Detecting Variable Length Anomaly Patterns in Time Series Data

GVHD: PGS.TS Dương Tuấn Anh

HV: Ngô Duy Khánh Vy

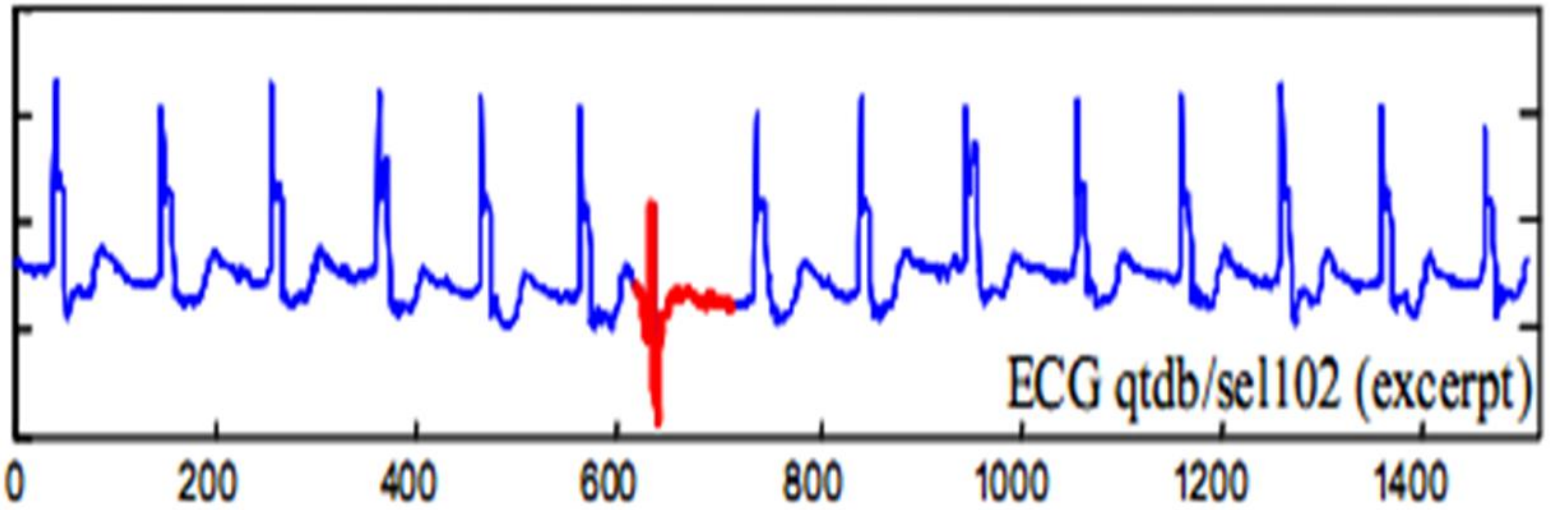
Content

- Introduction
- Some Definitions
- Related Works
- Proposed Algorithms
- Experimental Evaluation
- Conclusion
- Q&A

Introduction

- A time series is a sequence of data points made over a continuous time interval
- Anomaly Patterns are subsequences that do not conform to a well defined notion of normal behavior
- Application areas that explore such time series anomalies include fault diagnostics, intrusion detection, fraud detection, auditing and data cleansing

Introduction



unusual subsequence in time series data

Introduction

- Anomalous subsequences detection is a challenging topic
- The exact lengths of the unusual subsequence are vary and are often unknown.
- Best similarity/distance measures which can be used for different types of time series is not easy to determine.
- Time series in real applications are usually long and as the length increases the computational complexity also increases.

Some Definitions

- *Time Series*: A time series $T = t_1, t_2, \ldots, t_m$ is an ordered set of m real values measured at equal intervals.
- *Subsequence*: Given a time series T of length m, a subsequence C is a sampling of length $n < m$ of contiguous positions from T, i.e., $C = t_p, t_{p+1}, \ldots, t_{p+n-1}$, for $1 \leq p \leq m-n+1$. Sometimes, C is denoted as (s_p, e_{p+n-1}), where $s_p = t_p$ and $e_{p+n-1} = t_{p+n-1}$.

Some Definitions

- *Distance function*: *Dist*(*C, M*) is a positive value used to measure the difference between two time series *C* and *M*, based on some measure method
- *k-distance of a pattern*: Given a positive integer *k*, a pattern set *D* and a pattern $P \in D$, the *k*-distance of *P*, denoted as *k-dist*(*P*), is defined as the distance between *P* and a pattern $Q \in D$ such that.

  i) For at least *k* patterns $Q' \in D$ it holds that *Dist*(*D, Q'*) $\leq$ *Dist*(*D, Q*).

  ii) For at most *k-1* patterns $Q' \in D \setminus \{Q\}$ it holds that *Dist*(*D, Q'*) < *Dist*(*D, Q*).

Some Definitions

- *Non-Self Match*: Given a time series *T*, its two subsequences *P* of length *n* starting at position *p* and *Q* starting at position *q*, we say that *Q* is a non-self-match to *P*, if $Dist(P, Q) \geq e$ or $|p - q| \geq n$, where *e* is a given value of distance threshold.
- *Anomaly factor*: For any pattern set *D* and a pattern $P \in D$, *k-dist*(*D*) denotes all *k-dist* of patterns, anomaly factor of pattern *P* defined as the ratio of *k-dist*(*P*) to *median*(*k-dist*(*D*)).
- *Anomaly Pattern*: Given any pattern set *D*, a pattern $P \in D$, *P* is anomaly only if its anomaly factor is larger than *a*, where *a* is the threshold of anomaly pattern

Related Works

- Hot Sax (E. Keogh et al. 2005).
- WAT (Y. Bu et al. 2007).
- Method based on PAA bit representation and clustering (Li et al. 2013)
- Method based on Segmentation and Anomaly Factors (M.Leng et al. 2008)

HOT SAX

- Find discord of length *n.*
- A discord is a subsequence which have largest nearest neighbor distance.
- Use heuristic to improve brute-force.
- Distance function: Euclid distance.

