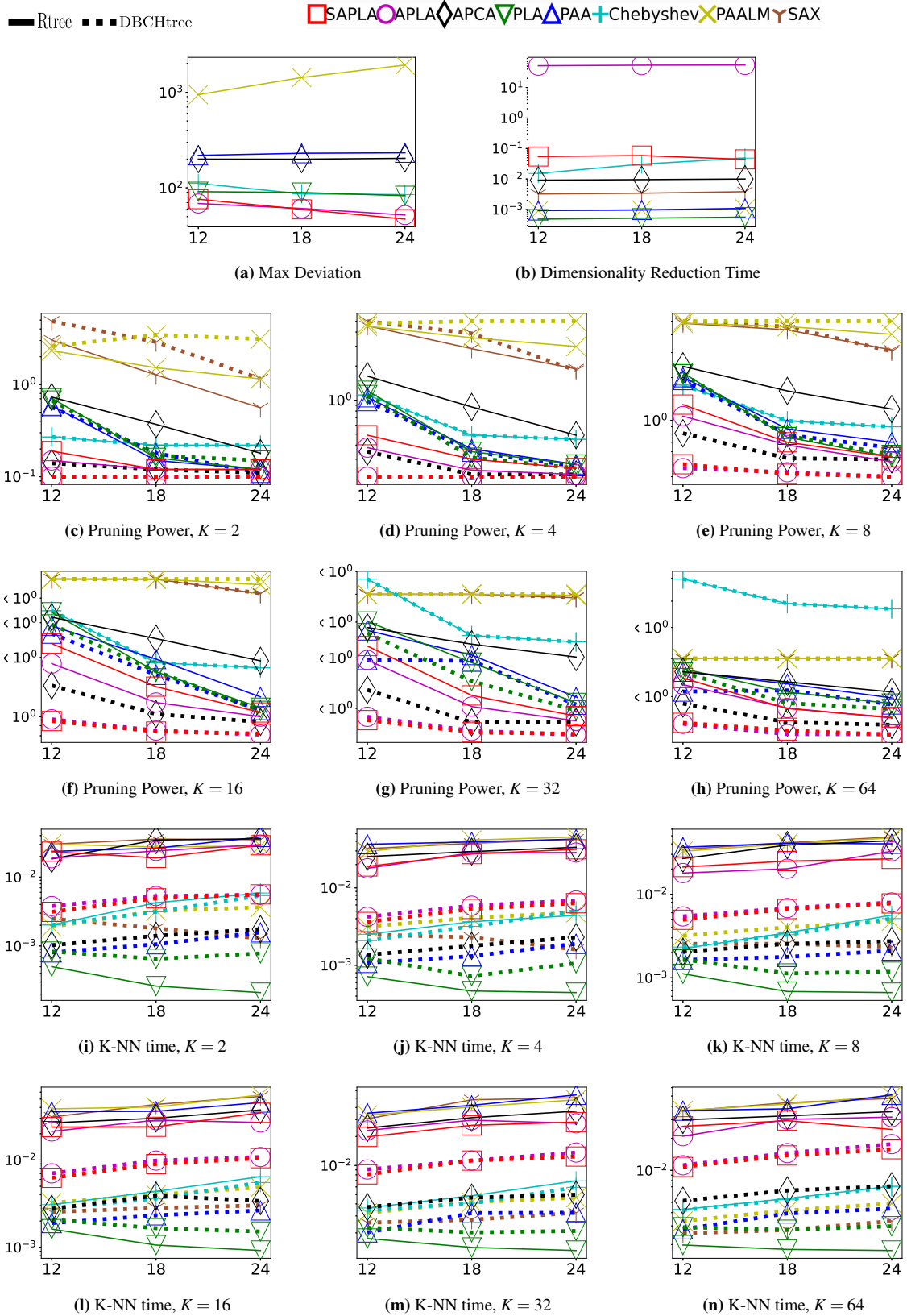


Figure A.7: Dataset 7EL, EthanolLevel. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

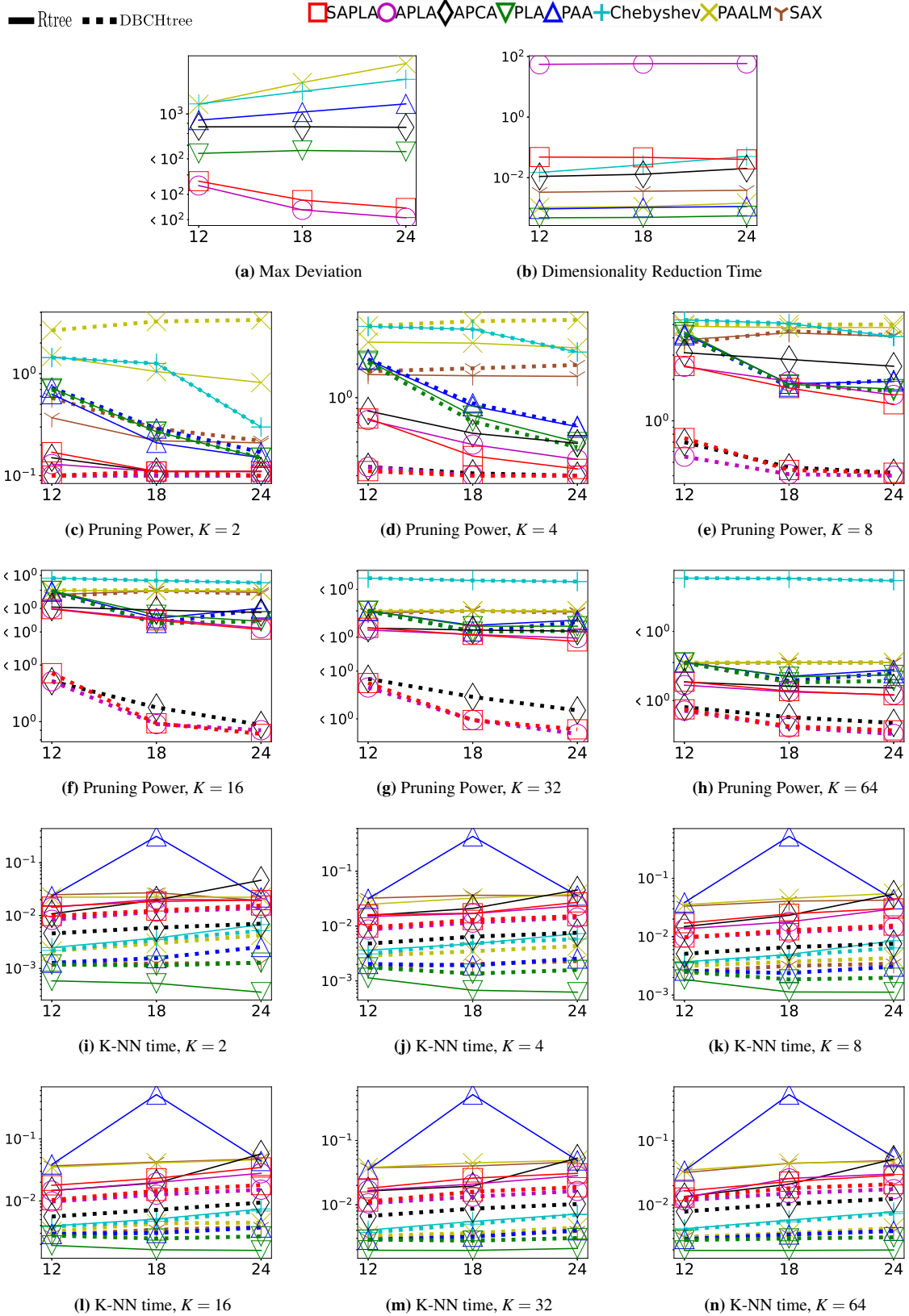


Figure A.8: Dataset 8CT, CinCECGTorso. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

