การค้นพบโมทีฟความยาวเหมาะสมสำหรับข้อมูลอนุกรมเวลาโดยใช้หลักการเอ็มดีแอล

นายสรชัย ยิ่งเจริญถาวรชัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

PROPER LENGTH MOTIF DISCOVERY FOR TIME SERIES DATA USING MDL PRINCIPLE

Mr. Sorrachai Yingchareonthawornchai

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2012

Thesis Title	PROPER LENGTH MOTIF DISCOVERY FOR TIME SERIES DATA USING MDL PRINCIPLE
By	Mr. Sorrachai Yingchareonthawornchai
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Chotirat Ann Ratanamahatana, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Engineering

(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

THESIS COMMITTEE

.....Chairman

(Professor Boonserm Kijsirikul, Ph.D.)

.....Thesis Advisor

(Assistant Professor Chotirat Ann Ratanamahatana, Ph.D.)

.....Examiner

(Assistant Professor Thanarat Chalidabhongse, Ph.D.)

.....External Examiner

(Songpol Ongwattanakul, Ph.D.)

สรชัย ยิ่งเจริญถาวรชัย : การค้นพบโมทีฟความยาวเหมาะสมสำหรับข้อมูลอนุกรมเวลาโดยใช้หลักการเอ็มดีแอล. (PROPER LENGTH MOTIF DISCOVERY FOR TIME SERIES DATA USING MDL PRINCIPLE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. โชติรัตน์ รัตนามหัทธนะ, 56 หน้า.

การค้นพบโมทีฟสำหรับข้อมูลอนุกรมเวลาเป็นสาขาหนึ่งของงานวิจัยการทำเหมืองข้อมูลอนุกรมเวลาซึ่งได้รับความสนใจเป็นอย่างมากในทศวรรษที่ผ่านมา งานวิจัยด้านการค้นพบโมทีฟที่ผ่านมาประสบความสำเร็จในการค้นพบโมทีฟได้อย่างรวดเร็ว แต่งานวิจัยเหล่านั้นจะต้องกำหนดค่าของพารามิเตอร์ต่างๆ พารามิเตอร์หนึ่งที่สำคัญอย่างยิ่งและยากในการกำจัดคือความยาวโมทีฟที่จะค้นหา เพราะความยาวที่เหมาะสมของโมทีฟนั้นหาได้ยากแม้กระทั่งสำหรับผู้เชี่ยวชาญในสาขานั้นๆ ก็ตาม ถึงแม้ผู้ใช้จะสามารถระบุค่าความยาวโมทีฟมากกว่าหนึ่งค่าได้ การนำโมทีฟที่ได้ในได้ละความยาวมาจัดอันดับเพื่อนำไปใช้งานจริงนั้นเป็นอีกปัญหาที่ท้าทาย เนื่องจากโมทีฟที่ได้อาจเหลื่อมล้ำและซ้ำซ้อนกันได้ และยิ่งกว่านั้นการเลือกทุกความยาวที่เป็นไปได้สำหรับการดำเนินอัลกอริทึมนั้นทำให้การทำงานช้ามาก อย่างไรก็ตามมีงานวิจัยน้อยมากที่ได้กล่าวถึงปัญหาความยาวของโมทีฟและนำเสนออัลกอริทึมในการแก้ปัญหา แต่งานเหล่านั้นไม่สามารถกำจัดพารามิเตอร์ไปได้ทั้งหมด มีอีกงานวิจัยที่สามารถกำจัดพารามิเตอร์ทั้งหมดได้ แต่มีปัญหาทางด้านความสามารถในการปรับขนาดได้ในกรณีข้อมูลที่มีขนาดใหญ่ และคุณภาพของโมทีฟที่ค้นพบ ปัญหาข้างต้นเป็นแรงบันดาลใจของการค้นพบโมทีฟความยาวเหมาะสมที่ปราศจากพารามิเตอร์โดยใช้หลักการเอ็มดีแอล (ความยาวเชิงการพรรณาน้อยสุด) โดยให้ผลลัพธ์สุดท้ายเป็นเซตของ “การบีบอัดโมทีฟ” จากหลักการเอ็มดีแอล โดยมีวิธีการวัดคุณภาพของผลลัพธ์โมทีฟและประสิทธิภาพของอัลกอริทึมที่ชัดเจนตามวัตถุประสงค์อันได้แก่ ความปราศจากพารามิเตอร์ ความถูกต้อง และความเร็ว อัลกอริทึมที่นำเสนอนั้นสามารถค้นพบโมทีฟที่ความยาวเหมาะสมได้โดยมีความแม่นยำสูง ยิ่งกว่านั้นอัลกอริทึมยังสามารถค้นพบโมทีฟที่นอกเหนือความคาดหมายได้ และงานนี้เร็วกว่างานก่อนหน้าเป็นอันดับขนาดเล็กน้อย

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2555

5570504321 : MAJOR COMPUTER ENGINEERING

KEYWORDS : MOTIF DISCOVERY / TIME SERIES / TIME SERIES MOTIF

SORRACHAI YINGCHAREONTHAWORNCHAI : PROPER LENGTH MOTIF DISCOVERY FOR TIME SERIES DATA USING MDL PRINCIPLE. ADVISOR : ASST. PROF. CHOTIRAT RATANAMAHATANA, Ph.D., 56 pp.

As one of the most essential data mining tasks, finding of frequently occurring patterns, i.e., motif discovery, has drawn a lot of attention in the past decade. Despite successes in speedup of motif discovery, most of the existing algorithms still require predefined parameters. The critical and most cumbersome one is time series motif length since it is difficult to manually determine the proper length of the motifs—even for the domain experts. In addition, with variability in the motif lengths, ranking among these motifs becomes another major problem, not to mention possible redundancy of sub-motifs. Ultimately, running all possible ranges of parameter is computationally expensive. There are some attempts to deal with the parameter issue. However, they cannot truly eliminate parameters. There is a work that is considered the first parameter-free motif discovery, but it is not as efficient due to scalability problem and solution quality. These problems inspired this work to introduce a novel parameter-free time series motif discovery based on MDL (Minimum Description Length) principle. This work requires no parameter as input. The output is ranked classes of motifs or "K-compression Motif," that are based on MDL principle. The experiments have been designed to correspond to the objectives: parameter-freeness, correctness, and speed. The proposed algorithm manages to discover proper length of motifs with consistently high accuracy. In addition, the proposed algorithm manages to discover "unexpected" motifs of time series. As for speed, the proposed algorithm is a few magnitudes faster than previous work.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Engineering Advisor's Signature .....

Academic Year : 2012 .....

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Assistant Professor Dr. Chotirat Ratanamahatana, my research advisor, for her patient guidance, encouragement, and useful critiques of this work. Her advises helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor for my M.Eng study.

I would also like to extend my thanks to the rest of my thesis committee: Professor Dr. Boonserm Kijsirikul, Assistant Professor Dr. Thanarat Chalidabhongse, and Dr. Songpol Ongwattanakul for their insightful comments and questions.

To the students at Machine Intelligence and Knowledge Discovery Lab, I am grateful for the intellectual discussions in lab's meetings. I would like to thank my fellow labmates— Navin Madicar, Supawadee Saengsri, Sura Rodpongpun, Haemwaan Sivaraks, and Dr. Vit Niennattrakul for welcoming me as a friend and helping to develop ideas in this thesis.

I greatly appreciate the financial support from Returning Student Scholarship of academic year 2012 given through the Department of Computer Engineering, Chulalongkorn University to drive my research significantly forward. Additionally, I am thankful to the administrative staffs for being supportive during my whole time attending the Department of Computer Engineering.

Finally, I would like to thank my parents and my brother for understanding and supporting me throughout my life.

Contents

List of Tables

Page

List of Figures

CHAPTER I

INTRODUCTION

Time series mining is a branch of data mining; the data itself are temporally ordered. The time series data can be from various domains such as speech recognition, business data analysis, human medical analysis, etc. For example, Figure 1.1 portrays Electrocardiogram (ECG). The figure is the output signal of the electrical activity of the heart over a period of time. The ECG is used for measuring rate and regularity of a human heart. Notice that each data point is ordered chronologically.

There is a variety of tasks of time series mining including Clustering[8][9], etc. Motif Discovery [1][2][5][6][7][10][11][13] is the interest of this work.