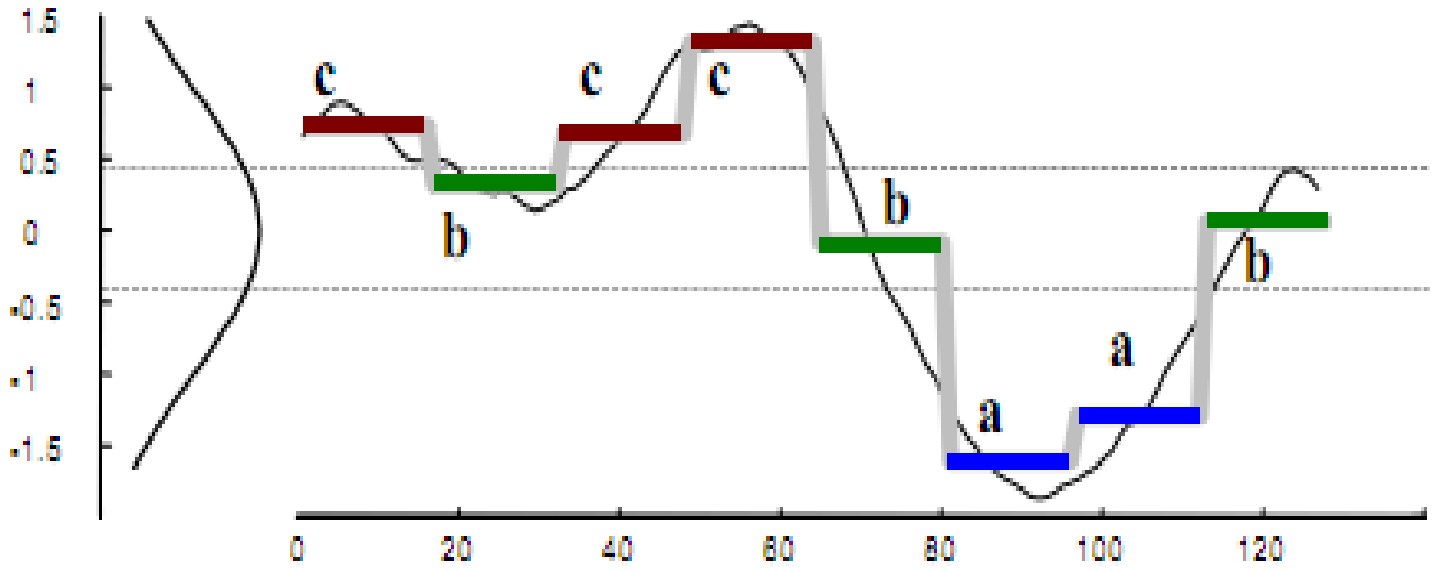
HOT SAX

```
Function [dist, loc ]= Heuristic_Search(T, n, Outer, Inner )
 best_so_far_dist = 0
 best_so_far_loc = NaN
For Each p  in T ordered by heuristic Outer      // Begin Outer Loop
    nearest_neighbor_dist = infinity
    For Each q  in T ordered by heuristic Inner  // Begin Inner Loop
        IF | p – q | ≥ n                          // non-self match?
            IF  Dist ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)  < best_so_far_dist
                Break                             // Break out of Inner Loop
            End
            IF  Dist ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)  < nearest_neighbor_dist
                nearest_neighbor_dist = Dist ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)
            End
        End                                       // End non-self match test
    End                                           // End Inner Loop
     IF nearest_neighbor_dist > best_so_far_dist
        best_so_far_dist = nearest_neighbor_dist
        best_so_far_loc  = p
     End
End                                               // End Outer Loop
Return[ best_so_far_dist, best_so_far_loc ]
```

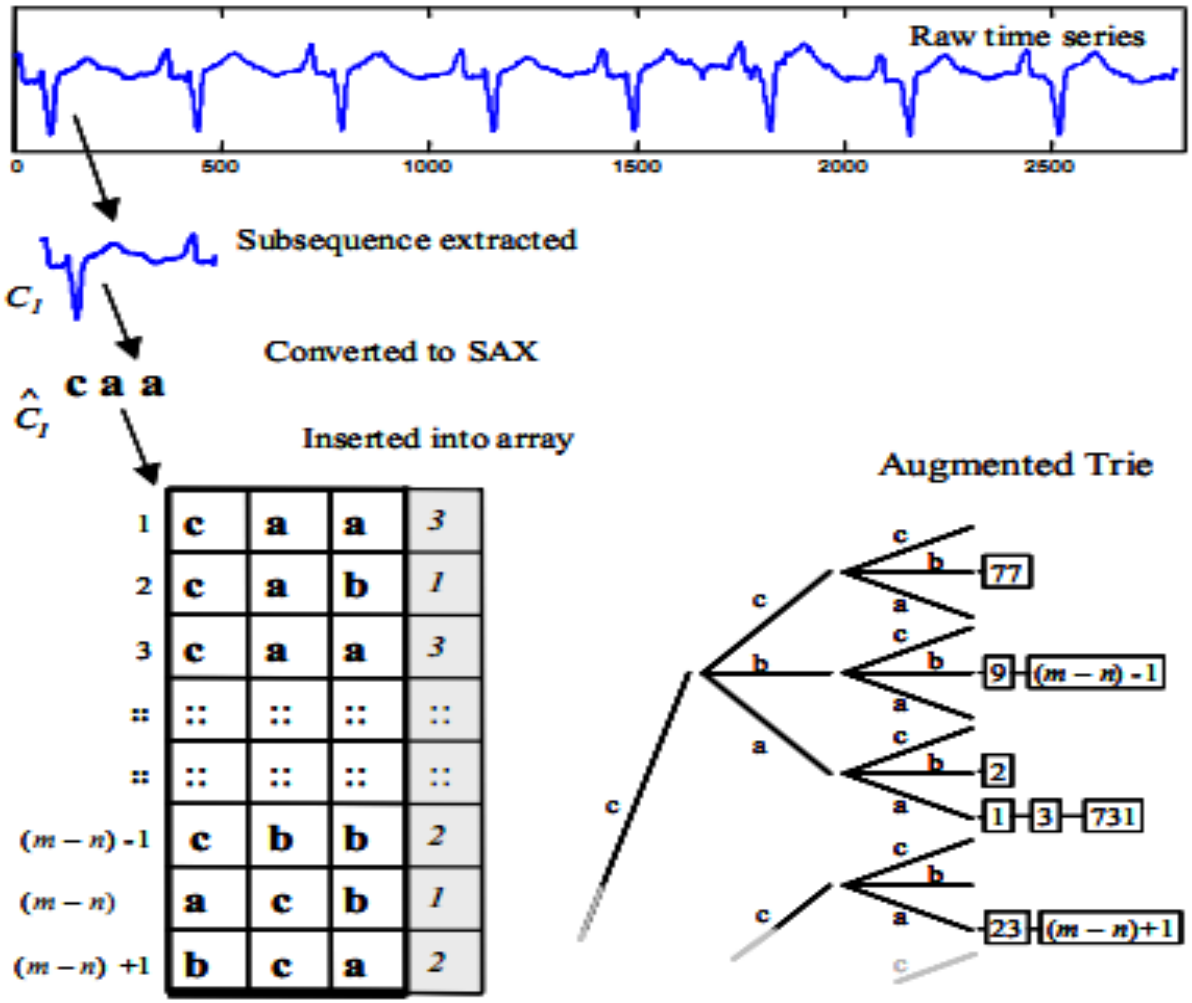
HOT SAX

- Present subsequences of length *n* to SAX words.
  - Use PAA for Dimensionality reduction
- Use an array and an augmented trie to embed all the SAX words