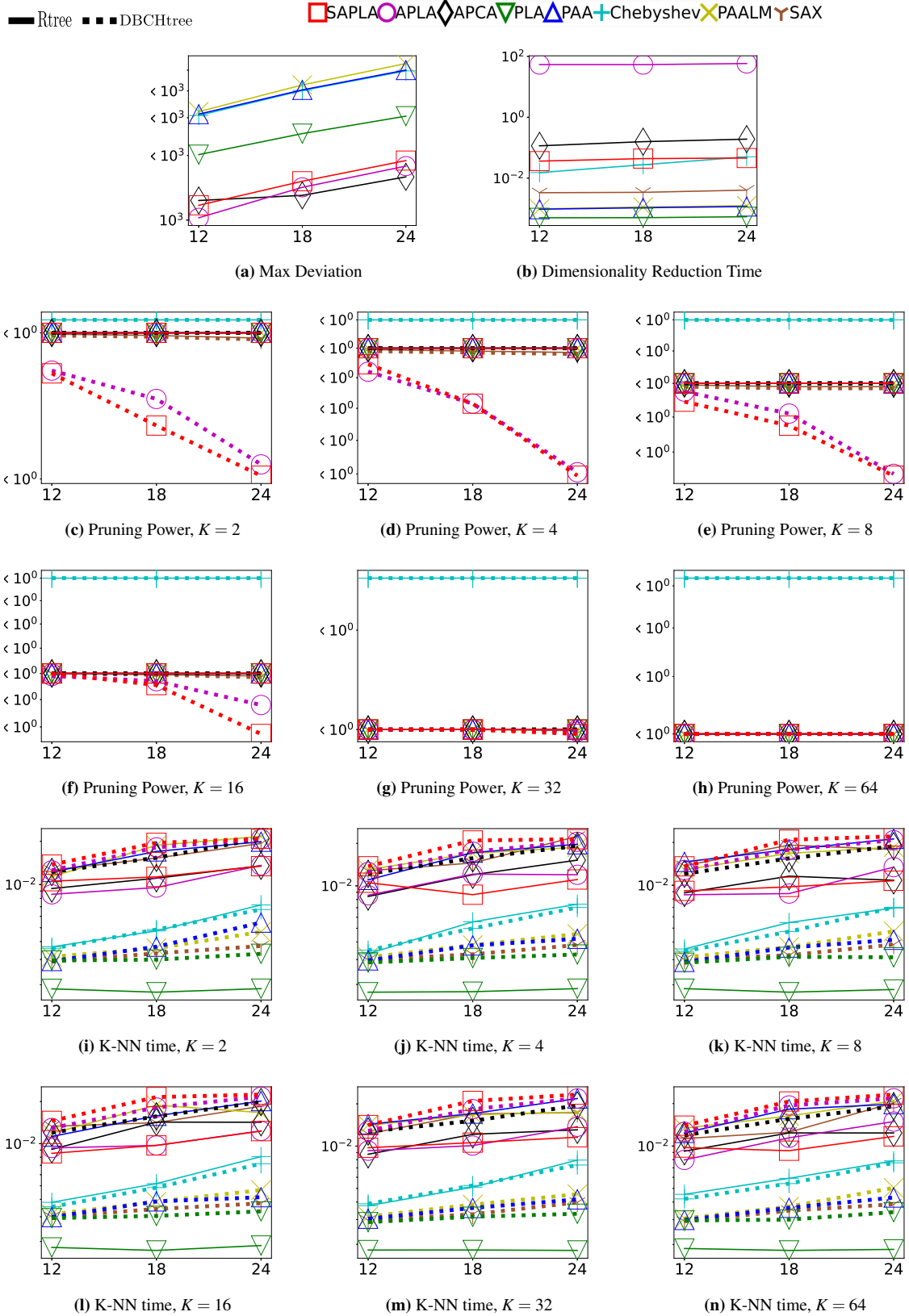


Figure A.9: Dataset 9SHG, SemgHandGenderCh2. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

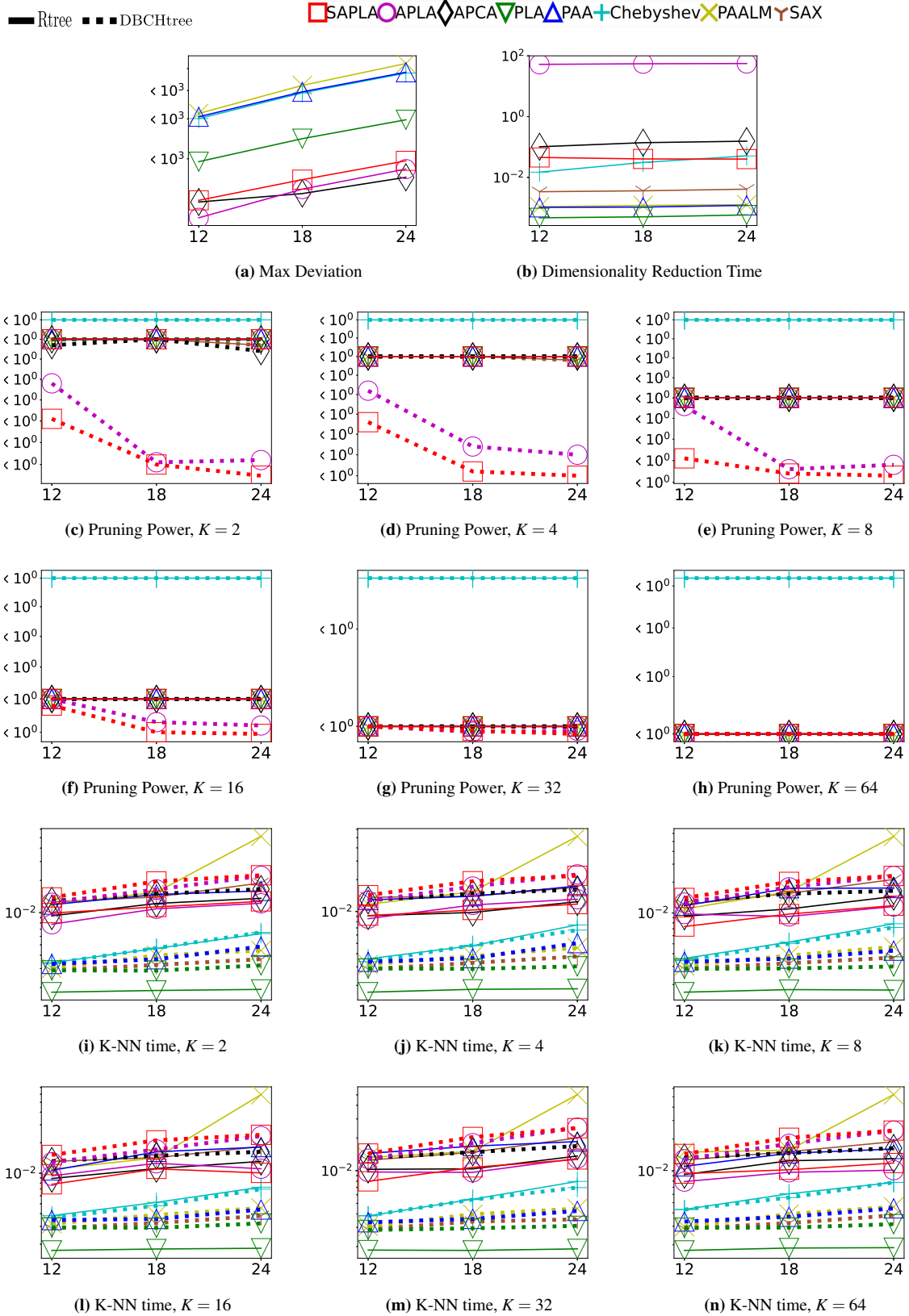


Figure A.10: Dataset 10SHM, *SemgHandMovementCh2*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