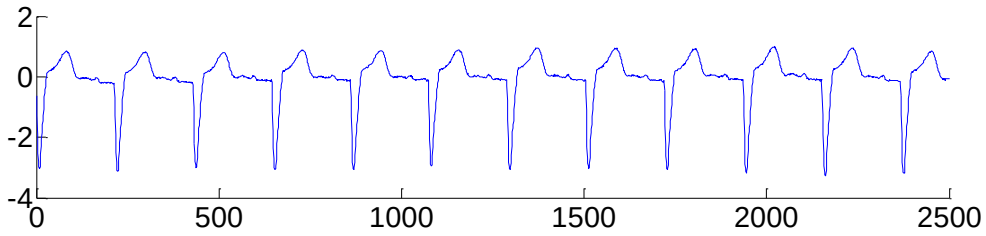

Figure 1.1: An Electrocardiogram (ECG) data. The x-axis represents the time interval. Notice the data has temporal ordering. The time series data are from [14].

1.1 Background and Significance of the Problem

Finding repeated latent patterns in the time series or *Motif Discovery* is the fundamental tasks for the higher level tasks such as Clustering. The domain has been actively explored since the past decade. The problem is generally stated as follows:

Given a time series, what are the frequently occurred patterns?

The answer depends on the definition of the pattern or motif. There are several possible definitions. Some examples of these definitions are given as the following:

- A motif is a pair of subsequences that have the **closest** distance in the time series. [2]

- The *k*-motif is groups of matched subsequences that are ranked by the **number** of elements in the group, i.e., the 1-motif is a group that contains the maximum number of matched subsequences. Matched subsequences are determined by the distance that is less than a real number *R*. [6]

Even if the definitions of motif differ, they are not mutually exclusive. In general, the motif similarity measurement is based on its shape. Some papers focus on the similarity [2] while others focus on frequency [6]. Since the definitions of motifs in existing works assume a length to be discovered is fixed, in variable-length motifs aspect, this work provides a new definition of the motif which considers both similarity and frequency of motifs by ability to compress by its motif. In this regard, MDL (Minimum Description Length) is utilized [8][10][12] as a compression basis.

In order to discover the closest pair of motifs of length *m*, a brute force search can be done with $O(mN^2)$ time complexity, where *N* is a length of time series. A large number of motif discovery algorithms have been developed along with several techniques in order to achieve both speed and quality of the solutions. For instance, Exact Motif Discovery algorithm [2] has proposed an optimization by using triangular inequality to prune off the search space. However, users do still suffer from selecting a set of parameters; an initial window size is a typical one. The existing works require a length of the motif as a parameter for a reason that it is configurable; this is an unrealistic assumption that motif length be predefined by users. The choice of this parameter is sensitive and untenable for users [16] because it is hard to determine the proper length of motif by hands—even for domain experts.

For instance, suppose a user wants to determine the motifs in time series data in Figure 1.2. Since existing motif discovery algorithms require initial length of sliding window, the user needs to speculate length *L* manually or did some trials and errors of

various lengths until satisfaction. In fact, the time series in Figure 1.2 has two classes of motifs: one of length 163 and another of length 286.

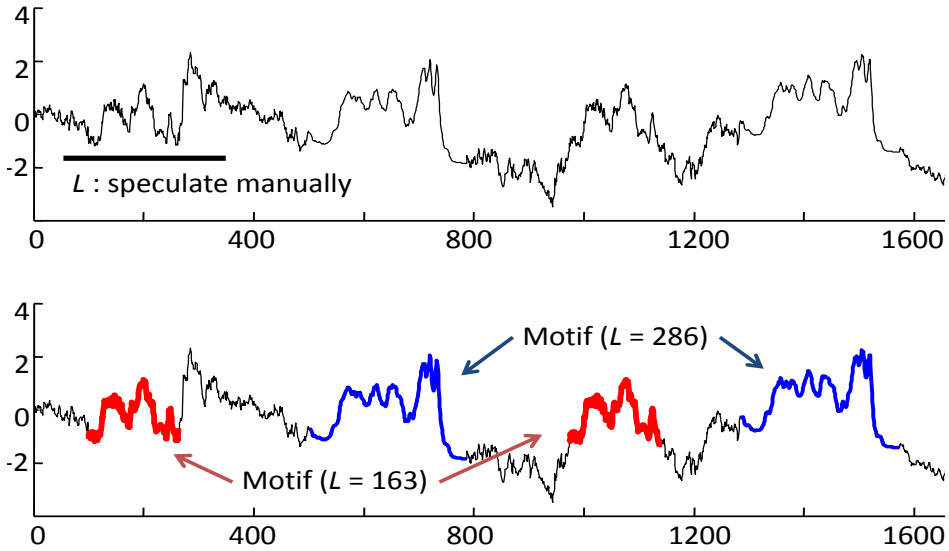

Figure 1.2: A typical time series data. A user needs to speculate a length of motif manually. More often, it is hard to speculate the exact proper lengths of motif as the exact lengths are 163 and 286.

In addition, the problem becomes more subtle if one attempts to exhaustively search for all possible motifs in all possible lengths since the number of discovered motifs will be huge, not to mention overlapping motifs in various lengths. It is also difficult to rank motifs because of their variability in lengths.

1.2 Objectives

To develop a novel motif discovery algorithm that has the following properties:

- *Parameter-freeness*: The algorithm does not have to be specified any parameter.
- *Correctness*: The algorithm is able to find classes of variable-length motifs. The motif class needs to be structurally equivalent, i.e., the subsequences are alike in shape. The Arrcuracy-on-Recall (AoR) and Accuracy-on-Detection (AoD)[15] are used for verifying calibers.
- *Speedup*: The algorithm has speedup over brute-force search. Ideally, all of the extraneous search spaces should be pruned off.

1.3 Scope

1. The experiments will carry on both synthetic and real data from various resources, e.g., UCR time series archives [3] and Physionet [14]. The proposed algorithm is compared to the previous work [7] in terms of efficacy using AoD (Accuracy-on-Detection) and Arrcuracy-on-Recall (AoR), since [7] has addressed the problem in common.

2. The input of proposed algorithm will be only time series data. The output will be the desired motifs with the proper length. All intrinsic parameters, such as the cardinality of discretization, must be trivial, i.e., variations setting of the intrinsic parameters make slight difference in terms of output.

1.4 Benefit

The proposed algorithm manages to discover time series motifs in a reasonable time without indicating any parameters. Users will no longer suffer from selecting a set of parameters to find motif in time series data

1.5 Publication

Sorrachai Yingchareonthawornchai, Haemwaan Sivaraks, Sura Rodpongpun, and Chotirat Ann Ratanamahatana. 2012. The Proper Length Motif Discovery Algorithm. In Proceedings of the 16th International Computer Science and Engineering Conference (ICSEC'12), Pattaya, Chonburi, Thailand.

CHAPTER II

LITERATURE REVIEW

2.1 Concept and Theory

2.1.1 Metric

A metric on a set X is a function (called the distance function or simply distance) $d : X \times X \longrightarrow R$ (where R is a set of real numbers). For all x, y, z in X, this function is required to satisfy the following conditions:

$d(x, y) \geq 0$ (Non-negativity)

$d(x, y) = 0$ if and only if $x = y$ (Identity of Indiscernible)

$d(x, y) = d(y, x)$ (Symmetry)

$d(x, z) \leq d(x, y) + d(y, z)$ (Triangular inequality)

2.1.2 Distance

A distance is a numerical description of how far between two objects are in the space. In mathematics, a distance is a function that is used to measure similarity. There are two popular distance measurement techniques: Euclidean Distance and Dynamic Time Warping.

Euclidean Distance:

Given two objects $X = (x_1, x_2, x_3, \ldots, x_n)$ and $Y = (y_1, y_2, y_{3, \ldots}, y_n)$, the Euclidean Distance, *ED*, is defined as,

$$ED(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

Dynamic Time Warping:

Dynamic Time Warping (DTW) is an algorithm that calculates an optimal warping path between two time series. The optimal path means the least global cost for warping. A warping path is a path through minimal distance matrix or dynamic programming. See Figure 2.1 for illustration.

