DISCOVERING TIME SERIES DISCORD BASED ON A DECRETE METHOD

Nguyen Thanh Son
HCM City University of Technology and Education, Vietnam

Received 04/03/2019, Peer reviewed 04/04/2019, Accepted for publication 9/5/2019

ABSTRACT

A time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Discord in a long time series is a subsequence which is the most different from all the rest of subsequences of that time series. Time series discord discovery is one of problems which has received a lot of attention lately. In this paper, we propose a new algorithm for time series discord discovery which is based on the discrete method called Symbolic Aggregate approXimation (SAX) method using distance measure in SAX space and Euclidean distance associated with the idea of early abandoning. Our proposed method only need to scan the database two times to discover time series discord exactly and it is very simple to implement. The experimental results showed that our proposed method outperforms the similar method proposed by Yankov et al., in terms of runtime while the accuracy is the same.

***Keywords*:** *time series; time series discord; SAX method; discord discovery; early abandoning.*

1. INTRODUCTION

A time series is a series of real numbers which represent data points indexed in time order. Time series data arise in so many applications of various areas ranging from science, engineering, business, finance, economy, medicine to government. Time series discord discovery is one of problems which has received an increasing amount of attention lately.

Time series discord is defined as a subsequence which is maximally different to all the rest of subsequences of a long time series. Time series discord discovering has been used for fault diagnostics, intrusion detection, data cleansing, etc.

In 2005, Keogh et al., introduced a formal definition of time series discord [1]. Since then, many algorithms for discovering time series discord has been proposed. Most of the proposed algorithms often come with the assumption that the data reside in main memory and they need to scan databases many times to discover a discord. For many real-world problems this is not the case. So, Yankov et al., proposed a new algorithm in which time series discord can be discovered with only two linear scans of the disk and a tiny buffer of main memory. Their proposed algorithm is exact and it is very simple to implement [2].

In our work, we proposed an algorithm for discovering time series discord which is based on SAX method using distance measure in SAX space and Euclidean distance associated with the idea of early abandoning. Similar to Yankov's algorithm, our proposed method only need to scan the database two times to discover time series discord exactly and it is very simple to implement. The first scan is used to selection discord candidates and the second one is used to refine the discord candidates for pruning false discords.

We experiment with the proposed algorithm on time series datasets of various areas. The experimental results showed that

our proposed method outperforms the similar method proposed by Yankov et al., in terms of runtime while the accuracy is the same.

The rest of the paper is organized as follows. In Section 2 we review related work and basic concepts. Section 3 describes our approach for discovering time series discord. Section 4 presents our experimental evaluation on real datasets. In section 5 we include some conclusions and suggestions for future work.

2. FUNDAMENTAL CONCEPTS AND RELATED WORK

2.1. Fundamental Concepts

In this subsection, we give some basic concepts and the definitions of the terms formally.

- **Definition 1**. *Euclidean distance:* Given a pair of time series $Q = \{q_1, \ldots, q_n\}$ and $C = \{c_1, \ldots, c_n\}$, the Euclidean distance between Q and C is defined as:

$$D(Q,C) = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \quad (1)$$

The Euclidean distance metric is the simplest method to measure the similarity of time series and has been widely used for pattern matching [3]. To speed up the calculation of Euclidean distance for a pair of time series we can use the idea of early abandoning introduced in [4].

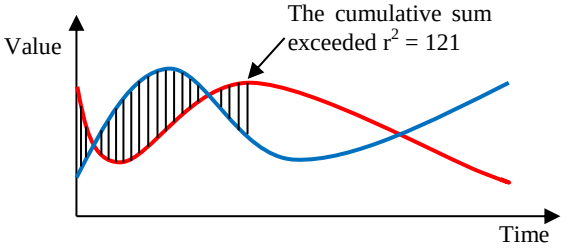

***Fig. 1** An illustration of the idea of early abandoning technique*

The idea of early abandoning is performed as follows: when the Euclidean distance is calculated for a pair of time series, if the cumulative sum is greater than the current best-so-far distance at a certain point we can abandon the calculation because this pair of time series is not the best match. Fig. **1** shows the intuition behind this technique. In this example, the current best-so-far distance is supposed of 11. At the point the squared Euclidean distance of 121 we can stop this calculation.