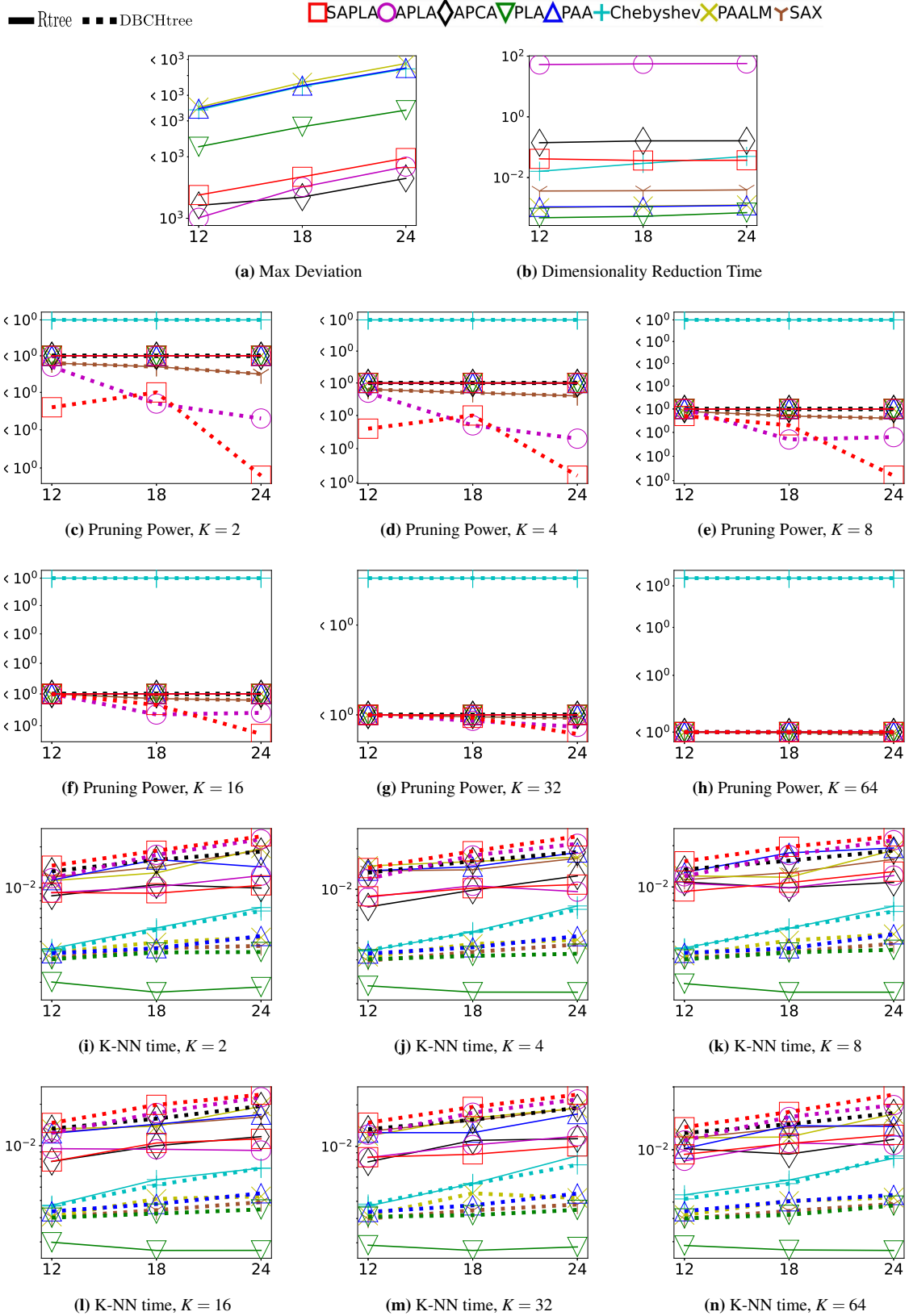


Figure A.11: Dataset 11SHS, *SemgHandSubjectCh2*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

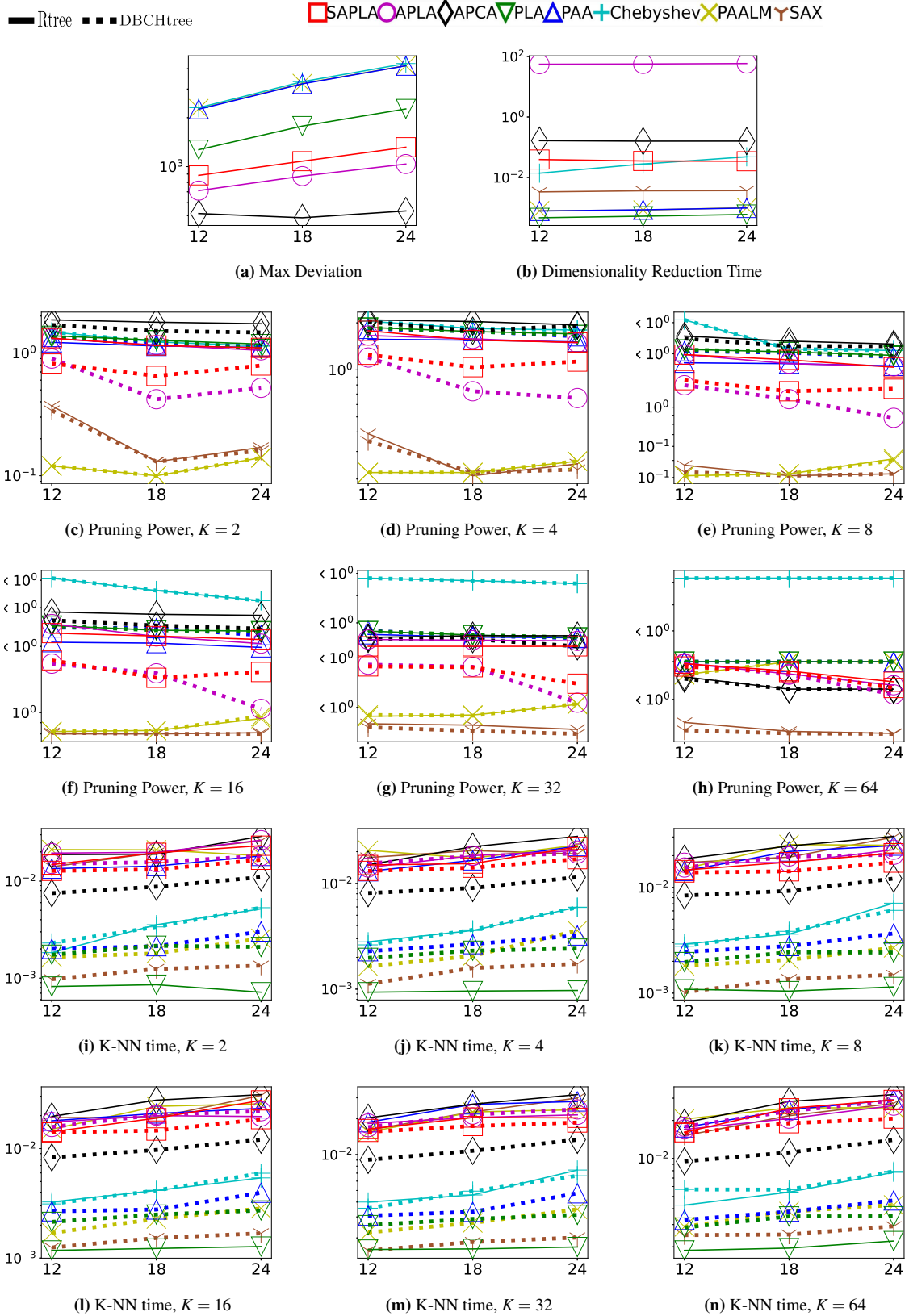


Figure A.12: Dataset 12ACS, ACSF1. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