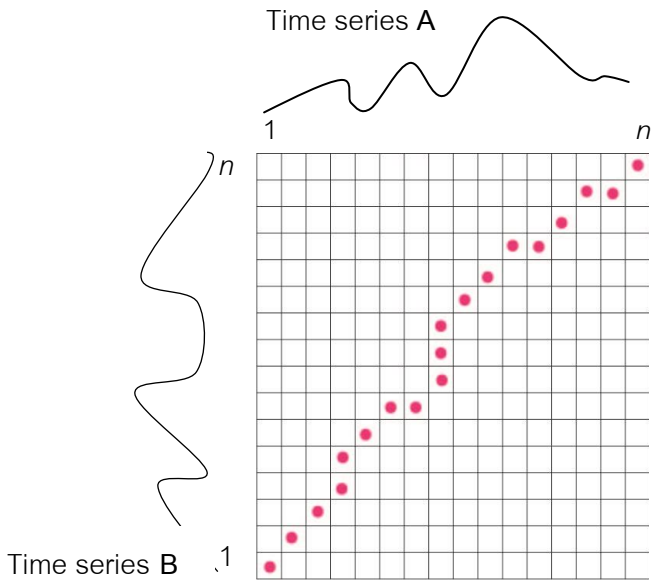

Figure 2.1: The minimal distance matrix which is carried out by dynamic programming. The warping path is denoted as spots.

2.1.3 Information theory

In information theory, Shannon entropy is widely used for defining the lossless compression technique. It quantifies the expected value of the information. In other words, it represents the lower bound of expected value (or average) of number of bits required to perform the lossless data compression on each data point. In this context, we simply call entropy. The entropy of a time series T is defined as:

$$Entropy(T) = \sum_i p_i \log_2(\frac{1}{p_i}) \quad (2.2)$$

where p_i is the probability that symbols of T_i will occur, and $p_i \log_2(\frac{1}{p_i})$ is defined as 0 if p_i = 0.

2.1.4 Anytime Algorithm

Anytime algorithms are the algorithm that can return a valid solution even if it is interrupted, and can resume to continue and update the best-so-far solution toward better solution until the anytime algorithm eventually terminates with the same answer that the batch algorithm would have done[17].

The most desirable properties of anytime algorithms[17] are:

- Interruptibility: While running, the algorithm can be interrupted at any time and outputs a valid or partial solution.
- Monotonicity: The quality of the result is a non-decreasing function of computation time.
- Measurable quality: The quality of an approximated result can be measured.
- Diminishing returns: The progress of solution quality is carried out at the early stage of execution.
- Preemptability: The algorithm can suspend, and resume with minimal overhead.
- Low Overhead: Total time taken by anytime algorithm is comparable to total time taken by batch algorithm.

2.2 Relevant research

The following literatures have definitions of motif in common; here are the definitions, unless the context otherwise defines in the subsection:

Definition 1: **Time Series**: A time series $T = t_1,\ldots,t_m$ is an ordered set of m real-valued variables.

Definition 2: **Subsequence**: Given a time series T of length m, a subsequence C of T is a sampling of length $n < m$ of contiguous position from T, that is, $C = t_p,\ldots,t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.

2.2.1 Finding Motifs in Time Series [6]

This work introduces a meaningful definition of time series motif and proposes an approximate motif discovery algorithm which is called Enumeration of Motifs through Matrix Approximation (EMMA).

Here are the non-trivial definitions:

Definition 1: **Match**: Given a positive real number *R* (called range) and a time series *T* containing a subsequence *C* beginning at position *p* and a subsequence *M* beginning at *q*, if Distance*(C, M)* $\leq$ *R,* then *M* is called a matching subsequence of *C*.

Definition 2: **K-Motifs**: Given a time series *T*, a subsequence length *n* and a range *R*, the most significant motif in *T* (called thereafter 1-Motif) is the subsequence C_1 that has the highest count of non-trivial matches (ties are broken by choosing the motif whose matches have the lower variance). The K^{th} most significant motif in *T* (called thereafter *K*-Motif) is the subsequence C_K that has the highest count of non-trivial matches, and satisfies $D(C_K, C_i) > 2R$, for all $1 \leq i < K$.

The paper presents a novel discrete representation that allows both dimensionality reduction and a definition of a lower bounding distance measure. The representation is carried on by using Piecewise Aggregate Approximation (PAA) as an intermediate step between the time series and desired representation. Then, the PAA representation is transformed to a discrete representation. The idea is the utilization of Gaussian distribution to define breakpoints to produce symbols with equiprobability. The distance measure of symbolic representation is defined as MINDIST function which is structurally the same as lower bounding approximation of the Euclidean distance using PAA representation except for the fact that the distance between the two PAA coefficients has been replaced with the function dist(). The dist() function is defined as a lookup table based on the area under Gaussian curve.

The intuition behind the algorithm consists of two steps. First, it uses the discrete representation to create a smaller matrix, which is guaranteed to contain all the subsequences which are necessary candidates for a motif. Then, the speedup is achieved by using Shasha and Wang's Approximation Distance Map (ADM) algorithm [6] to prune away an enormous fraction of the search space within the small matrix. The experiments are based on efficiency of the algorithm compared with a brute force manner.

This work utilizes the dimensionality reduction concept to find time series motif—achieving speedup over brute force manner. However, a lot of parameters should be

predefined. Their Find-1-Motif-Index(*T*,*n*,*R*,*w*,*a*) algorithm, has four parameters: *n* is window length, *R* is a range for finding *K-Motifs*, *w* and *a* are used for dimensionality reduction. The reason why it is arduous to set parameters is that the more parameters there are, the more choices there could be. According to simple counting techniques, the tuple (*n*,*R*,*w*,*a*) has scads of elements. In practice, users do suffer from selecting seemingly good parameters for specific tasks. They cannot even know the optimal parameter to get a good caliber of desired answer. Therefore, the algorithm for discovering motif with fewer parameters is highly demanded.

2.2.2 Exact Discovery of Time Series Motifs [2]

There have been a myriad of approximate algorithms to discover motifs in the literature. This work proposes a tractable exact algorithm to find time series motifs. In addition, the exact algorithm is fast enough to be used as a subroutine in higher level data mining algorithms. To begin with, definitions of the terms are required:

Definition 1: A **Time Series Database** (*D*) is an unordered set of *m* time series, possibly of different lengths.

Definition 2: The **Time Series Motif** of a time series database *D* is an unordered pair of time series $\{T_i, T_j\}$ in *D* which is the most similar among all possible pairs. More formally, $\forall$ *a*,*b*,*i*,*j* the pair $\{T_i, T_j\}$ is the motif iff $\text{dist}(T_i, T_j) \leq \text{dist}(T_a, T_b)$, $i \neq j$ and $a \neq b$.

The distance measure used in this context is Euclidean distance. The algorithm utilizes two notions: early abandoning and triangular inequality with additional optimization.

The early abandoning is a technique that reduces Euclidean distance calculation cost. A cumulative variable is being kept for each point. Given a best-so-far value, if the cumulative variable is larger than the *best-so-far*, the whole distance must then be larger than the *best-so-far*. Therefore, we can abandon the query. Figure 2.2 portrays the visualization[2], assuming the *best-so-far* is 144.

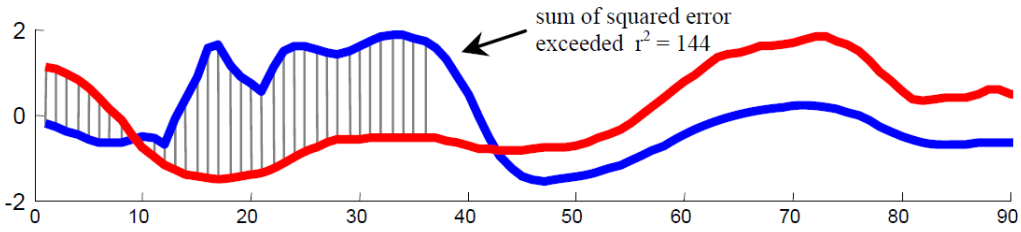

Figure 2.2 : A visualization of early abandoning. When the squared sum exceeds r^2, it is known that the full distance exceeds *r* as well. *r* is *best-so-far*.