HOT SAX



HOT SAX

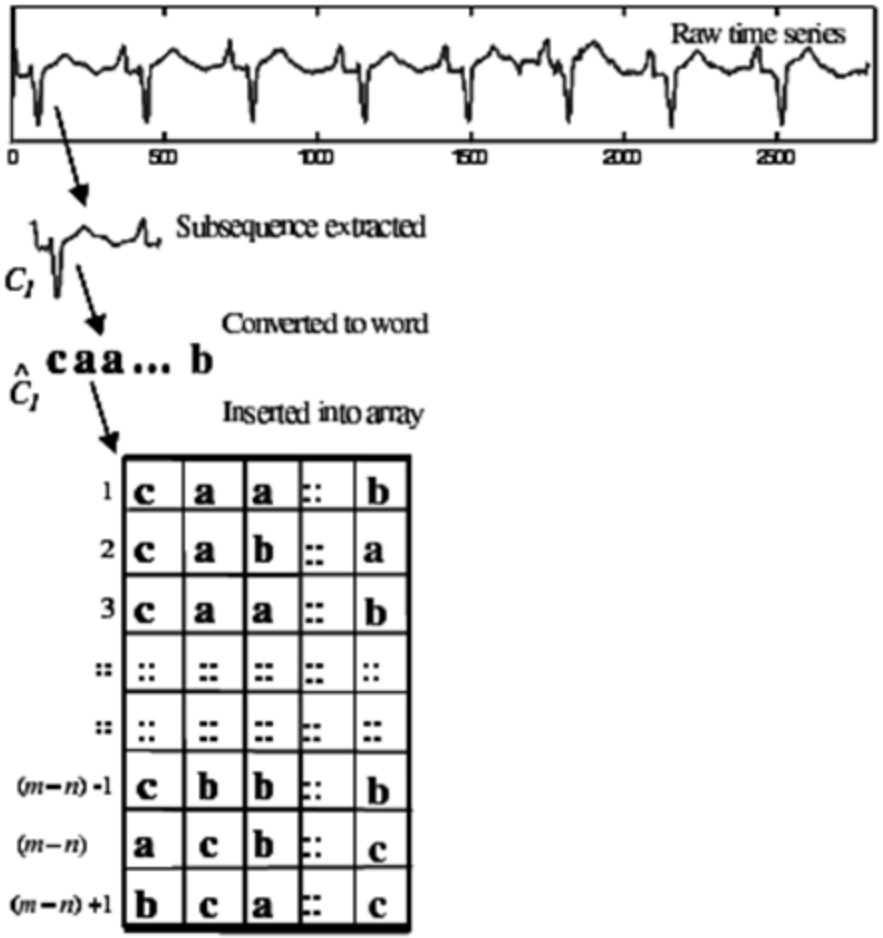

Raw time series
0
500
1000
1500
2000
2500
Subsequence extracted
C_1
Converted to SAX
$\hat{C}_1$ c a a
Inserted into array
1 c a a 3
2 c a b 1
3 c a a 3
:: :: :: :: ::
:: :: :: :: ::
(m − n) -1 c b b 2
(m − n) a c b 1
(m − n) +1 b c a 2
Augmented Trie
77
9
(m − n) -1
2
1
3
731
23
(m − n)+1

HOT SAX

- In outer loop, subsequences presented by words which have smallest appearing frequency in the array are checked first
- In inner loop, subsequences belong to the same left node in the augmented trie with the subsequence chosen at outer loop are checked first.

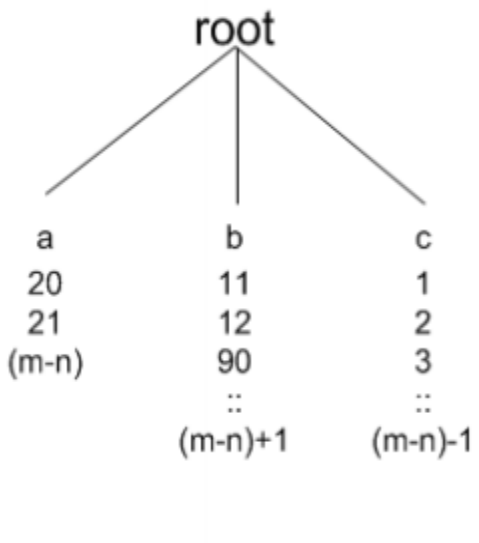
WAT

- Same HOT SAX but use Haar wavelet instead of PAA.

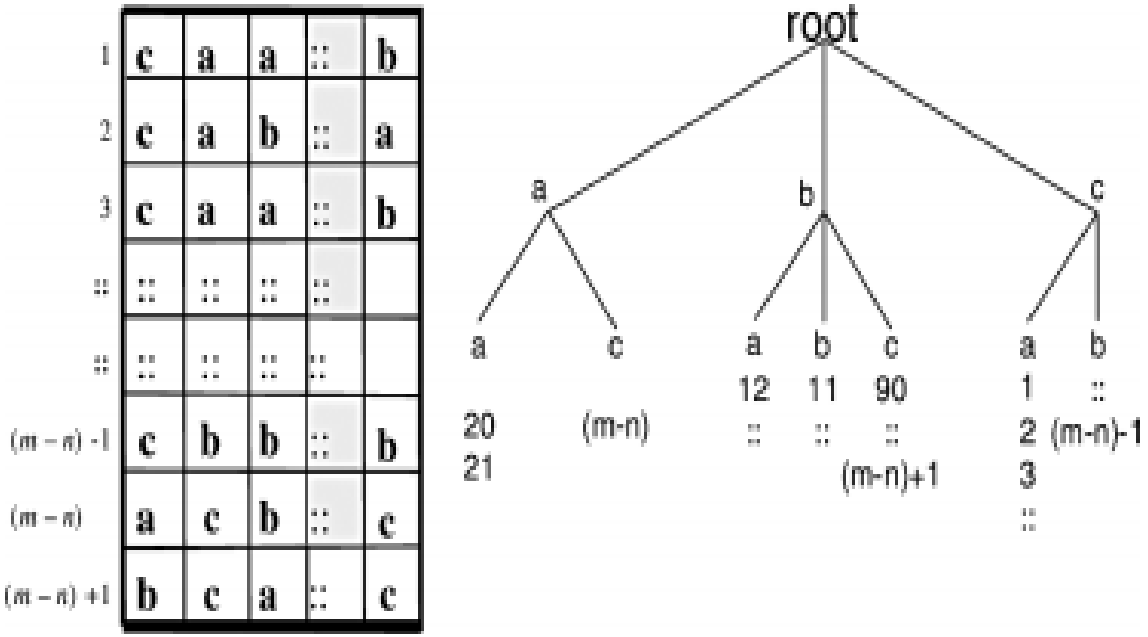


WAT

1	c	a	a	::	b
2	c	a	b	::	a
3	c	a	a	::	b
::	::	::	::	::	
::	::	::	::	::	
$(m-n)$ -1	c	b	b	::	b
$(m-n)$	a	c	b	::	c
$(m-n)$ +1	b	c	a	::	c