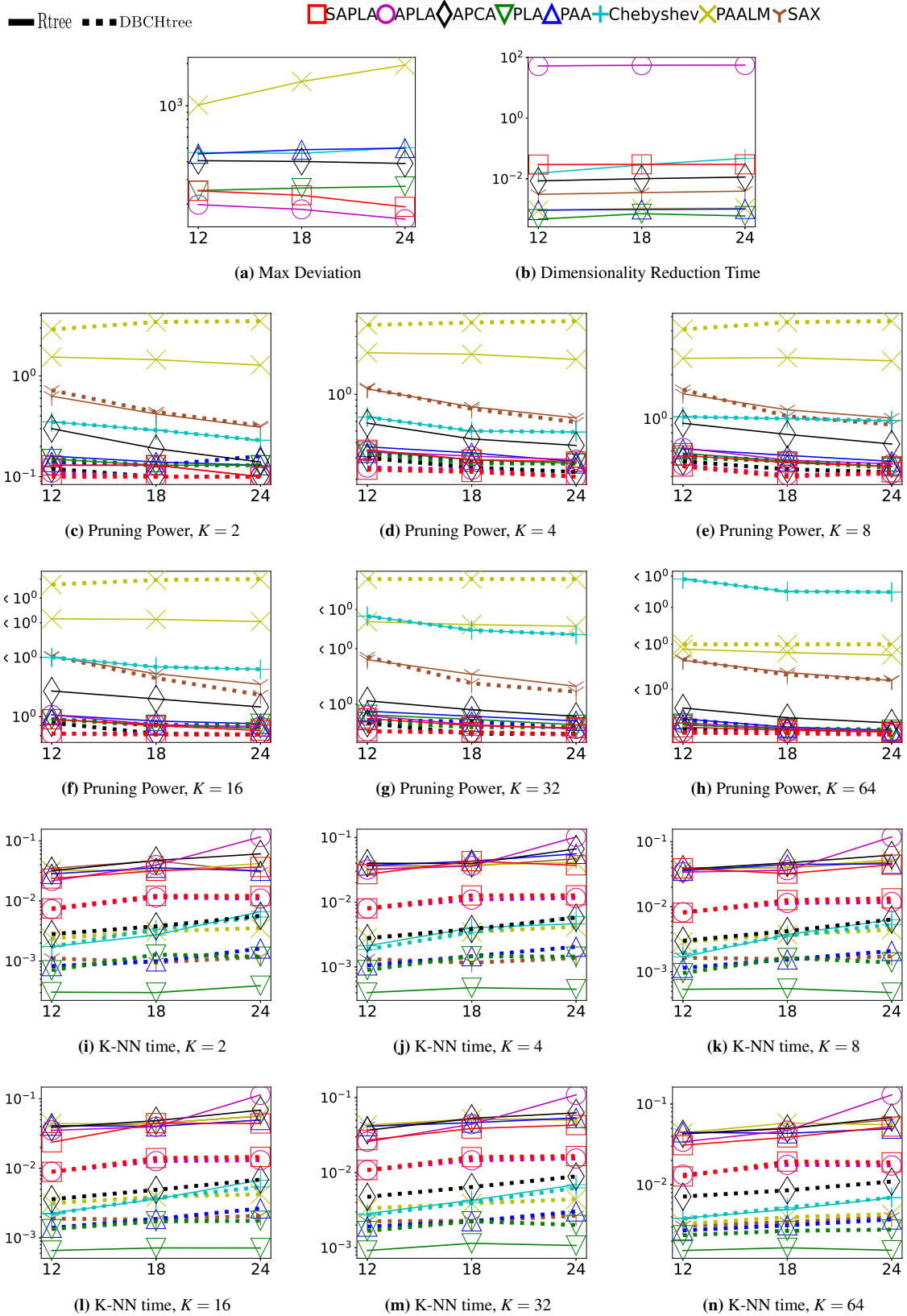


Figure A.13: Dataset 13EHS, EOGHorizontalSignal. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

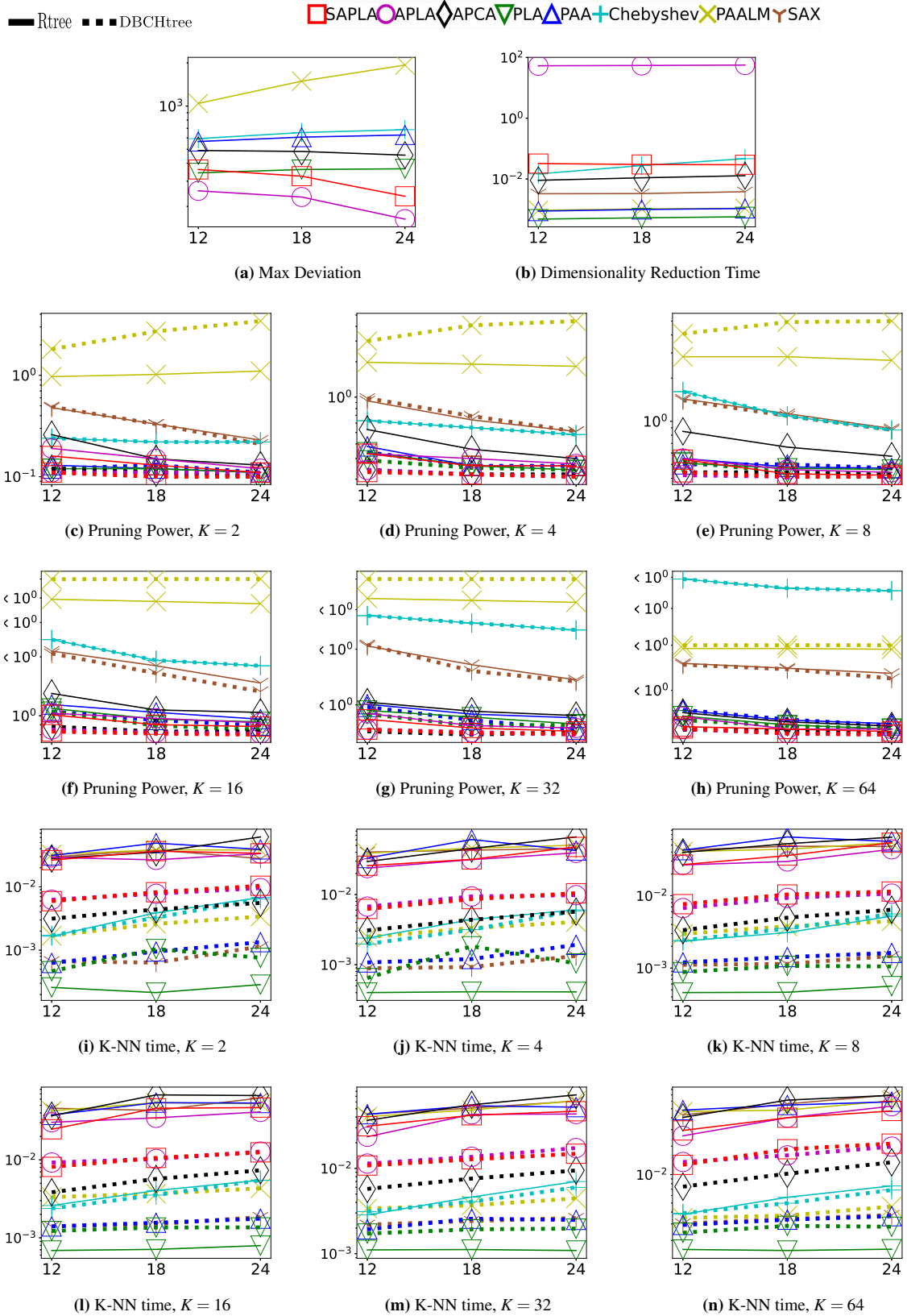


Figure A.14: Dataset 14EVS, *EOGverticalSignal*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