The triangular inequality with additional optimization is used in this work. If the number of subsequences are *N*, possible pairs are $N(N\text{-}1)/2$ which is $O(N^2)$. An arbitrary reference point is used to ease an exhaustive search. The other objects are then ordered by their distances to the reference point. In this manner, we can record the distance between adjacent pairs by subtracting them directly using reference point. This action gives heuristic information. In the next step, the search of the closet pairs begins using the arranged linear ordering of remaining objects. The process of pruning away a large fraction of search occurs because we begin the search by the smallest lower bound distance. If the *best-so-far* value is not updated within the offset, it is known that the rest cannot update the best-so-far either. Therefore, the rest are abandoned, resulting in a pair of motif. The additional optimization is carried on by using multiple reference points to strengthen tightness of the lower bound.

This work introduces a good celerity of method to find a pair of time series motif using triangular inequality. Even if this algorithm has reduced parameters to a single *w*, it is still burdensome to use since such information is unavailable to users. For complex and long time series data, it is extremely difficult to determine the proper length of motif to be discovered because intrinsic motifs are invisible to a user. If a user wants to find other motifs in different lengths, they themselves must choose another window size. In addition to the parameter issue, the algorithm focuses on finding the “smallest” distance of the pair of subsequences. Hence, the algorithm for finding the proper length of motifs without parameter is highly beneficial to users and in demand.

2.2.3 Parameter-Free Motif Discovery for Time Series Data [7]

Most of this works focus on time complexity of motif discovery in time series data. However, the problem of predefined parameters could be an untenable task for naïve users. This work proposes a novel algorithm for finding proper motifs based on ranking score function.

Definition 1: **Time Series Motif** M_w is a pair of similar subsequences in T with a specified sliding window length (motif length) w, defined as $M_w = (S_{L_1}^w; S_{L_2}^w; MDist; NDist)$ where $L_1$and L_2 are the starting locations of subsequences in time series T, $L_1 < L_2$, $MDist = EU(S_{L_1}^w, S_{L_2}^w)$, and $NDist$ is a normalized Euclidean distance between subsequences $S_{L_1}^w$ and $S_{L_2}^w$ calculated by $NDist = MDist/w$.

Definition 2: ***k*th-Motif** (k^{th}-M_w), that is used in the proposed scoring function to discover "Best Motif", is a motif that has the k^{th} similar pair of subsequences in T with a specified fixed sliding window length (motif length) w, where k^{th}-Motif and i^{th}-Motif are not overlapped and $1 \leq i \leq k$.

Definition 3: **Best Motif** is a pair of the longest similar subsequences (S_i^w ; S_j^w) that is discovered in some locations where EUC(S_i^w ; S_j^w) $\leq R$, and R is a maximum Euclidean distance.

Definition 4: ***k*th-Best Motif** (k^{th}-BM) is a motif that has the k^{th} highest score from the scoring function that is based on number of similar-location motifs and similarity of subsequence pairs.

There are four steps in this literature. First, the algorithm begins by find all possible motifs ranged from two to half of time series size. The MK [2] algorithm is used as a subroutine to quickly find each motif. Second, each motif is grouped into Motif Groups (MG) under two conditions: they must overlap within group and a threshold for pruning too short motif. The next step, each group needs to find a representative. This can be done by finding a motif that has minimum normalized Euclidean distance in its group. Then, the high distance group will be pruned off by the threshold of median representative of all groups. Finally, remaining groups are scored by proposed scoring

function to find k^{th}-Best Motif. The experiments yielded a better Accuracy-on-Detection (AoD) than the previous work.

This work provides a ranking function, in the algorithm, for variable lengths of motifs to address the aforementioned problems, i.e., parameter free. However, there are flaws in the algorithm: it is not genuinely parameter free. The selection system is jaundiced; it is hard to optimize and the experiments were carried out on the synthetic data. Here are the elaborations. Even if the algorithm claimed that it is a non-parametric algorithm, it contains latent parameters: the thresholds of pruning off the small and big fractions are indeed intrinsic parameters since the solution qualities depend on these thresholds. Also, there is no proof of optimality of these thresholds in the literature; no guarantee of the all motif will be discovered. Therefore, by performing unfair cutting the fraction of motif, it is highly plausible to prune off the potential candidate as well—leading to poor quality of discovered motifs. Even if this work uses MK [2] as a sub function, the task is still laborious because they run on virtually all possible sizes

2.2.4 Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data [8]

This work gives novel definitions for time series clustering from streams. Also, the Minimum Description Length (MDL) framework is shown to be an efficient and effective method for time series clustering.

In this context, the entropy is used and generalized to the Description Length of clusters. There are three operators for clustering: creating, adding and merging. The criteria, which judge whether a subsequence should be joined into clusters, are carried out by MDL principle. The algorithm begins with finding the best initial pair of subsequences to combine to create a new cluster. In the subsequent step, there are additional choices, i.e., create or add. We can either add a subsequence to the existing cluster or create a new cluster. For the third step, there are three options: create, add, and merge cluster. At this stage, the algorithm terminates when, first, data are used up or, second, the other three options cannot make a lower description length of cluster.

Even if this work is a clustering task, some concepts are applicable to motif discovery such as MDL principle. MDL principle has been variously utilized for decades. The ranking issues can be solved by using MDL framework.

2.2.5 Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle [10]

This work has basically two phases. The first phase is to reduce dimensionality of time series data by utilizing Principal Component Analysis (PCA) into one dimensional time series data. Then, the motif discovery is executed on the reduced form of time series by using MDL principle as a criterion. The most significant motif is based on the model that has the minimum description length of data. This work introduces a load of definition of description length to describe the time series data. The experiments are carried out by using motion capture datasets. This work shows that the PCA can eliminate irrelevant feature of the data. Also, the motif discovered from the dataset is the model given by MDL principle.

This work has shown that MDL-based motif is a good alternative to find motif other than being frequent or close enough. However, this work contains a lot of parameters as they mentioned in the open problems section. Several parameters include a number of segments, unique SAX symbols, and the threshold of distance *R*. They also mentioned that the choice of segment of PAA representation has a big influence on motif discovery. Hence, parameter-free motif discovery algorithm is a big demand.

CHAPTER III

PROPER LENGTH TIME SERIES MOTIF DISCOVERY

One of the most confusing terminologies in motif discovery is the definition of motifs per se. As described earlier, there are various definition motifs based on purpose of applications. Fortunately, the basis unit of definition can be bifurcated into two orthogonal perspectives: similarity basis and frequency basis (refer to Section 1.1 for details). Hence, the discussion of motif definitions is based on those definitions.

This work has two fundamental contributions. First, this work proposes alternative definition of variable-length motifs that allows motif to have more than two occurrences (more than a pair of motif). In addition, there are myths about parameter-freeness. This work clarifies the myths in the principle level. Some myths will be clarified in the experiment section. Second, apart from the proposed motif definition and parameter-free principle, this work introduces the proper length motif discovery algorithm for time series data. The algorithm mainly consists of searching for closest motifs [2] in a given length and ranking variable-length motifs via MDL principle [8][10]. In order to achieve parameter-freeness, additional methods are required. The main theme is that algorithm should be executed on only-needed basis to minimize running time. The algorithm utilizes heuristic information from MDL scoring function for reducing running time in practice—giving a huge advantage as shown in the next chapter.

This chapter is organized as follow. First, in order to promote mutual understanding of *keywords* in this work, the definitions and notations are necessary. As MDL principle is a cornerstone to this algorithm, the illustration is included. The realization of MDL principle is carried out by using information entropy. Next, the intuition of the algorithm is explored. This section introduces an overview of the algorithm. Another important component, lower bounding of Euclidean distance, is also explicated in brief details. In addition, more techniques are incorporated in order to achieve goals— being parameter-free while maintaining speed and accuracy. Finally, the algorithm in details including pseudo code is presented line-by-line.

3.1 The parameter-free myths.

This section presents common pitfalls about parameter-freeness.