WAT

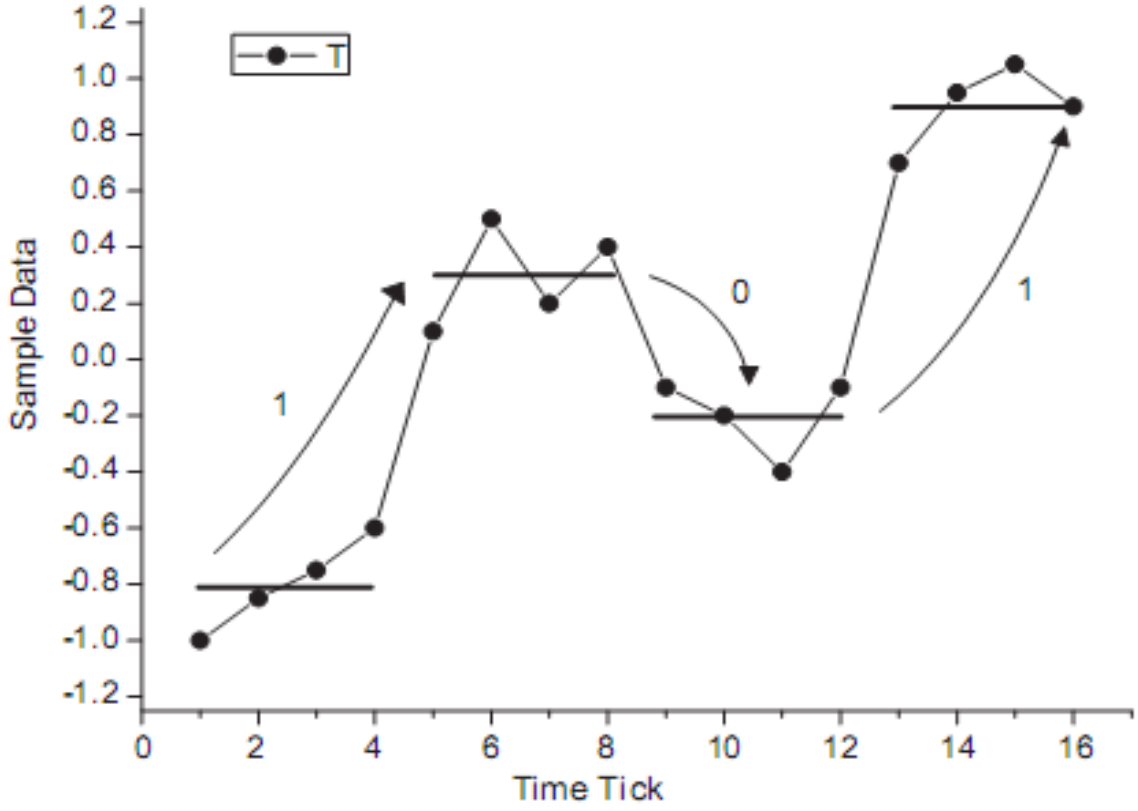

1 c a a :: b
2 c a b :: a
3 c a a :: b
:: :: :: :: ::
:: :: :: :: ::
(m – n) -1 c b b :: b
(m – n) a c b :: c
(m – n) +1 b c a :: c
root
a
b
c
a c
a b c
a b
20
21
(m-n)
12 11 90
:: :: ::
(m-n)+1
1 ::
2 (m-n)-1
3
::

WAT

- In outer loop, subsequences presented by words which have smallest appearing frequency in the array are checked first
- In inner loop, subsequences belong to the same left node in the tree with the subsequence chosen at outer loop are checked first.

Method based on PAA bit representation and clustering

- PAA Bit representation of a series



Method based on PAA bit representation and clustering

- Bit distance between of two PAA bit representation $s = \{s_1, s_2, \ldots, s_{w-1}\}$ and $t = \{t_1, t_2, \ldots, t_{w-1}\}$
- $BitSeries\_Dist(s,t) = \frac{1}{w-1}\sum_{1}^{w-1} BDist(s_i, t_i)$
- Here, $BDist(s_i, t_i) = \begin{cases} 1 & if\ s_i \neq t_i \\ 0 & otherwise \end{cases}$

Method based on PAA bit representation and clustering

- k bit representations are randomly chosen as k cluster centres
- A bit representation is assigned to the nearest cluster.
- In outer loop, subsequences which have bit representations belong to the smallest cluster (cluster has the fewest elements) is checked first
- In inner loop, subsequences belong to the same cluster with the subsequence chosen at outer loop are checked first.

Method based on PAA bit representation and clustering

- 3 above algorithms need to know the length of anomaly subsequence in advance.

Method based on Segmentation and Anomaly Factors

- Proposed by Leng et al. in 2008
- The method includes 2 phase
- Phase 1: use quadratic regression model to segment time series. Regression function is defined as $f(t) = \beta_0 + \beta_1 t + \beta_2 t^2$,
- Phase 2: Calculate anomaly factors for the subsequences and show abnormal patterns.

1. Let $s_1 = 1$ denote the start position of the first segment, l denote the initial length of each segment, $m = s_1$ and then update this segment, the updating procedure is as follows. Calculate the values of β_0 , $\beta_1 t$, and β_2 and the sum of squared errors:

$$SSE = \sum_{i=p}^{p+l-1} (f(i) - t_i)^2$$

2. If SSE < ε_1 let $l = l + 1$, go to step 1, else go to step 3.

3. Let (s_1, e_1) denote this segment, e_1 denotes the end position of this segment and $e_1 = l - 1$.

4. Let $i = 1$, if $Dist((s_1, e_1), (s_1 + i, e_1 + i)) \leq \varepsilon_2$, increment i by 1, calculate it, repeat this procedure until $Dist((s_1, e_1), (s_1 + i, e_1 + i)) > \varepsilon_2$ or $i > (e_1 - s_1)$, let $s_2 = e_1 + i$, the s_2 is the start position of the next segment.

5. Let $m = s_2$, calculate e_2 using the steps 1-3, and then obtain the second segment (s_2, e_2).

6. Calculate s_j using the step 4, let $m = s_j$, calculate the value of e_j using the steps 1 – 3, and then obtain the j-th segment (s_j, e_j).

7. Repeat the above procedure until the end position of the segment is n.

1. Find the maximum value, l_{max} and minimum value l_{min} of the lengths of all extracted segments.
2. Calculate the distance matrix $D = (d_{ij})_{m \times m}$ of segments:

$$d_{ij} = \min_{l_{\min} \le l \le l_{\max}} Dist((s_i, e_i), (s_j, s_j + l)) \quad (1)$$

where $1 \le i, j \le m$ and $i \ne j$.

3. Compute k-distance of each segments based on the distance matrix, let $k\text{-}dist(D)$ denote the set of all these distances.
4. Calculate median($k\text{-}dist(D)$)
5. Calculate anomaly factor of each segment, and determine whether each segment is an anomaly pattern or not basing on the anomaly factor threshold a.
6. Given two anomaly patterns (s_i, e_i) and (s_j, e_j), if they are overlapped, merge them into one pattern.