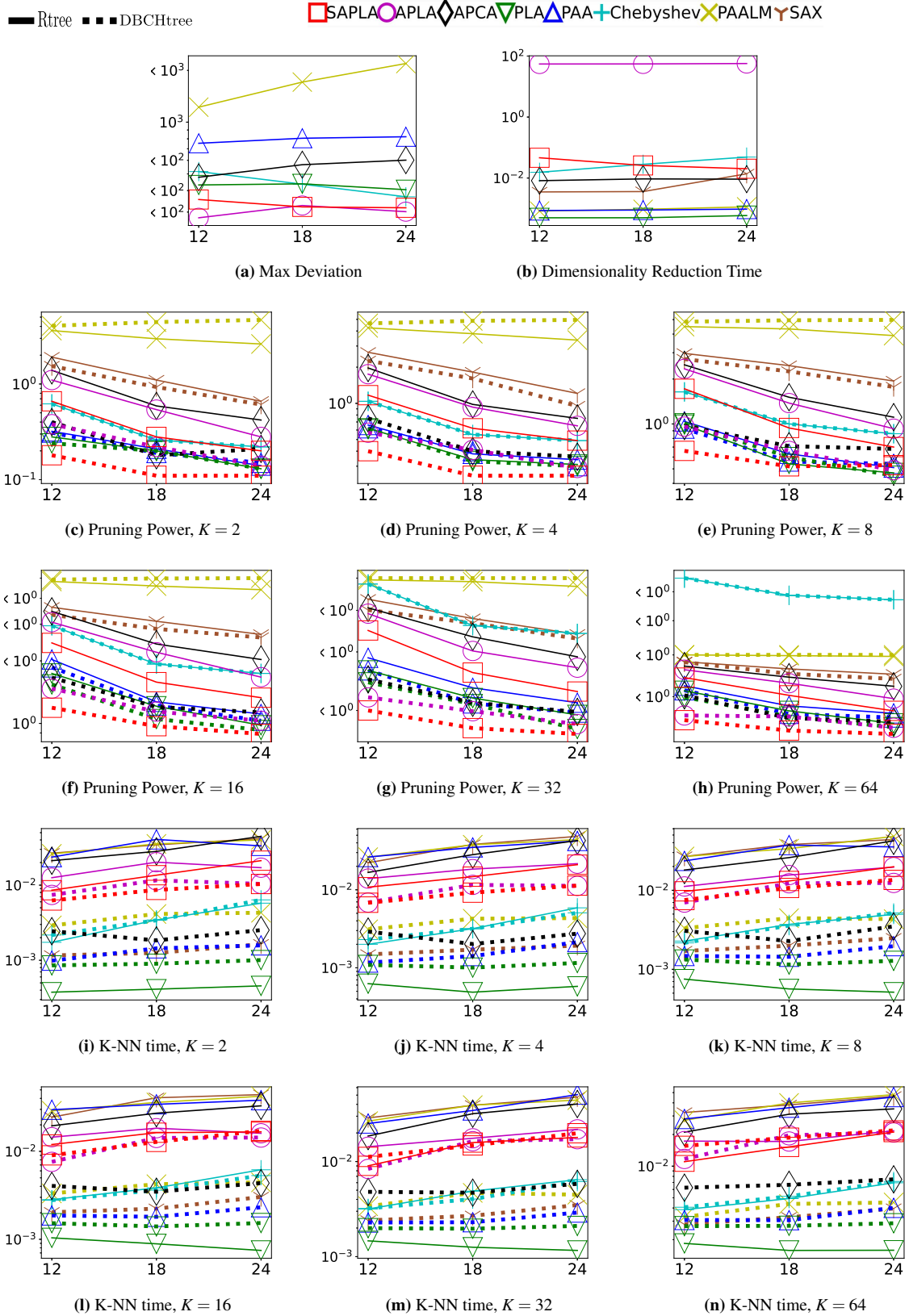


Figure A.15: Dataset 15H, *Haptics*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

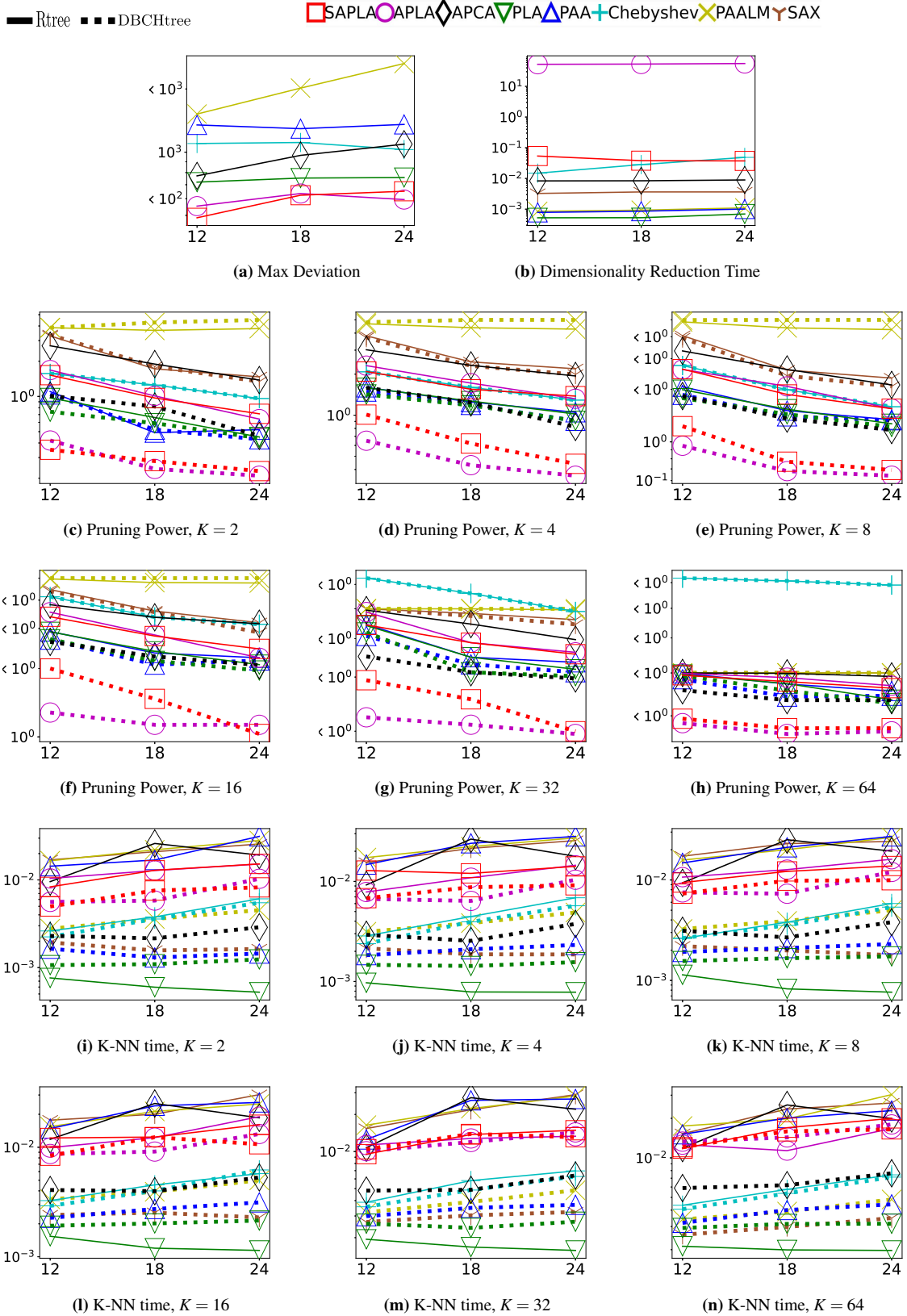


Figure A.16: Dataset 16M, Mallat. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