Myth I: *There is no such parameter-free algorithm because every choice is a parameter.*

This statement has two meanings depending on a meaning of parameter-freeness. The first meaning of parameter-freeness is a technical view such that every possible choice including hard-coding or threshold is *literally* a parameter. This meaning is too rigid and has no practical use. Most importantly, this meaning obfuscates users to misinterpret a useful meaning of parameter-freeness. The other meaning of parameter-freeness is more apropos: there is no non-trivial parameter in the algorithm. Non-trivial parameter is best illustrated with obvious useful parameter-free algorithm like Merge sort algorithm. Merge sort algorithm does have a trivial parameter: selecting a size of sub-problems. The common approach is to divide into equal sub-problems and recursively compute. One may questions why not divide into arbitrary non-equal size despite being a parameter? The answer in parameter-free aspect is that it is a trivial parameter. Even if it slightly affects the performance, virtually all of the choices make Merge sort asymptotically the same. That is, in practical point of view, Merge sort is *parameter-free* despite the fact that it is configurable. Therefore, this work *follows* the second meaning of parameter-freeness: there is no non-trivial parameter in the algorithm.

For these two myths, the arguments are presented. However, the experimental evidences will be presented in Chapter IV.

Myth II: *parameter-free algorithm's correctness is sacrificed in exchange for parameter-freeness.*

The output of a parameter-free algorithm is generally an excessive set. The challenge of to be an effective parameter-free algorithm is how to select answers correctly as a satisfied answers with these abundant answers. It is sometimes difficult to

determine an appropriate subset as satisfied answers. However, to resolve this problem, this work introduces the basic notion of "minimal superset" of answers. Furthermore, the parameter-free version's accuracy will not deteriorate with respect to that of non-parameter-free version.

Myth III: *parameter-free algorithm is always not faster than non-parameter-free algorithm.*

In a common sense, making an algorithm parameter-free is a matter of running a sufficient range condition of a parameter. It is easy to believe that it exhaustively searches all possible instances including extraneous spaces. However, that is not always the case. With an efficient design of a parameter-free algorithm, there exists a case that a parameter-free algorithm utilizes some benefits or heuristic information during *run* time, which is impossible to do by hand or *static* time. Thereby, parameter-free algorithm can inherently outperform non-parameter-free algorithm. The correct counterpart is "parameter-free algorithm is not always faster than non-parameter-free algorithm."

3.2 Terms and Definitions

Definition 1: A time series T of length n is an ordered set of real numbers of a sequence $t_1, t_2, t_3, \ldots, t_n$.

This work concerns about finding subsequences or patterns of the time series

Definition 2: A subsequence C of length m of the time series T of length n is a subset of an ordered sequence $t_1, t_2, t_3, \ldots, t_n$ with the consecutive position $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$. where $1 \leq i \leq n - m + 1$.

In order to obtain subsequences, time series is extracted by sliding windows.

Definition 3: The sliding windows of length w are all possible subsequences of the time series extracted by a window of length w.

For each sliding window, it is possible that an overlapping pair of motifs may occur. This kind of overlapping pair is called a trivial match.

Definition 4: A pair of the subsequence of length *w* is trivial when at least a single pair of positions is overlapped.

Definition 5: Given a window length *L*, k^{th}-*Motif Candidate* (called thereafter MC_L^k) is the k^{th} closest distance of a non-trivial pair of subsequence of length *L* in time series *T*.

Before giving a definition of motifs, the notion of MDL principle is a prerequisite. Here is a basic intuition of compression techniques. The original time series *T* has a bit-level representation whose length or description length is defined as a function *DL*(*T*). Our algorithm works as attempting to losslessly compress the data by finding approximately repeated structures or motifs and encoding the differences of the hypothesis. It is sufficient to consider a motif as a model or hypothesis, *H*, to encode the original time series *T* only its neighbors with the hypothesis. After encoding with the hypothesis, the size of the remaining data, *DL*(*T* | *H*), is reduced by factoring out the common structure of hypothesis. The hypothesis has size in bits as *DL*(*H*). The total cost is *DL*(*T*|*H*) + *DL*(*H*).

At this stage, there are two observations. First, when motifs of length *L*, as a hypothesis, compress their common structures, the quality of the compression depends on the similarity between hypothesis and its occurrences: the more similar they are, the more ability they can compress. Likewise, when the time series contains a number of motifs sharing the same structure, the quality of compression also depends on the frequency of the occurrences having the same structure as the hypothesis: the more frequent motifs are, the more ability they can compress. Therefore, two observations imply that a good motif is a model that has both similarity and frequency properties. In other words, the best model out of a model class is the model whose the encoding cost for both data and model combined is the lowest. Figure 3.1 portrays two obvious different compressions between good and bad compression. The gist is that abstraction of a good motif can be realized by applying the MDL principle.

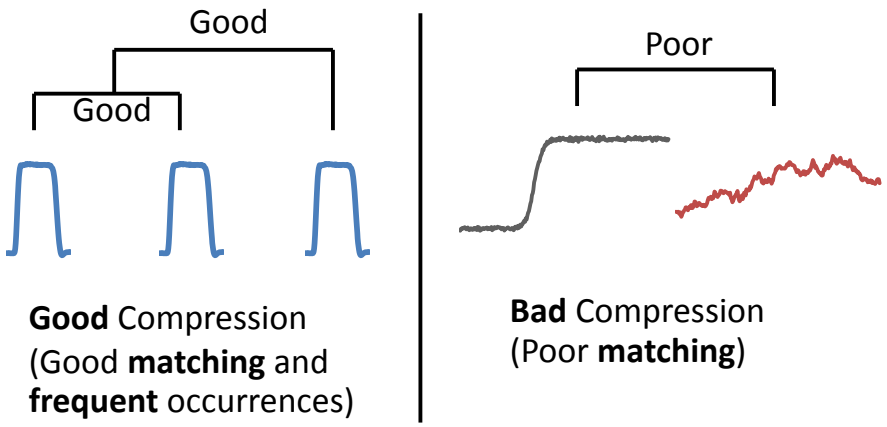

Figure 3.1 : An example of two obvious cases. A good motif class is a motif that well matches to its neighbors and has lots of neighbors. In contrast, a bad motif class is a motif that poorly matches and has few neighbors. In this abstraction, we measure goodness by ability to compress.

Example:

Given a subsequence *A* of length 64, the subsequence *H* is the most felicitous hypothesis *H* for compressing a subsequence *A* (Figure 3.2). Note that the *entropy* is lossless data compression. Therefore, both hypothesis *H* and a subsequence *A-H* are needed to be stored because the subsequence *A-H* requires hypothesis *H* to reconstruct to be an original subsequence, i.e., a subsequence *A.* Next, the description lengths of each subsequence are computed. The entropy of *A* is *Entropy*(*A*) = 4.779, and description length of *A* denoted as *DL*(*A*) = 64*4.779 = 305.861. The entropy of *A-H* is *Entropy*(*A-H*) = 2.048, and description length of *A-H* denoted as *DL*(*A-H*) = 64*2.048 = 131.042. The hypothesis *H* is the special case because it is merely two connected lines. Three points are required to store two connected lines. Each point requires one byte. Thus, description length of *H* denoted as *DL*(*H*) = 3*8 = 24. Here is the summary.

$$DL(A) = 305.861 \text{ vs.} \tag{3.1}$$

$$DL(A\text{-}H) + DL(H) = 131.042 + 24 = 155.042 \tag{3.2}$$

This saves enormous amount of bits by using an appropriate hypothesis. In fact, it is easy to use a representative from a pair of motif as a hypothesis in time series data.

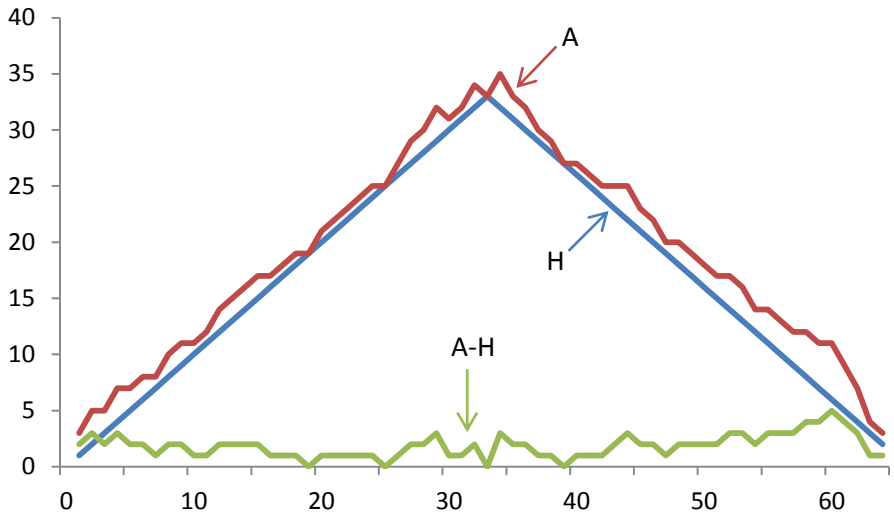

Figure 3.2: A 64-length subsequence. The top-most line represents a typical subsequence extracted from the time series, denoted as *A*. The middle line represents hypothesis *H*. The bottom line represents *A-H*.