- **Definition 2**. *Time series*: A *time series* is a real value sequence of length n over time, i.e. if T is a time series then $T = (t_1, \ldots, t_n)$ where t_i is a real number.

- **Definition 3**. *Subsequence*: Given a time series $T = (t_1, \ldots, t_n)$, a *subsequence* of length $m < n$ of T is a sequence $S = (t_i, \ldots, t_{i+m-1})$ with $1 \le i \le n - m + 1$.

Since all subsequences may potentially be discords, we have to compare any subsequence to all remaining subsequences. However, the best matches of a subsequence tend to be located some points to the left or to the right of the subsequence in question. Such matches are called trivial matches and they have to be excluded from the result of discovering discords.

- **Definition 4**. *Non-trivial match*: Given a time series T, containing a subsequence C_p of length m beginning at position p and a matching subsequence C_q beginning at q, we say that C_q is a non-trivial match to C_p if $|p - q| \ge m$.

- **Definition 5**. *Time series discord*: Given a time series T, the subsequence C of length n is the most significant discord in T if the distance to its nearest non-trivial match Q is largest. It means that for an arbitrary subsequence $M \in T$, $\min(D(C, Q)) \ge \min(D(M, P))$, where Q, P are subsequences in T and Q, P are non-trivial matches of C and M , correspondingly.

- **Definition 6**. K^{th} *Time series discord*: Given a time series T, the subsequence C of length n beginning at position p is the K^{th} significant discord in T if C has the K^{th} largest distance to it nearest non-trivial match

and there is no overlap region between C and the i^{th} discord beginning at position q, for all $1 \leq i < K$. It means $|p - q| \geq n$.

- **The zero mean normalization.**

A time series $T = \{t_1, t_2, \ldots, t_n\}$ can be transformed to a normalized sequence $T' = \{t'_1, t'_2, \ldots, t'_n\}$ so that T' has mean of zero and a standard deviation of one by following formular:

$$t'_i = (t_i - \text{mean}(T)) / \text{std}(T)$$

where, mean(T) is the mean value and std(T) is the standard deviation of time series T.

- **The PAA representation.**

A time series T of length n, $T = (t_1, t_2, \ldots, t_n)$, can be transformed into the w dimensional space ($w << n$) based on PAA method by following steps: first, time series T is divided into w equal-sized segments, then the mean value of the data within each segment is calculated. The vector of these values will be an approximation representation of time series T. It means that a time series T of length n can be represented approximately in a w dimensional space by a vector $V = v_1, v_2, \ldots, v_w$, where v_i is calculated by the following formula [5]:

$$v_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j \qquad (2)$$

- **The SAX representation.**

Given the normalized time series which have highly Gaussian distribution. The breakpoints that will produce equal-sized areas under Gaussian curve can be simply determined by looking them up in a statistical table. For example, in Table 1 we show the breakpoints that divide a Gaussian distribution of equal-sized regions from 3 to 8.

The breakpoints here are sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a Gaussian curve from β_i to $\beta_{i+1} = 1/a$, where a is the number of equal-sized areas under Gaussian curve ($\beta_0 = -\infty$ and $\beta_a = \infty$)

Table 1. *A lookup table for determining breakpoints with a from 3 to 7.*

β_i \ a	3	4	5	6	7
β_1	-0.43	-0.67	-0.84	-0.97	-1.07
β_2	0.43	0	-0.25	-0.43	-0.57
β_3		0.67	0.25	0	-0.18
β_4			0.84	0.43	0.18
β_5				0.97	0.57
β_6					1.07

After determining the breakpoints, we can transform a PAA representation of time series to SAX representation by mapping all PAA coefficients to symbols as follows: all PAA coefficients less than the smallest breakpoint are mapped to the symbol ‘a’, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol “b”, etc. Fig. **2** illustrates an example of a PAA representation of time series is mapped into SAX symbols with $a = 3$ (using three symbols ‘a’, ‘b’ and ‘c’. In this example, the PAA representation is mapped into SAX sequence ‘baabccbc’.