Method based on Segmentation and Anomaly Factors

- Don’t need to know the length of anomaly patterns in advance
- Use DTW to calculate distance between two subsequences of different length
- Time consuming

Proposed Algorithms

- Use Homothetic Transformation + Modified Euclidean Distance instead of DTW
- Reduce the number of distance calculating in building distance matrix.
- Proposed a segmentation method based on important extreme points as alternative segmentation method

New distance function

- A Homothetic Transformation with center *O* and ratio *k* transforms the point *M* to the point *M'* such that

$$\overrightarrow{OM'} = k \times \overrightarrow{OM} .$$

New distance function

- Apply to time series, a homothety transforms a time series *T* of length *n (T = {y1, y2, …, yn})* to time series *T'* of length *n'* by performing the following steps.
- First, compute *Y_MAX = MAX (y1, …, yn), Y_MIN = MIN(y1, …,yn).*
- Second, set the center I of homothety with the coordinates *X_C = n/2, Y_C = (Y_MAX + Y_MIN)/2*. Next, perform the homothety with the center *I* and the ratio *n'/n*

New distance function

Function $Dist(X, Y)$

if $(X.length == Y.length)$ **return** $Euclid(X, Y)$

else

$MEAN = |(Y.length + X.length)| / 2$

$Y' = Homothety(Y, MEAN)$

$X' = Homothety(X, MEAN)$

return $Euclid(X', Y')$

New distance function

- Minimum Euclidean Distance between

$T' = \{T'_1, T'_2, \ldots, T'_{N'}\}$ and $Q' = \{Q'_1, Q'_2, \ldots, Q'_{N'}\}$

$$D(T', Q') = Min\left\{\sum_{i=1}^{N'} (T'_i - Q'_i - b)^2\right\}$$

$$b = \frac{1}{N'}\sum_{i=1}^{N'} (Q'_i - T'_i)$$

Time Complexity of new distance function: O(n)

Reduce the number of distance calculating

- Propose parameter *r*
- Replace formula (1) to

$$d_{ij} = \min_{l_{lower} \le l \le l_{upper}} Dist((s_i, e_i), (s_j, s_j + l)) \quad (1')$$

Where

$$l_{upper} = \lfloor l_{avg}(1+ r) \rfloor$$

$$l_{lower} = \lceil l_{avg}(1- r) \rceil$$

l_{avg} is mean length of the subsequences.

Alternative segmentation method

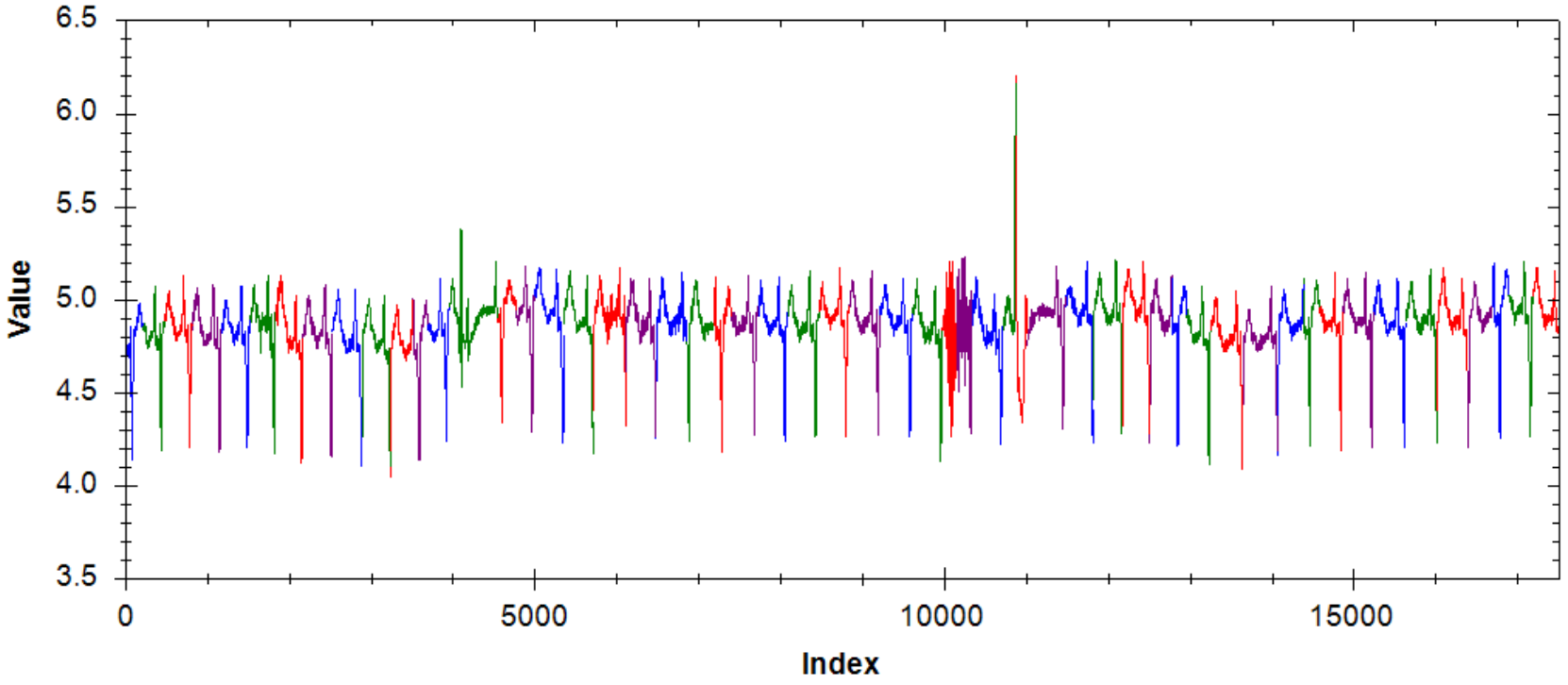
- Based on the concepts of Important Extreme Points, proposed by Pratt and Fink in 2002
- Extract all important extreme points of the time series *T*. The result of this step is a sequence of extreme points $EP = (ep_1, ep_2, \ldots, ep_l)$
- Compute all the candidate patterns iteratively. A candidate pattern $CP_i(T)$, $i = 1, 2, \ldots, l-2$ is the subsequence of *T* that is bounded by extreme points ep_i and ep_{i+2}.