As discussed in chapter 1, the definition of motifs vary – depending on usage in a domain of application. Previous work [2][6][13] has an assumption that length of motif is predefined by a user, so the definition of motifs holds the same assumption. As for variable length of motif discovery, parameter-free motif discovery [7] has first defines K^{th} best motifs. K^{th} best motifs are based on location-and-similarity supported motifs. However, it does not support the notion of both similarity and frequency of motif occurrences. As this work concerns similarity and frequency of motifs occurrences, it uses compression basis to define variable-length motifs.

Definition 6: K-Compression Motif: Given a time series T, the most significant motif in *T* (called thereafter the 1-Compression Motif) is the subsequence or hypothesis, having at least two occurrences, whose the encoding cost for its neighbor non-trivial subsequences encoded with hypothesis and the hypothesis is the lowest. The K^{th} most significant motif in T is K^{th}-class of subsequences having at least two occurrences that has the K^{th}-lowest encoding cost for both data and hypothesis.

Definition 6 implies that to be an interesting motif, a subsequence must well compress among its neighbors. The more similar they are, the higher compression rate it is. Also, the more frequently they occur, the higher compression rate it gains.

Having described intuitions of compression techniques and time series motifs in this work, now it is worth exploring the quantized version of "ability to compress time series by motifs." This can be realized as MDL principle. Note that MDL principle is well utilized in many time series tasks, Definitions 7 to 14 follow those from [8] and described here for readability.

Time series can have various offsets and amplitudes. In maintaining its shape, normalization is necessary as it helps maintaining meaningful results [8].

Definition 7. A discrete normalization function is a function to discretize the real value of time series into *a*-bit discrete values of range [1, 2^a] as follows,

$$DiscreteNorm(T) := round(\frac{T - \min}{\max - \min}) * (2^a - 1) + 1 \quad (3.3)$$

where min and max are the minimum and the maximum value of subsequence T, respectively. According to [8], a = 64 for the rest of this work.

Euclidean distance between two time series is used for similarity comparison.

Definition 8: The distance between two subsequences is defined as the Euclidean distance, given by

$$Dist(T_1, T_2) := \sqrt{\sum_{i=0}^{L} (t_{1,i} - t_{2,i})^2} \quad (3.4)$$

. As for description length, Huffman code can be used [12]. However, information entropy is more simple and intuitive for describing description length. It also provides a good lower bound for representing bits [8].

Definition 9: The entropy of a time series T is defined as:

$$Entropy(T) := \sum_i p_i \log_2(\frac{1}{p_i}) \quad (3.5)$$

where p_i is the probability that a symbol in T_i will occur.

Notice that the time series is already discretized (Definition 8) before calculating the entropy. Next, the description length (DL) of the time series is defined as,

Definition 10: The description length DL of time series *T* of length *m* is the total number bits required to represent itself, given as

$$DL(T) := m * Entropy(T) \tag{3.6}$$

Definition 11: A *hypothesis H* is a subsequence used to encode one or more subsequences with the same length.

Definition 12: The *Conditional Description Length* of time series *T* with a hypothesis *H*, *DL*(*T* | *H*) is equal to *DL*(*T* – *H*) where *T* – *H* is *T* encoded with *H*, and *T* – *H* represents a minus vector operation.

Definition 13: A *Description Length of a Class* (*DLC*) C is the number of bits required to represent all subsequences in the group.

$$DLC(C) := DL(H) + \sum_{A \in C} DL(A \mid H) \tag{3.7}$$

The center of the group is hypothesis H obtained by the average of all members in the group.

The original time series data may be huge. Compressing the whole time series yields unnecessary works. Another way to measure ability to compress is to measure by *bitsave* after each operation, i.e., the value of description length of before-compression minus after-compression, compressing a time series with a hypothesis.

Definition 14: A *bitsave* is the number of bits saved after applying an operation. It can be calculated from the difference of the description length, before and after, i.e.,

$$bitsave := DL(old) - DL(new) \quad (3.8)$$

This work uses two operations: creating a motif class and adding a member to a motif class. They are defined as follows,

Creating a new group C' from subsequence A and B:

$$bitsave := DL(A) + DL(B) - DLC(C') \quad (3.9)$$

Adding a subsequence A to an existing group C:

$$bitsave := DL(A) + DLG(C) - DLC(C') \quad (3.10)$$

Finally, measuring an "ability to compress" is done by calculating *bitsave* from the difference of a description length in each operation. (Note that the terms MDL and *bitsave* are interchangeable)

3.3 Intuitions behind algorithm

After providing background and notation, this section introduces intuitions behind motif discovery algorithm. The overview of the algorithm is portrayed in Figure 3.3. The algorithm starts from the smallest to the largest possible of sliding windows. Suppose the algorithm is running at length *L*, a question is how to find hypotheses from the raw time series? Note that definition 6 sets a constraint that a hypothesis should occur at least twice. The first clue is to search for the closest pair of subsequence of length *L*, which is a good candidate for constructing a hypothesis to compress the time series data. The second step is to compute MDL of the pair. At step 3, it is possible that there are more than two occurrences of the hypothesis. The other occurrences can be iteratively found via the next nearest neighbor of the hypothesis, i.e., closest subsequence matching. Step 3 is repeated until the last neighbor cannot compress well enough, or new MDL is not better than the previous MDL. At step 6, the answer is updated by this group. The answer-updated operation will be clarified in the next

section. Finally, step 1 is repeated until k^{th} closest pair cannot compress anymore (cf. Defintion 14).

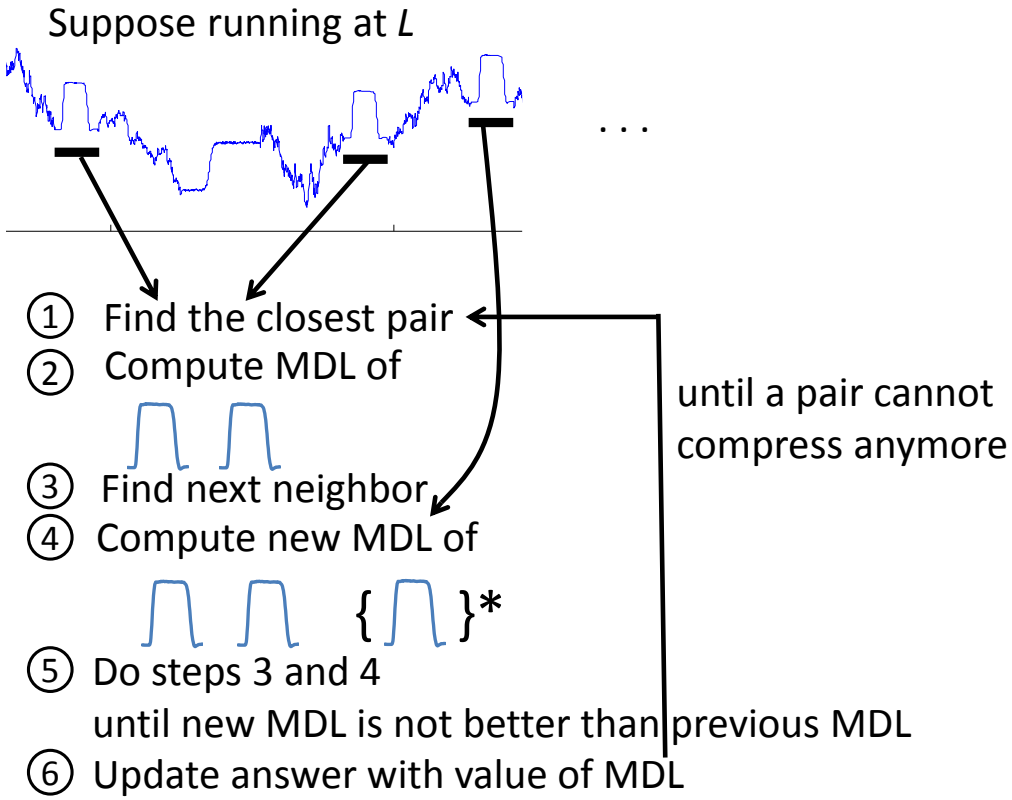

Figure 3.3 :The general framework of the algorithm.