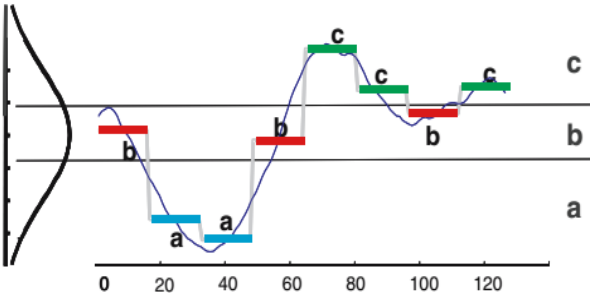

Fig. 2 *An example of a PAA representation of time series is mapped into SAX symbols ([5])*

Note that, assuming normal distribution is really true for a large group of the time series except a small subset of time series does not follow. So, the efficiency can be slightly deteriorated if the time series is not obeyed normal distribution. However, the correctness of the algorithm is not affected. The correctness of the algorithm is ensured by the lower-bounding property of the distance measure in the SAX space [5].

- **The distance measure in SAX space.**

To compare the similarity between two SAX sequence Q and C we can use the distance measure, $D_{SAX}(Q, C)$ which is defined as follows [5]:

$$D_{SAX}(Q,C) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(d(q_i,c_i))^2} \quad (3)$$

where n: the length of time series in the original space

w: the length of SAX sequence

$d(q_i, c_i)$: the distance between two i^{th} symbol in two sequences correspondingly.

The distance $d(q_i, c_i)$ can be implemented using a lookup table as illustrated in table 2. In this case the distance between two symbols can be read off by examining the corresponding row and column.

Table 2. *A lookup table for a = 4*

	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

The value in the cell at row r and column c of the lookup table can be calculated by the following fomular:

$$cell(r,c) = \begin{cases} 0 & if\ |r-c| \le 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)} & \text{otherwise} \end{cases} \quad (4)$$

2.2. Related Work

Many algorithms have been introduced to solve the time series discord discovery problem since it was formalized in 2005 [1]. In [1] Keogh et al. proposed a fast heuristic technique (called Hot SAX) for pruning quickly the data space and focusing only on the potential discords. Fu et al. proposed a new algorithm based on Haar Wavelet transform to determine dynamically the word size for the compression of subsequences in 2006 [6]. In 2007, Bu et al. proposed a new method called WAT (Wavelet and augmented trie) which is based on Haar Wavelet transforms and augmented trie to mine the top-k discords from time series data [7]. Also in this year, Chuah et al. proposed an anomaly detection method. It is based on time series analysis in order to determine whether a stream of real-time sensor data contains any abnormal heartbeats. If anomaly exists, that time series segment will be transmitted via the network to a physician so that experts can further diagnose the problem and take appropriate actions [8].

In 2010, Lin et al. introduced a new approach to the anomaly detection problem. First, this method uses subseries to join to obtain the similarity relationships among subseries of the time series data. Then it converts the anomaly problem to graph-theoretic problem which can be solved by existing graph-theoretic algorithm [9]. A new method, called Disk aware discord discovery is proposed by Yankov et al. This method includes two phases: (1) a candidate selection phase and (2) discord refinement phase. In phase 1, the algorithm performs a linear scan through the database T. Each $T_i \in T$ is validated to see if it is likely to be a discord or it is omitted. Phase 2 accepts as an input a candidate set $C \subset T$, which is a result from phase 1. It will prune the set C to retain only the true discord [2].