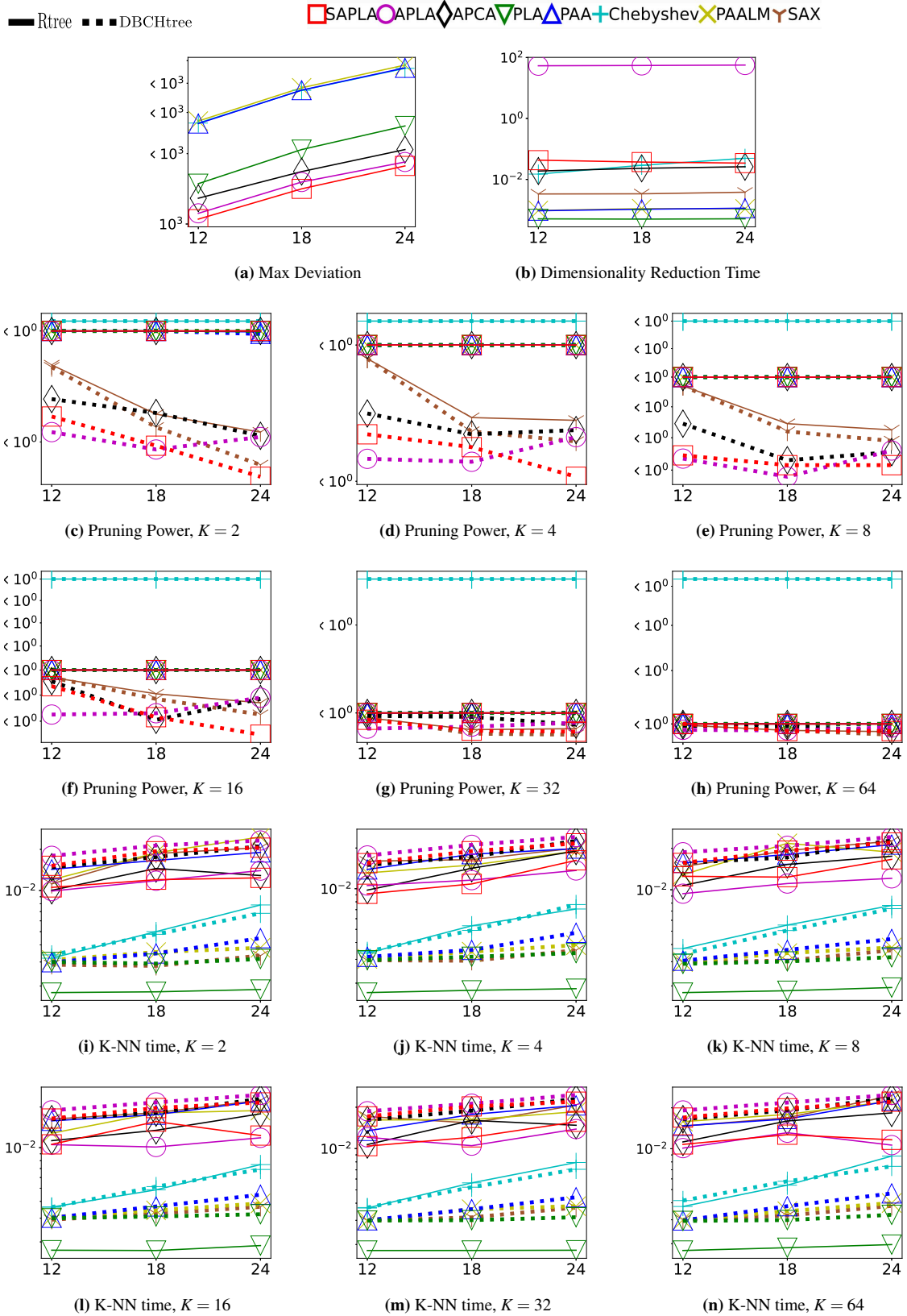


Figure A.17: Dataset 17PM, Phoneme. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

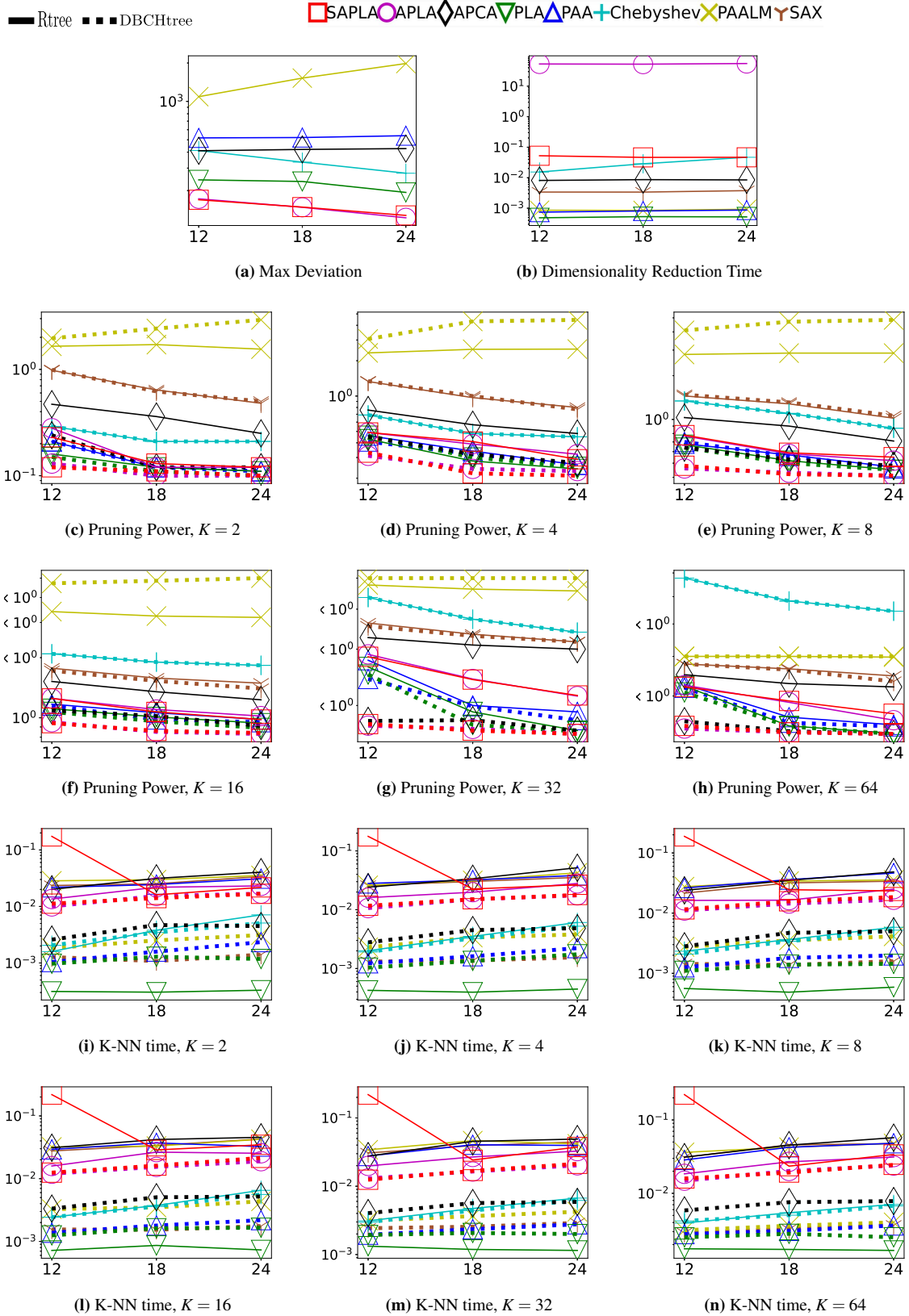


Figure A.18: Dataset 18SLC, *StarLightCurves*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