3.3.1 Compression by MDL principle

After encoding time series data with a hypothesis, the size of remaining data is reduced by factoring out the common structures or motifs. The compression rate is calculated by measuring *bitsave* for each operation (Definition 14).

The method of *bitsave* calculation is similar to [8]. First, a group is created from the closest pair. After group creation, *bitsave* is computed by the difference of number of bits between before and after an operation. Next, all neighbors of the hypothesis (or a model) are to join group until either neighbors are used up or new *bitsave* is less than zero.

3.3.2 Lower bounding

Previous section explains how to calculate *bitsave* for each group operation. Note that the method of finding the closest pair is not yet mentioned. This section will explain how to quickly find the closest pair by utilizing a notion called “lower bound.” This work follows the same techniques as [2] including techniques called early abandoning.

Given a subsequence of length *m*, comparing all pairs using Euclidean distance to discover the closet pair takes O(*mn*(*n*-1)/2) comparisons, where *n* is length of time series subtracted by *m*. One way to mitigate this complexity is to use triangular inequality of Euclidean Distance as heuristic information for pruning off extraneous search space. This notion is well utilized in [2] called early abandoning and lower bound. Even though the worst case running time is still asymptotically the same as brute force method, the running time in practice is much faster [2].

Here is a brief description. The key approach is to randomly select a sliding window as a pivot. Then, calculate distance of all sliding windows, and sort the other sliding windows by their distance to the pivot. This linear ordering of the data has useful heuristic information, as shown in Figure 3.4 Notice that the result of the sorted data has its own “lower bound” distance between adjacent pair. Therefore, the search will begin by using the *best-so-far* to enhance the speed. To illustrate, consider this example,

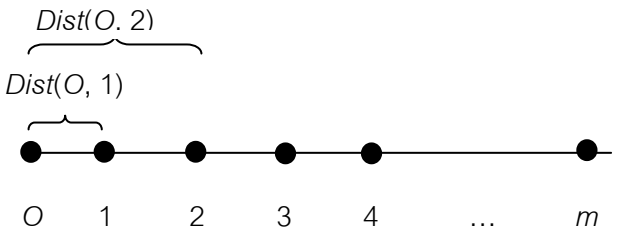

Figure 3.4: Linear ordering of the data. *O* is a pivot.

In Figure 3.4, *O* is a pivot. The other subsequences in sliding windows are calculated by distance which used pivot as a reference. So, the ordering are *Dist*(*O*, 1), *Dist*(*O*, 2), *Dist*(*O*, 3) …, *Dist*(*O*, *m*). Without loss of generality, positions 1 and 2 are a point of interest as illustrated in the example. the lower bound distance between

position 1 and position 2 can be determined by using the triangular inequality property as follows.

$$Dist(O,1) - Dist(O,2) \leq Dist(1,2) \quad (3.11)$$

The lower bound distance can be calculated in a constant time. In order to find the smallest distance of a pair, it starts by setting offset variable to 1, and scans from left to right until the end of this ordering. The offset is incremented when there exists an update on *best-so-far* variable and scan from left to right again until there is no update on best-so-far value in an offset. By scanning from left to right and broadening the offset, it is possible to have all $\frac{m(m-1)}{2}$ pairs in this search [2].

In short, for each offset, the *best-so-far* value is updated when there is a pair of its true distance which is less than current *best-so-far*. If the *best-so-far* value is not updated within all pairs in same offset, it is easy to see that the rest of the unsearched pairs will not be able to update the best-so-far value as their lower bound distance would exceed the *best-so-far*. Therefore, all unsearched pairs can be discarded. This would save enormous amount of computation time.

However, running on all possible lengths is computationally expensive even if the lower bounding is utilized. This work incorporates two additional techniques to tackle this difficulty: designing to be an anytime algorithm via updating answer by ability to compress and early termination.

3.3.3 Updating answer by ability to compress (*bitsave*)

The algorithm initializes a set of answer as an empty set. While running, the answer is updated or added by a new entry. Each answer or entry is represented as a tuple, <*L*, <*positions*>, *bitsave*>, where *L*, *positions*, *bitsave* are length of subsequences, vector of position of occurrences in time series, and ability to compress respectively. For example, a tuple <50, <100,500>, 80> represents a pair of subsequences of length 50 occurred at position 100 and 500 with the bitsave value of 80.

Figure 3.5 portrays the mechanism of answer update. Suppose currently there are two entries in the answer set, a new entry has length of 150, three elements and *bitsave* value of 365. As in definition 6, the interesting motifs are *hypotheses* that have more ability to compress or more *bitsave*. Therefore, updating method is forthright. For each answer in answer set that overlapped with the new entry, *bitsave* value is a criterion for replacing the new entry if its *bitsave* is higher. If the overlapped entry does not exist, the new entry will be added in the answer set.

The overlapping of two entries is occurred when two of the following are met,

1. The number of motifs in entry is equal.
2. There exists one-on-one overlapping pair of each element in the entry.

As you can see, the answer set is being updated while running. This meets any-time algorithm property; as time pass the solution will evolve toward optimal solutions. Hence, the proposed algorithm is an anytime algorithm.

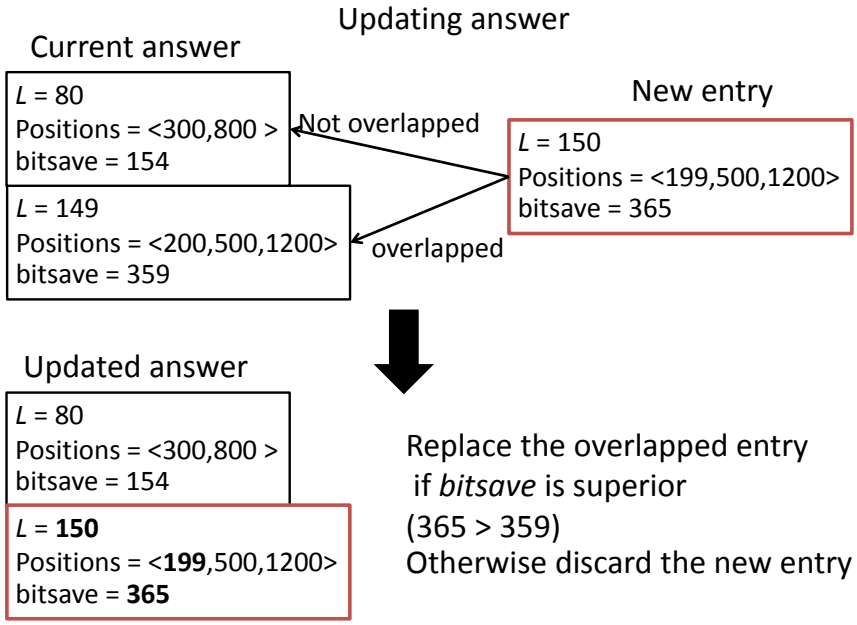

Figure 3.5 : The example of updating answer set. the new entry is regarded as a classes or group of motifs from previous steps. First, we locate the overlapped element. If overlapping element does not exist, we add new entry in answer set. Otherwise, replace the new entry with overlapped entry when bitsave is greater.

The following example illustrates how the update operation manages to discover *proper* length motifs.

Example (Dataset in Figure 1.1 revisited):

Consider Figure 3.6, the motifs are two highlighted subsequences. The motif of length 286 is an object of interest. Since the algorithm discover each length in an increasing order, roughly two pairs labeled as “too small” are discovered first. Then, these two pairs are inserted into the answer set as new entries because there is no entry yet. While running to next length, the bigger size ones of the same location are discovered and updated again and again. This iteration continues until the optimal one is reached whose compression ratio is the best, as denoted “optimal.” When the bigger pair labeled as “too large” comes to update, it has lower compression ratio or *bitsave* than the optimal one. This motif will never be updated with the larger length since the larger length one contains *noises* or *irrelevant* part as denoted in left part of the larger pair. This example illustrates a case when *proper* length motifs are discovered. This is also true in general case.