In 2011, Buu et al. proposed a new time series discord discovery algorithm, called HOTiSAX. This algorithm incorporates iSAX (**i**ndexable **S**ymbolic **A**ggregate appro**X**imation) representation in Hot SAX instead of SAX representation [10]. Khanh et al. proposed a new method for discord discovery in time series, called WATiSAX. This algorithm employs iSAX representation in WAT algorithm [11]. Luo et al. proposed a new method which exploits a recurrence structure of time series and uses a reference function that makes the search algorithm more efficient and robust [12]. Jones et al.

introduced a new algorithm for discovering anomalies in real-valued multidimensional time series. First, this method uses an exemplar-based model for detecting anomalies in single dimensional time series then uses a function that predicts one dimension from a related one [13].

Pavel Senin et al. proposed a new algorithm which use grammar induction to aid anomaly detection without any prior knowledge in 2015. First, this algorithm discretizes continuous time series values into symbolic form, then it infers a context free grammar. Finally, the algorithm uses its hierarchical structure to effectively and efficiently discover anomalies [14].

In 2016, T. S. Nguyen proposed a new algorithm for discovering time series discord based on R*-tree. This method needs a single scan over the entire time series database and a few times to read the original disk data in order to validate the results [15].

A construction to discover the time series discord under the multi-party privacy preserving is proposed by Chunkai Zang et al. in 2017 [16]. This method is based on the homomorphic encryption and it allows the inputs to be encrypted under different independent public keys. C. M. Pham et al. proposed a new algorithm for discovering discord in streaming time series, called HS-Squeezer. It is an improvement of HOT SAX algorithm in which this method uses clustering instead of augmented trie to arrange two ordering heuristics in HOT SAX [17]. A novel computational framework to discover discords from multivariate time series data, called LRRDS (Local Recurrence Rate based Discord Search) is proposed by Min Hu et al. in 2019. First, the original time series data is transformed to a recurrence plot. Then this plot is analyzed to discover discords [18].

3. OUR APPROACH

In this section, the discussion is limited to the case where the time series database T contains $|T|$ separate time series of length n because all subsequences extracted from a time series T can form a *subsequence database* in which each subsequence can be regarded as a time series. In case that the database contains subsequences from a long time series the basic algorithm is unchanged but the trivial matches must be rejected.

Our proposed method can be divided into two stages: the candidate selection stage and the refinement stage. In the candidate selection stage, first each normalized time series is transformed into the Piecewise Aggregate Approximation (PAA) representation and then the PAA representations are mapped into the discrete strings using Symbolic Aggregate approXimation (SAX) method. After that discord candidates will be discoved in the SAX space using the distance measure $D_{SAX}(Q, C)$. In refinement phase, the true discord will be retrieved based on the candidate set by finding on this small candidate set in original space using Euclidean distance associated with the idea of early abandoning.

Note that, each time series need to normalize to have mean of zero and a standard deviation of one before transforming it to PAA representation because it is understook it is meaningless to compare time series with different offsets and amplitudes [3].

Fig. **3** illustrates the algorithm for discovering discords based on SAX representation. In this figure, stage 1 is shown from line 1 to line 13. In this stage, the algorithm will find all discord candidates based on comparing SAX sequences. To do this the algorithm needs to perform a scan through the database. For each time series which are normalized to have mean of zero and a standard deviation of one, they are transformed to SAX sequences based on PAA dimensional reduction method. In Fig. **3**, S is a list of SAX representations (line 1) and C is used to contain IDs of discord candidates (line 2). For each time series T_i in the database, the algorithm compares it to

every sequence in C (line 3 to line 13). If the candidate already in C is not a discord candidate, it is removed from this set (line 7 and line 8). At the end of this phase, the ID of the time series T_i is added to C if it is likely to be a discord (line 12).

Stage 2 which is similar to that of Yankov' method [2] is illustrated from line 14 to line 27. In this phase, the algorithm uses the discord candidate set C which is the result in phase 1.