Alternative segmentation method

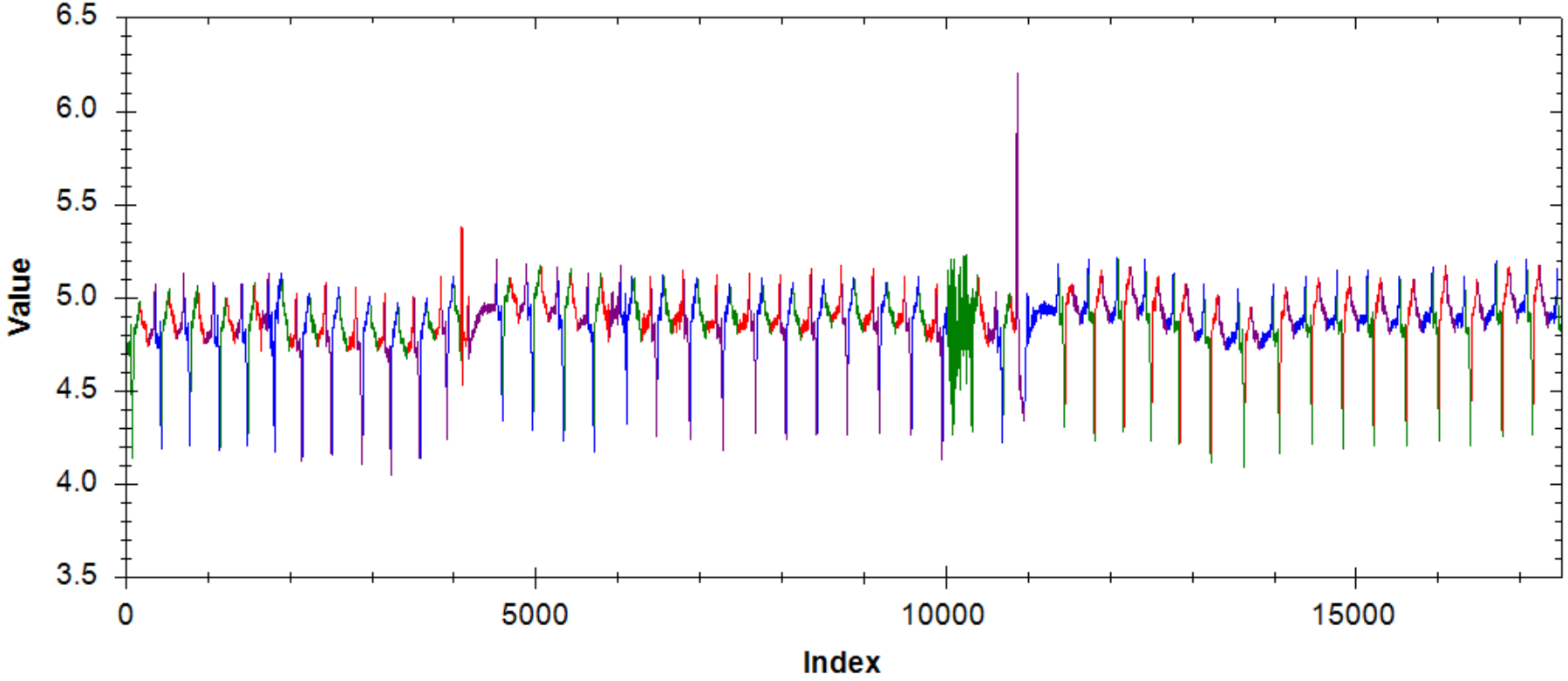
- **Parameter:**

- Compression rate: $R > 1$

- Lower bound for the distance two extracted extreme points: *min_length.* So the minimum length of a subsequence is 2*min_length*



Segmentation based on quadratic regression



Segmentation based on important extreme points

Experimental Evaluation

- VL_QR| HT: The proposed algorithm with quadratic regression
- VL_EP| HT: The proposed algorithm with important extreme points.
- Compare results and running time among VL_QR| HT, VL_EP| HT and HOT SAX
- Compare running time between VL_QR| HT with the original algorithm proposed by Leng et al.

Experimental Evaluation

- Experiments environment : Intel® Core™ 2 Duo 2.0GHz, Ram 3072MB PC
- All algorithms are implemented in Microsoft Visual C# programing language

Compare VL_QR| HT, VL_EP| HT and HOT SAX

- Use 8 time series in diverse domains:

ECG 108, ECG 308, ERP, Memory, Power Demand In Italy, Dutch Power Demand, Stock20 and TEK16

- To compute location difference of anomaly subsequences use:

$$d = |p - q|/l \times 100$$

where p is the start position of the anomaly pattern found by proposed algorithm and q is the start position of the anomaly pattern found by HOT SAX and l is it's length

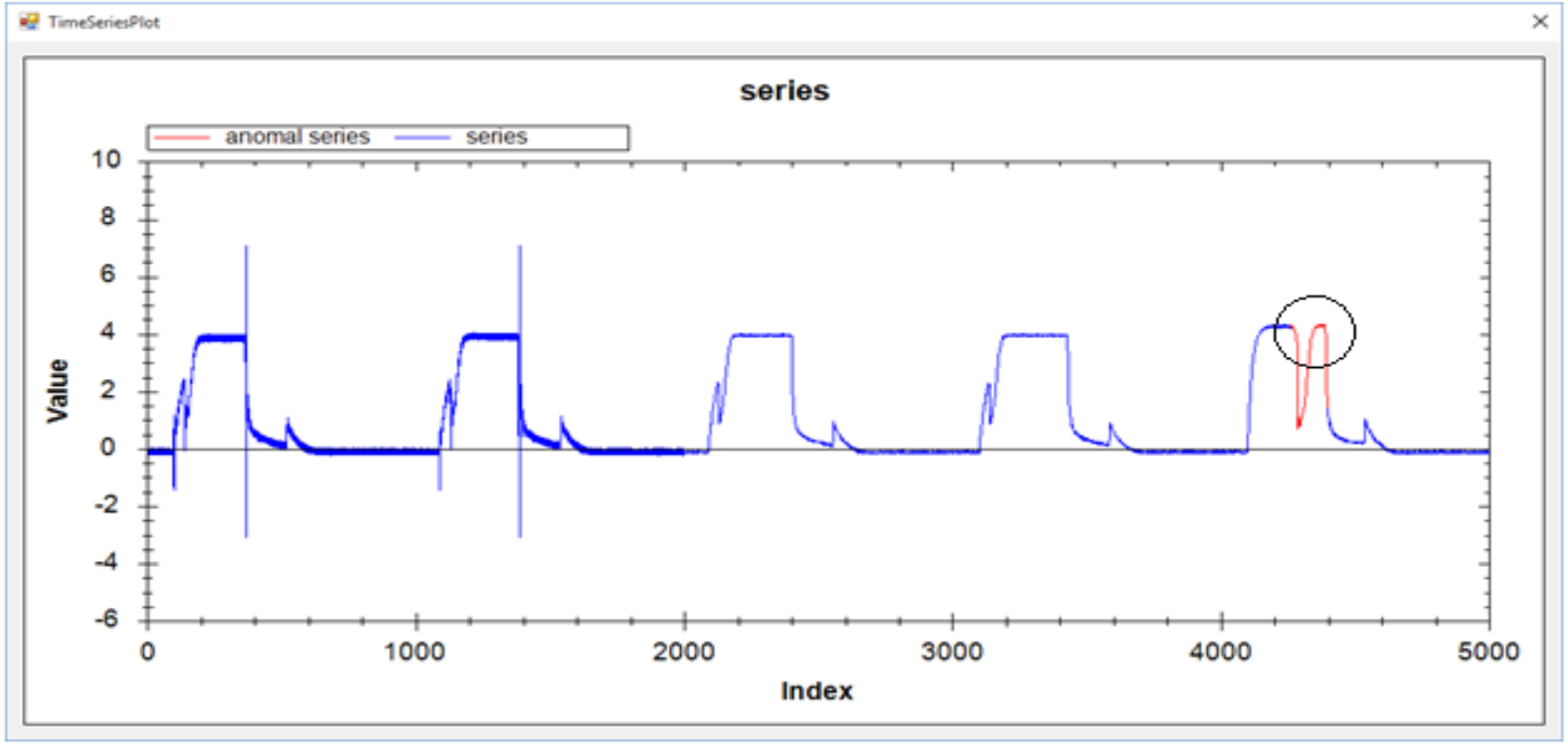
Parameter values in the three algorithms for each series