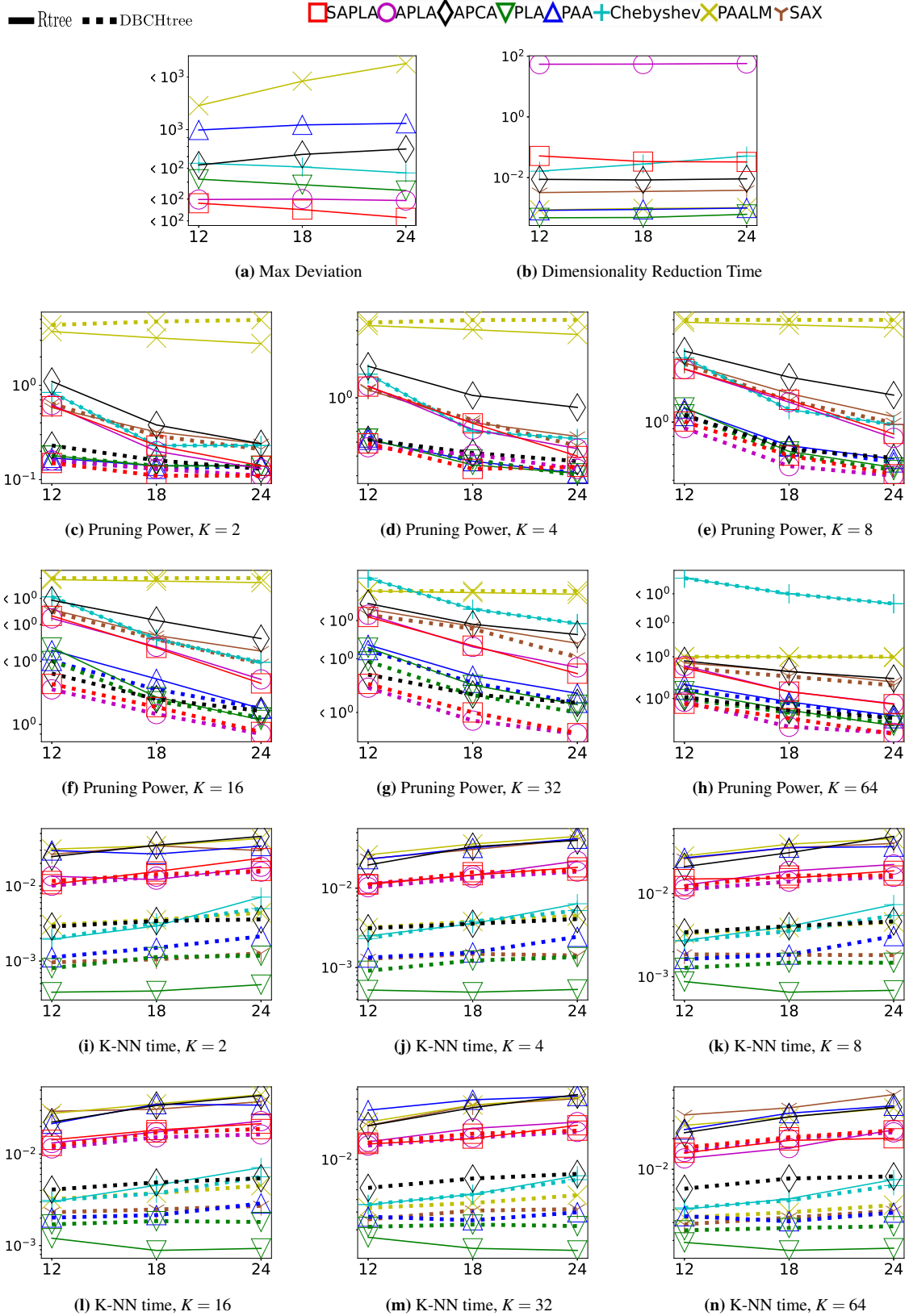


Figure A.19: Dataset 19MSR, *MixedShapesRegularTrain*. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

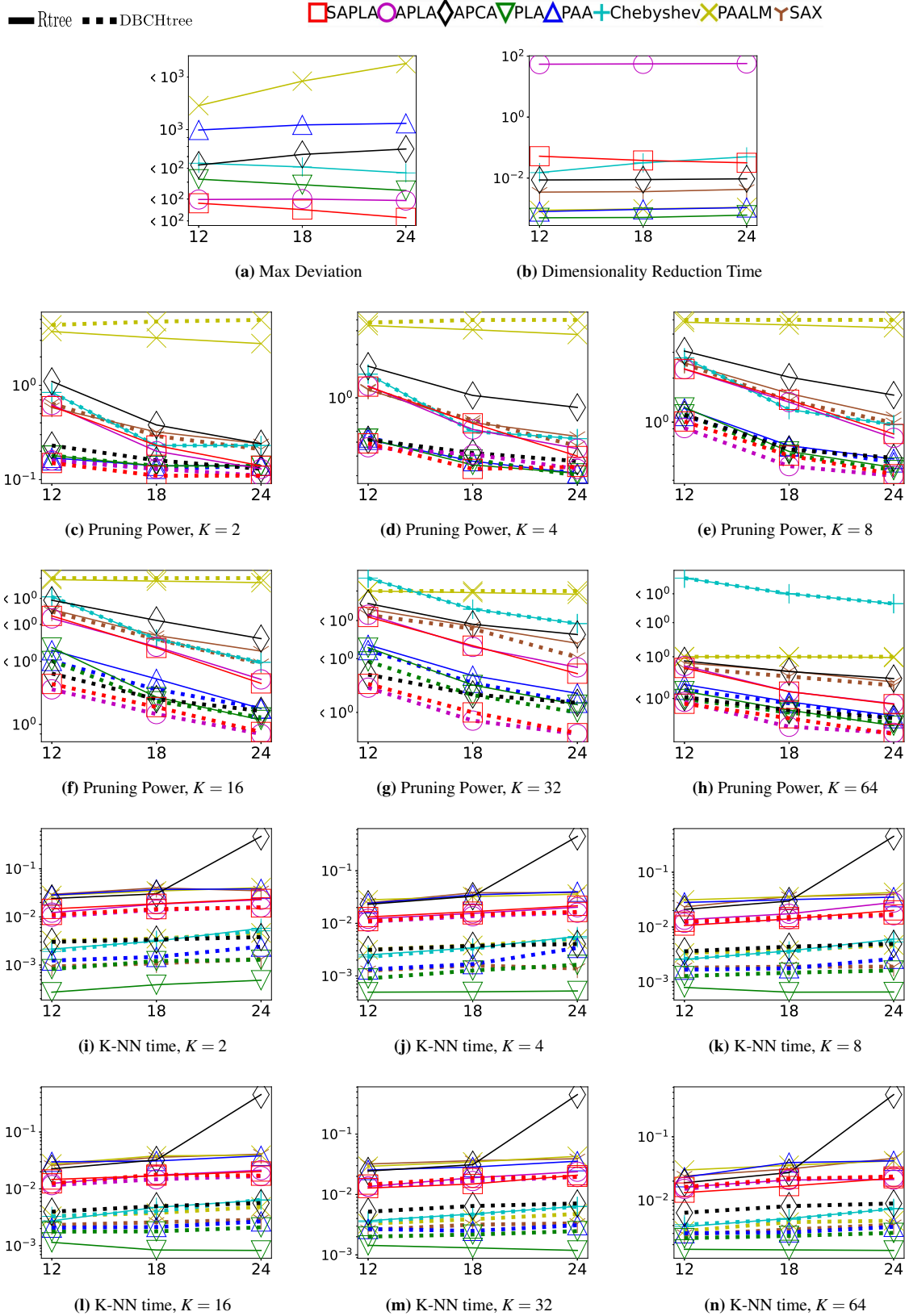


Figure A.20: Dataset 20MST, MixedShapesSmallTrain. $K = 2, 4, 8, 16, 32, 64$. We show the max deviation, dimensionality reduction time, pruning power, K-NN time of each dataset. The solid line represents original R-tree. The dashed line represents DBCH-tree. The horizontal axis is representation coefficients number M .

List of References

- [1] Appendix, source code, full report, datasets, May 2021. URL <https://sites.google.com/view/sapla0/home>.
- [2] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.
- [3] S. Baisch and G. H. Bokelmann. Spectral analysis with incomplete time series: an example from seismology. *Computers & Geosciences*, 25(7):739–750, 1999.
- [4] P. M. Barnaghi, A. A. Bakar, and Z. A. Othman. Enhanced symbolic aggregate approximation method for financial time series data representation. In *2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012)*, pages 790–795. IEEE, 2012.
- [5] V. Bettaiah and H. S. Ranganath. An analysis of time series representation methods: data mining applications perspective. In *Proceedings of the 2014 ACM Southeast Regional Conference*, page 16. ACM, 2014.
- [6] C. Bissell. *Control engineering*. Routledge, 2017.
- [7] B. Bollobás, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 454–456. ACM, 1997.
- [8] P. J. Brockwell and R. A. Davis. *Time series: theory and methods*. Springer Science & Business Media, 2013.
- [9] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2004.