Algorithm Discovering discords based on SAX

Input: T : normalized time series database
w : dimensional number in feature space
a : number of symbols using in SAX representation
ε_1 : discord threshold in SAX space
ε_2 : discord threshold in original space
Output: C: list of the true discords
dist: list of nearest neighbor distance to the discords

```
1: S1 = SAX_representation(T1, w, a)
2: C = 1
3:  For i = 2 to |T| do {
4:      Si = SAX_representation(Ti, w, a)
5:      candidate = true
6:      For  ∀j ∈ C do {
7:          If (DSAX(Si, Sj) < ε1) then {
8:              Remove j from C
9:          candidate = false
10:           }
11:     }
12:     if (candidate) then C = C ∪ i
13: }
14: For i = 1 to |C| do disti = ∞
15: For ∀Ti ∈ T do {
16:       For ∀j ∈ C do {
17:           if  Ti = = Tj  then   continue
18:           d = D_Early_abandon(Ti, Tj, distj)
19:           if d < ε2 then {
20:               C = C \ j
21:               dist = dist \ distj
22:               }
23:           else  {
24:               distj = min(distj, d)
25:           }
26:       }
27: }
```

***Fig. 3** The algorithm for discovering discords based on SAX representation.*

Although all sequences in C are assumed to be discords. But it is unknown which items in C are true discords, and what their actual discord distances are. So, the algorithm needs to prune the set C in order to remove false discords by checking all candidate sequences in original space. Initially, all distance from a candidate sequence to its nearest neighbor are set to infinity (line 14). Then the algorithm compares all items in C to each time series in the database. The distance between two sequence in original space is calculated by using Euclidean distance associated with the idea of early abandoning for optimization (line 18). Based on the distance calculated in the original, for each T_i there are three situations [2]:

(1) If the distance between a discord candidate in C and T_i is greater than the current value of $dist_j$, the algorithm does nothing.

(2) If the distance between a discord candidate in C and T_i is less than ε_2, this candidate can be permanently removed from C because it can not be a discord (line 20 and line 21).

(3) If the distance between a discord candidate in C and T_i is less than the current value of $dist_j$, but still greater than ε_2, the current distance to the nearest neighbor is updated (line 24)

4. EXPERIMENTAL RESULTS

Our experiments are conducted on a Core i5, 2.4 GHz, 4.0 GB RAM. Visual Microsoft C# is used to implement algorithms.

We tested on three different publicly available datasets: Stock, ECG and Federal Fund. They are published in the internet for free public download [19]. We compare our proposed approach to the similar method proposed by Yankov et al., in term of run time and accuracy. This method is selected to compare because it is similar to our proposed method and it is an exact algorithm.

We conducted the experiments on the datasets with cardinalities ranging from 2000 to 15000 and the different lengths of discord from 64 to 1024. With SAX representation, we set the number of breakpoints to 3.

In this paper, we only show some typical experimental results for brevity. Fig. **4** shows the running time from the experiments of two methods on the Stock dataset of size 10000 with different discord lengths.

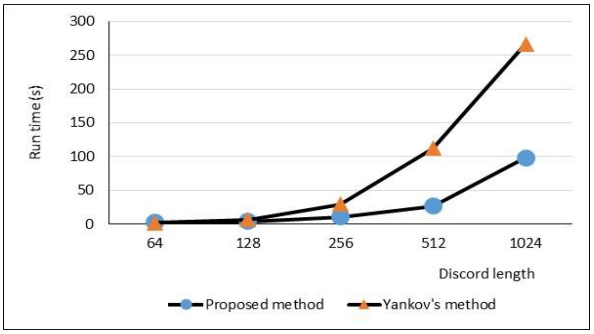

Fig. 4 *The running time of two methods on the Stock dataset with different discord lengths.*

As you can see in this figure, the runtime of our proposed method is less than that of Yankov's method. The different in runtime is negligible in the case of short discord length. But it will be much different when the discord length increases.

Fig. **5** shows the running time from the experiments of two methods on the Stock dataset with different sizes. The fixed discord length is 512.