Datasets	VL_QR| HT	VL_EP| HT	HOT SAX
ECG108 (17500 points)	$\varepsilon_1 = 5.0$, $\varepsilon_2 = 0.3$, $\alpha = 3.5$, $r = 0.1$	$R = 1.04$, $min\_lenth = 50$, $\alpha = 4$, $r = 0.1$	$n = 600$, $PAA\_size = 60$
ECG308 (1300 points)	$\varepsilon_1 = 51000$, $\varepsilon_2 = 50$, $\alpha = 2.2$, $r = 0.1$	$R = 1.03$, $min\_lenth = 10$, $\alpha = 2.5$, $r = 0.23$	$n = 60$, $PAA\_size = 6$
ERP (5000 points)	$\varepsilon_1 = 3.0$, $\varepsilon_2 = 1.0$, $\alpha = 3.0$, $r = 0.15$	$R = 1.42$, $min\_lenth = 10$, $\alpha = 3.5$, $r = 0.1$	$n = 100$, $PAA\_size = 10$
Memory (6875 points)	$\varepsilon_1 = 8.0$, $\varepsilon_2 = 0.1$, $\alpha = 2.2$, $r = 0.1$	$R = 1.1$, $min\_lenth = 40$, $\alpha = 1.6$, $r = 0.1$	$n = 100$, $PAA\_size = 20$
Power Demand Italy (7000 points)	$\varepsilon_1 = 100000$, $\varepsilon_2 = 100$, $\alpha = 2.5$, $r = 0.1$	$R = 1.8$, $min\_lenth = 20$, $\alpha = 3.0$, $r = 0.1$	$n = 300$, $PAA\_size = 30$
Dutch Power Demand (9000 points)	$\varepsilon_1 = 31000000$, $\varepsilon_2 = 500$, $\alpha = 1.6$, $r = 0.2$	$R = 1.7$, $min\_lenth = 180$, $\alpha = 1.33$, $r = 0.25$	$n = 1200$, $PAA\_size = 120$
Stock20 (5000 points)	$\varepsilon_1 = 2.0$, $\varepsilon_2 = 0.1$, $\alpha = 3.5$, $r = 0.1$	$R = 1.01$, $min\_lenth = 200$, $\alpha = 1.5$, $r = 0.1$	$n = 700$, $PAA\_size = 70$
TEK1 (5000 points)	$\varepsilon_1 = 14.5$, $\varepsilon_2 = 0.1$, $\alpha = 2.0$, $r = 0.1$	$R = 1.02$, $min\_lenth = 50$, $\alpha = 1.5$, $r = 0.1$	$n = 128$, $PAA\_size = 8$

Positions of anomaly patterns detected by VL_QR|HT and VL_EP|HT and HOT SAX

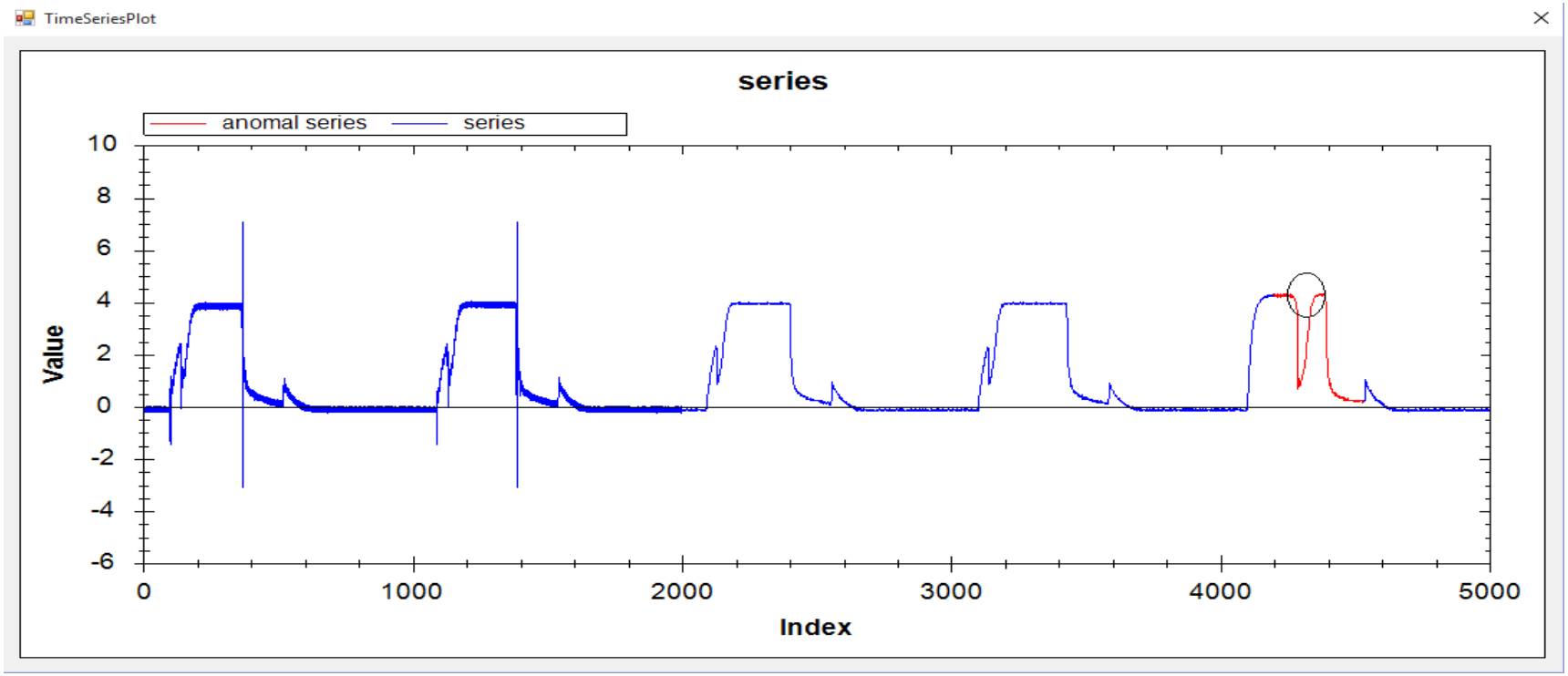
Dataset	VL_QR|HT		VL_EP|HT		HOT SAX
	Start position	Location difference (d)	Start position	Location difference (d)	Start position
ECG 108	10875	1.5%	10792	12.3%	10866
ECG 308	656	33%	673	5%	676
ERP	2617	32%	2602	17%	2585
Memory	2425	2%	2399	11%	2421
Power Demand Italy	5311	4%	5383	13.33%	5343
Dutch Power Demand	6424	1.3%	6277	11%	6466
Stock20	2961	1.7%	2960	1.6%	2949
TEK16	4262	24%	4197	27%	4231