- [10] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2004.
- [11] A. Camera, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh. Beyond one billion time series: indexing and mining very large time series collections with isax2+. *Knowledge and information systems*, 39(1):123–151, 2014.
- [12] N. A. Chadwick, D. A. McMeekin, and T. Tan. Classifying eye and head movement artifacts in eeg signals. In *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, pages 285–291. IEEE, 2011.
- [13] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188–228, 2002.
- [14] K.-P. Chan and W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, page 126. IEEE, 1999.
- [15] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. Indexable pla for efficient similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 435–446. VLDB Endowment, 2007.
- [16] J. Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.
- [17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.
- [18] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
- [19] D. A. Dickey and W. A. Fuller. Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica: Journal of the Econometric Society*, pages 1057–1072, 1981.

- [20] F. Eichinger, P. Efron, S. Karnouskos, and K. Böhm. A time-series compression technique and its application to the smart grid. *The VLDB Journal*, 24(2):193–218, 2015.
- [21] S. Elsworth and S. Güttel. Abba: adaptive brownian bridge-based symbolic aggregation of time series. *Data Mining and Knowledge Discovery*, 34(4):1175–1200, 2020.
- [22] D. Evans and R. Cox. Channel modeling for powerline communications in series-connected pv inverters. In *2018 9th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pages 1–7. IEEE, 2018.
- [23] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.
- [24] C. Faloutsos, H. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 2–20. IEEE, 1997.
- [25] E. D. Feigelson, G. J. BABu, and G. Caceres. Autoregressive times series methods for time domain astronomy. *Frontiers in Physics*, 6:80, 2018.
- [26] P. G. Ferreira, P. J. Azevedo, C. G. Silva, and R. M. Brito. Mining approximate motifs in time series. In *International Conference on Discovery Science*, pages 89–101. Springer, 2006.
- [27] M. Fuad and M. Marwan. Genetic algorithms-based symbolic aggregate approximation. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 105–116. Springer, 2012.
- [28] C. C. Funk, P. J. Peterson, M. F. Landsfeld, D. H. Pedreros, J. P. Verdin, J. D. Rowland, B. E. Romero, G. J. Husak, J. C. Michaelsen, A. P. Verdin, et al. A quasi-global precipitation time series for drought monitoring. *US Geological Survey Data Series*, 832(4), 2014.
- [29] P. Funk and N. Xiong. Extracting knowledge from sensor signals for case-based reasoning with longitudinal time series data. In *Case-Based Reasoning on images and signals*, pages 247–284. Springer, 2008.
- [30] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84,

- pages 47–57, New York, NY, USA, 1984. ACM. ISBN 0-89791-128-8. doi: 10.1145/602259.602266. URL <http://doi.acm.org/10.1145/602259.602266>.
- [31] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.
- [32] W. K. Härdle, N. Hautsch, and L. Overbeck. *Applied quantitative finance*, volume 2. Springer, 2017.
- [33] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, June 1999. ISSN 0362-5915. doi: 10.1145/320248.320255. URL <http://doi.acm.org/10.1145/320248.320255>.
- [34] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265–318, 1999.
- [35] B. Huguency. Adaptive segmentation-based symbolic representations of time series for better modeling and lower bounding distance measures. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 545–552. Springer, 2006.
- [36] M. Ivanović and V. Kurbalija. Time series analysis and possible applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on*, pages 473–479. IEEE, 2016.
- [37] K. Kawagoe and T. Ueda. A similarity search method of time series data with combination of fourier and wavelet transforms. In *null*, page 86. IEEE, 2002.
- [38] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.
- [39] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3): 263–286, 2001.
- [40] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162, 2001.

- [41] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Kdd*, volume 98, pages 239–243, 1998.
- [42] R. S. Koijen, H. Lustig, and S. Van Nieuwerburgh. The cross-section and time series of stock and bond returns. *Journal of Monetary Economics*, 88:50–69, 2017.
- [43] T. C. Krey. Channel waves as a tool of applied geophysics in coal mining. *Geophysics*, 28(5):701–714, 1963.
- [44] Z. Li and Z. Wang. Adaptive cusum of the q chart. *International Journal of Production Research*, 48(5):1287–1301, 2010.
- [45] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.
- [46] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.
- [47] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.
- [48] V. Ljosa and A. K. Singh. Apla: Indexing arbitrary probability distributions. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 946–955. IEEE, 2007.
- [49] B. Lkhagva, Y. Suzuki, and K. Kawagoe. Extended sax: Extension of symbolic aggregate approximation for financial time series data representation. *DEWS2006 4A-i8*, 7, 2006.
- [50] W.-K. Loh, S.-W. Kim, and K.-Y. Whang. Index interpolation: an approach to subsequence matching supporting normalization transform in time-series databases. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 480–487. ACM, 2000.
- [51] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, and Y. Mu. Piecewise linear approximation of streaming time series data with max-error guarantees. In *2015 IEEE 31st International Conference on Data Engineering (ICDE)*,, pages 173–184. IEEE, 2015.
- [52] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

- [53] F. Mörchen, A. Ultsch, and O. Hoos. Extracting interpretable muscle activation patterns with time series knowledge mining. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 9(3):197–208, 2005.
- [54] Y. Morinaka, M. Yoshikawa, T. Amagasa, and S. Uemura. The l-index: An indexing structure for efficient subsequence matching in time sequence databases. In *Proc. 5th PacificAisa Conf. on Knowledge Discovery and Data Mining*, pages 51–60, 2001.
- [55] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *null*, page 338. IEEE, 2004.
- [56] B. Pesaran and M. H. Pesaran. *Time series econometrics using Microfit 5.0: A user's manual*. Oxford University Press, Inc., 2010.
- [57] N. D. Pham, Q. L. Le, and T. K. Dang. Two novel adaptive symbolic representations for similarity search in time series databases. In *2010 12th International Asia-Pacific Web Conference*, pages 181–187. IEEE, 2010.
- [58] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *ICDE*, page 0212. IEEE, 2002.
- [59] R. A. G. Psaila and E. L. Wimmers Mohamed &It. Querying shapes of histories. *Very Large Data Bases. Zurich, Switzerland: IEEE*, 1995.
- [60] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.
- [61] R. Rezvani, P. Barnaghi, and S. Enshaeifar. A new pattern representation method for time-series data. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
- [62] B. Ripley. Pattern recognition and neural networks. *Cambridge University Press Cambridge*, 1996.
- [63] S. Saltenis and C. S. Jensen. Indexing of moving objects for location-based services. In *Proceedings 18th International Conference on Data Engineering*, pages 463–472. IEEE, 2002.

- [64] P. Schäfer and M. Höggqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology*, pages 516–527, 2012.
- [65] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. *ACM Sigmod Record*, 27(2):154–165, 1998.
- [66] J. Shieh and E. Keogh. i sax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2008.
- [67] B. Sjö. Lectures in modern economic time series analysis. *Editia a doua, Universitatea din Linköping, Suedia*, 2011.
- [68] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith. Efficient retrieval for browsing large image databases. In *Proceedings of the fifth international conference on Information and knowledge management*, pages 11–18. ACM, 1996.
- [69] R. Xue, W. Yu, and H. Wang. An indexable time series dimensionality reduction method for maximum deviation reduction and similarity search. In J. Stoyanovich, J. Teubner, P. Guagliardo, M. Nikolic, A. Pieris, J. Mühlig, F. Özcan, S. Schelter, H. V. Jagadish, and M. Zhang, editors, *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, pages 2:183–2:195. OpenProceedings.org, 2022. doi: 10.48786/edbt.2022.08. URL <https://dx.doi.org/10.48786/edbt.2022.08>.
- [70] C. K.-C. Yu and W.-O. Li. A fundamental question about the application of high-density electroencephalography and time-series analysis in examining synchronous networks during sleep—does the use of different referencing and data preprocessing methods really matter? *Sleep and Hypnosis (Online)*, 20(1):67–84, 2018.