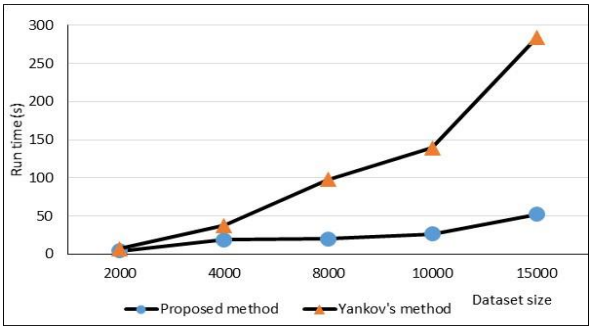

Fig. 5 *The running time of two methods on the EEG dataset with different size and fixed discord length of 512.*

This figure also shows that the runtime of our proposed method is less than that of Yankov's method in this case. The different in runtime is negligible in the case of small database size. But it will be much different when the database size increases.

Fig. **6** shows the running time from the experiments of two methods on the three different datasets with the fixed size of 10000. The fixed discord length is 128.

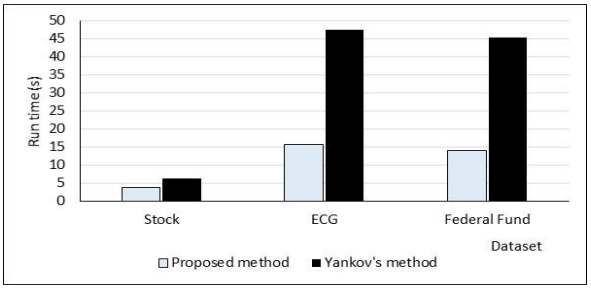

Fig. 6 *The running time of two methods on three datasets with fixed size of 10000 and fixed discord length of 128.*

This figure also shows that the runtime of our proposed method is less than that of Yankov's method in this case.

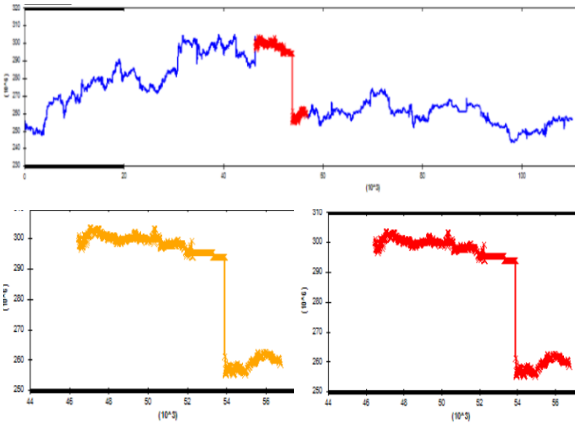

Fig. 7 *The plots of Stock dataset (above figure) and the discords discovered by Yankov's method (below and left figure) and by proposed method (below and right figure).*

For time series discord discovering problem, the accuracy of the proposed algorithm is usually based on human analysis of the discords discovered by that algorithm ([1], [6], [7], [20]). That means if the discords identified by a proposed algorithm on most of the test dataset are almost the same as those identified by the baseline discord discovery algorithm (here Yankov' algorithm is chosen as the baseline algorithm), we can say that the proposed discord discovery algorithm brings out the same accuracy as the baseline algorithm.

Fig. **7** shows the plots of Stock dataset (above figure) and discord of length 1024 discovered from the experiments of two methods on EEG dataset of size 10000. As you can see in figure 7, the discords

discovered by two methods are exactly the same. Experiment on the remain datasets also shows similar results.

5. CONCLUSIONS

We have introduced a new algorithm to discover discord in a long time series which uses SAX representation associated with the distance measure in SAX space and Euclidean distance combining with the idea of early abandoning. The experiments on the different datasets demonstrate that our proposed method outperforms the original method in terms of runtime while the accuracy is the same.

In the future, we plan to research the development of an analytical method for the choice of discord length.

REFERENCES

[1] E. Keogh, J. Lin, J and A. Fu, "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence," in *the 5th IEEE International Conference on Data Mining (ICDM 2005)*, 2005.

[2] D. Yankov, E. Keogh and U. Rebbapragada, "Disk aware discord discovery: Finding unusual time series in terabyte sized datasets," *Knowledge and Information Systems,* vol. 17, no. 2, pp. 241-261, 2008.