Positions of anomaly patterns detected by VL_QR|HT and VL_EP|HT and HOT SAX



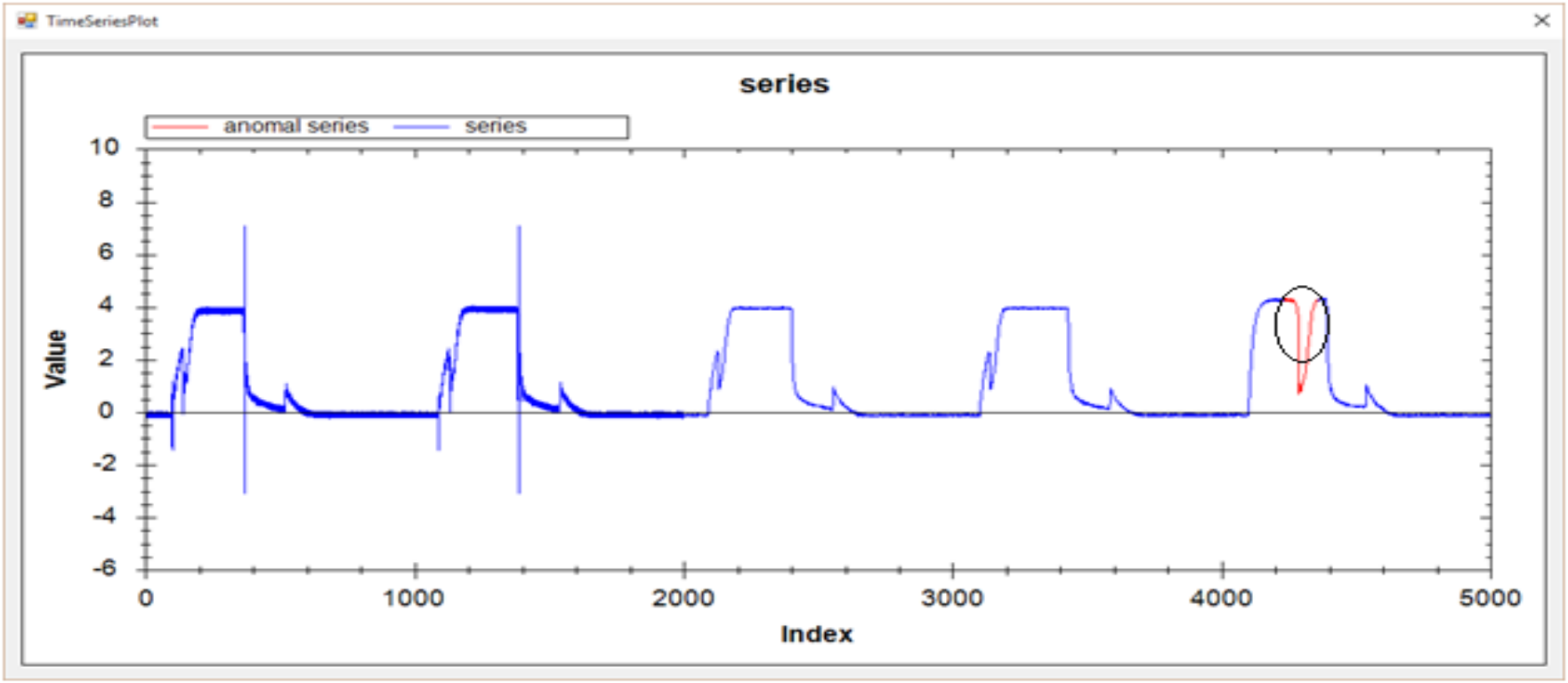
- Result of VL_QR | HT with series TEK16

Positions of anomaly patterns detected by VL_QR|HT and VL_EP|HT and HOT SAX



- Result of VL_EP | HT with series TEK16

Positions of anomaly patterns detected by VL_QR|HT and VL_EP|HT and HOT SAX



- Result of HOT SAX with series TEK16

Running time among VL_QR|HT, VL_EP|HT and HOT SAX (s)

Dataset	VL_QR| HT	VL_EP| HT	HOT SAX
ECG 108	10	47	458
ECG 308	0.31	2	13
ERP	7	14	60
Memory	4	6	79
Power Demand Italy	9	11	102
Dutch Power Demand	1	9	302
Stock20	1	1	94
TEK16	1	1	52

Running time between VL_QR| HT with the original algorithm (s)

Dataset	VL_QR|HT	VL_QR|DTW
ECG 108 (1000 points)	1	14
ECG 308 (1300 points)	1	11
ERP (1000 points)	1	5
Memory (1000 points)	1	4
Power Demand In Italy (3000 points)	6	200
Dutch Power Demand (3000 points)	1	24
Stock20 (3000 points)	2	194
TEK16 (2000 points)	1	81

Conclusion

- VL_QR|HT and VL_EP|HT bring out a remarkable improvement for the original algorithm in time efficiency without compromising anomaly detection accuracy.
- Experimental results on eight datasets demonstrate these algorithms outperform the VL_QR|DTW in time efficiency.
- Not easy to estimate the regression error threshold ε_1 and the non-self match threshold ε_2 in the segmentation phase of VL_QR|HT.
- Not easy to find the *R* in VL_EP|HT.
- Future work: apply some more advanced method of time series segmentation.

REFERENCES

- D. Berndt, J. Clifford, “Finding patterns in time series: a dynamic programming approach”, Journal of Advances in Knowledge Discovery and Data Mining, AAA/MIT Press, Menlo Park, CA, pp. 229-248, 1996.
- Y. Bu, T.W. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin, “WAT: Finding top-K discords in time series database”, Proceedings of the 2007 SIAM International Conference on Data Mining (SDM'07), Minneapolis, MN, USA, April 26-28, 2007.
- V. T. Huy, “Anytime k-medoids clustering of time series under dynamic time warping using an approximation technique”, Master Thesis, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam, 2015.
- E. Keogh, “Exact indexing of dynamic time warping”, Proceedings of 28th International Conference on Very Large Data Bases , Hong Kong, pp. 406-417, 2002.
- E. Keogh, J. Lin, and A. Fu, “HOT SAX: efficiently finding the most unusual time series subsequence”, Proc. of 5th ICDM, Houston, Texas, pp. 226–233, 2005.
- E. Keogh, www.cs.ucr.edu/~eamonn/discords/ (accessed on January 24 - 2015).
- N.H. Kha and D. T. Anh, “From cluster-based outlier detection to time series discord discovery”, In: Trends and Applications in Knowledge Discovery and Data Mining – PAKDD 2015 Workshops: Big_PMA, VLSP, QIMIE, BAEBH, Ho Chi Minh City, Vietnam, May 19-21, X. L. Li et al. (Eds.), LNAI 9441, Springer, pp. 16-28, 2015.
- D. Lemire, “Faster retrieval with a two-pass dynamic-time-warping lower bound”, Pattern Recognition, vol. 42, no. 9, pp. 2169-2180, 2009.
- M. Leng, X. Chen and L. Li, “Variable length methods for detecting anomaly patterns in time series”, Proc. of Int. Symposium on. Computational Intelligence and Design (ISCID'08), Vol. 2, 2008.
- G. Li, O. Braysy, L. Jiang, Z. Wu, Y. Wang, “Finding time series discord based on bit representation clustering”, Knowledge-Based Systems, vol.52, pp. 243-254, 2013.
- A. L. I. Oliveira, F.B.L. Neto, and S.R. L. Meira., “A method based on RBF-DAA neural network for improving Novelty detection in time series”, Proc. of 17th International FLAIRS Conference, AAAI Press, Miami Beach, Florida, USA, 2004.
- K. B. Pratt and E. Fink, “Search for patterns in compressed time series”, International Journal of Image and Graphics , vol. 2, no. 1, pp. 89-106, 2002.
- H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition”, IEEE Trans. Acoustics, Speech, and Signal Proc., vol. ASSP-26, pp. 43-49, 1978.
- S. Salvador, P. Chan, “Learning states and rules for time series anomaly detection”, Applied Intelligence, vol. 23, no.3, pp. 241 - 255. 2005.
- C. D. Truong, H. N. Tin, D. T. Anh, “ Combining motif information and neural network for time series prediction”, Int. Journal of Business Intelligence and Data Mining, vol. 7, no. 4, pp. 318-339, 2012.

Q & A

Thank you!