[3] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and," in *the 8th ACM SIGKDD Int'l Conference on Knowledge*, Edmonton, Alberta, Canada, 2002.

[4] A. Mueen, E. Keogh, Q. Zhu, S. Cash and B. Westover, "Exact Discovery of Time Series Motifs," in *Proc. of SIAM Int. Conf. on Data Mining*, 2009.

[5] Jessica Lin, Eamonn Keogh, Li Wei, Stefano Lonardi, "Experiencing SAX: a novel symbolic representation of time series," *Journal of Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107-144, 2007.

[6] A. Fu, O. Leung, E. Keogh and J. Lin, "Finding Time Series Discords Based on Haar Transform," in *Lecture Notes in Computer Science, Advanced Data Mining and Applications*, Heidelberg, Springer Berlin, 2006.

[7] Y. Bu, T-W. Leung, A. Fu, E. Keogh, J. Pei and S. Meshkin , "WAT: Finding Top-K Discords in Time Series Database," in *the 2007 SIAM International Conference on Data Mining (SDM'07)*, Minneapolis, MN, USA, 2007.

[8] Chuah, Mooi Choo, and Fen Fu, "ECG anomaly detection via time series analysis," in *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, Springer Berlin Heidelberg, 2007.

[9] Lin Yi, Michael D. McCool, and Ali A. Ghorbani, "Motif and anomaly discovery of time series based on subseries join," in *IAENG International Conference on Data Mining and Applications*, 2010.

[10] H. T. Q. Buu and D. T. Anh, "Time Series Discord Discovery Based on iSAX Symbolic Representation," in *the Third International Conference on Knowledge and System Engineering (KSE 2011)*, Hanoi, Vietnam, 2011.

[11] N. D. K. Khanh and D. T. Anh, "Time series discord discovery using WAT algorithm and iSAX representation," in *the Third Symposium on Information and Communication Technology (SoICT'12)*, ACM New York, NY, USA, 2012.

[12] W. Luo, M. Gallagher and J. Wiles, "Parameter-free search of time-series discord," *Journal Of Computer Science And Technology*, vol. 28, no. 2, pp. 300-310, 2013.

[13] M. Jones, D. Nikovski, M. Imamura, T. Hirata, "Anomaly Detection in Real-Valued Multidimensional Time Series," in *Proc. of 2014 ASE BIGDATA/ SOCIALCOM/ CYBERSECURITY Conference*, Stanford University, 2014.

[14] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, "Time series anomaly discovery with grammar-based compression," in *18th International Conference on Extending Database Technology (EDBT)*, Brussels, Belgium, 2015.
[15] T. S. Nguyen, "time series discord discovery based on r*-tree," *Journal of Science, HCM City University of Education, Special Issue: Natural Science and Technology,* vol. 12, no. 90, pp. 133-144, 2016.
[16] Chunkai Zhang, Haodong Liu and Ye Li, "Time Series Discord Discovery Under Multi-party Privacy Preserving," in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC),* 2017.
[17] C. M. Pham, M. D. Bui and T. A. Duong, "Discord Discovery in Streaming Time Series based on an Improved HOT SAX Algorithm," in *The Ninth International Symposium on Information and Communication Technology (SoICT 2018)*, Danang, Vietnam, 2018.
[18] Min Hu, Xiaowei Feng, Zhiwei Ji, Ke Yan, Shengchen Zhou, "A novel computational approach for discord search with local recurrence rates in multivariate time series," *Information Sciences,* vol. 477, no. March 2019, pp. 220-233, 2019.
[19] E. Keogh and T. Folias, "The UCR Time Series Data Mining Archive," 2013. [Online]. Available: http://www.cs.ucr.edu/~eamonn/.
[20] E. Keogh, S. Lonardi, B. Chiu, "Finding surprising patterns in a time series database in linear time and space," in *KDD 2002: Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2002.

Corresponding author:
Nguyen Thanh Son
HCM City University of Technology and Education
Email: sonnt@hcmute.edu.vn