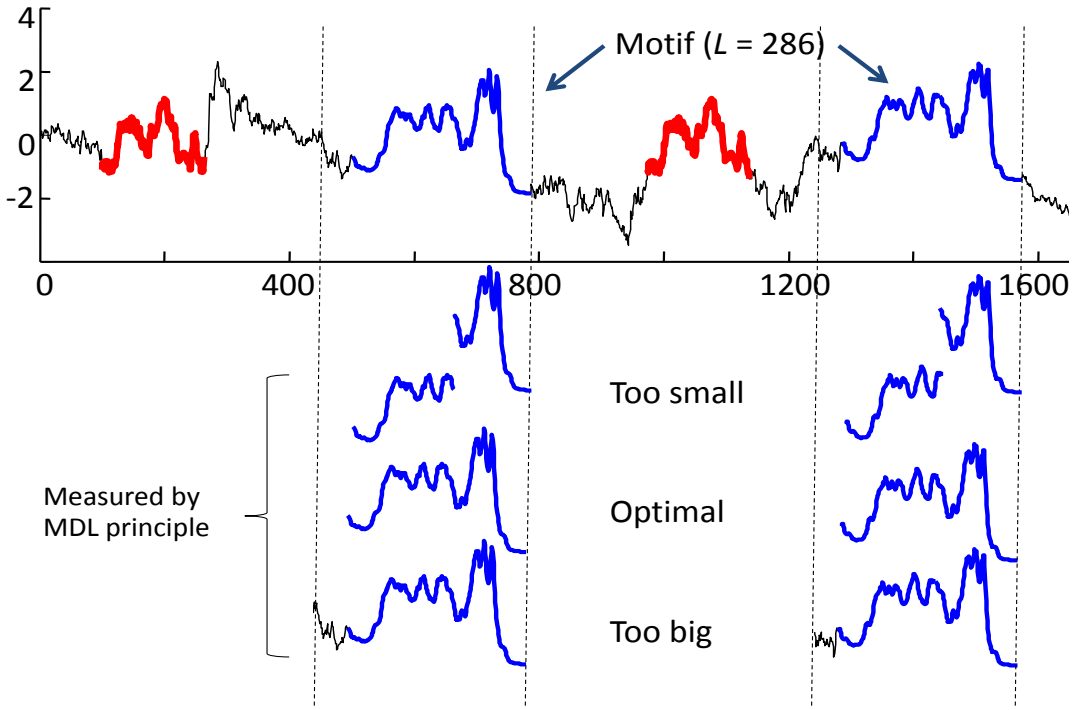

Figure 3.6 : Update operation – example.

3.3.4 Early Termination

Fixed a motif of interest, some readers may notice that, while updating toward proper/optimal length of motif, the larger length of that motif cannot be compressed well as the proper/optimal length does. This is also happening for any other motifs of arbitrary lengths. When consider over all motifs in a time series data, it is possible that, for some larger lengths of sliding windows, all subsequences are either unable to compress well or cannot update any answer (since it is labeled as "Too big" as in Figure 3.6). This condition is for halting the algorithm since running on larger lengths yields no new results. In short, when this condition is met, running algorithm can be terminated *without* losing any potential motifs. This action can prune off extraneous search space and thus save an enormous amount of time. This notion is called "Early Termination."

Formally, at length *L*, suppose k_0 is the first integer that *bitsave* of creating group of $MC_L^{k_0}$ is negative. If, for $k < k_0$, MC_L^k cannot update or add any new entity to answer set, then all of motifs have been discovered and the algorithm will halt with non-fault dismissals.

The following lemma provides performance guarantee for Early Termination.

Lemma1: Given that patterns have the identical shape, if the time series *T* contains a largest pair of pattern of length *M*, the algorithm will terminate at least at length *M*+1.

Proof: The algorithm triggers Early Termination when there is no update in the answer set. Therefore, it is sufficient to focus only the largest pair of patterns in the dataset. In order to prove this lemma, it is imperative to show that the *bitsave* of the largest patterns is always higher than that of smaller sub-patterns.

Suppose there are pattern p_1,p_2 of length *M*, the description length of a motif class of p_1 and p_2 is $DL(p_1) + DL(p_2)$. When encoding with hypothesis H_M = average of p_1 and p_2, the new description length becomes,

$$DLC(C_M) = DL(H_M) + DL(p_1 \mid H_M) + DL(p_2 \mid H_M) \quad (3.12)$$

The *bitsave* is calculated as the difference of description length between before and after encoding with H_M.

$$Bitsave(C_M) = DL(p_{1,M}) + DL(p_{2,M}) - DLC(p_{1,M}, p_{2,M}) \quad (3.13)$$
$$= M(Ent(p_{1,M}) + Ent(p_{2,M}) - Ent(H_M) - Ent(p_{1,M|} \mid H_M) - Ent(p_{2,M} \mid H_M))$$

Similarly, for other smaller sub-patterns of length m < M, *bitsave* of this sub-pattern class of $p_{1,m}$ and $p_{2,m}$ is

$$Bitsave(C_m) = DL(p_{1,m}) + DL(p_{2,m}) - DLC(p_{1,m}, p_{2,m}) \quad (3.14)$$
$$= M(Ent(p_{1,m}) + Ent(p_{2,m}) - Ent(H_m) - Ent(p_{1,m|} \mid H_m) - Ent(p_{2,m} \mid H_m))$$

Next, to show that Bitsave(C_M) > Bitsave(C_m), key claim is Entropy represents the regularity of the data. Since the patterns have the identical shape, its sub-patterns of smaller length also have the identical shape. The identical shape suggests the same regularity level of the data. Hence, the sum of entropy of sub-patterns is equal to the sum of entropy of the largest patterns. That is,

$$Ent(p_{1,m}) + Ent(p_{2,m}) = Ent(p_{1,M}) + Ent(p_{2,M}) \text{, and} \quad (3.15)$$
$$Ent(H_m) - Ent(p_{1,m|} \mid H_m) - Ent(p_{2,m} \mid H_m) = Ent(H_M) - Ent(p_{1,M} \mid H_M) - Ent(p_{2,M} \mid H_M)$$

Since M > m, then *bitsave*(C_M) > *bitsave*(C_m). QED.

Lemma1 suggests that even though *bitsave* function is not a monotonic to the length, it is not the case which pattern has the identical shape.

Note that the high regularity suggests essentially identical entropy for each smaller length of sub-patterns. However, when patterns have high regularity, it does not always mean that their shapes are identical. This means that Early Termination might be slightly lossy when the time series data is close to random.

3.4 Algorithm in details

Table 3.1: MDL based Motif Discovery Algorithm.

Input: Time series *T* Output: a collection of answers sorted by MDL	
1	**For** L=2 **to** T.length/2
2	{A,B}* := motifCandidateDiscovery(T,L)
3	**For each** k motif pair **in** {A,B}*
4	{group,bitsave}:= createGroup(A,B)
5	**if** bitsave < 0 **then break**
6	**while** hasNextNeighbor(T)
7	C := NextNeighbor(T,center,L)
8	{group,bs} := AddToGroup(group,C)
9	**if** bs > 0 **then** bitsave += bs
10	**else** break **end if**
11	**end while**
12	update(answer, group) // new entry
13	**end for**
14	**if** canEarlyTerminate(answer) **then break**
15	**end for**
16	**return** answer

A pseudo code is given in Table 3.1. The input is time series data. The algorithm begins by entering a **for** loop (line 1). In line 2, *MotifCandidateDiscovery* is the process of finding a motif candidate MC_L^k in a sliding window of length *L*. This function utilizes lower bounding and early abandoning techniques mentioned in previous section.

A **for** loop at line 3 represents running at different initial *hypotheses*. This loop continues until *bitsave* of creating group is negative denoted in line 5. At line 4, *A* and *B* indicate the location of subsequence in the time series. The function *createGroup* is called. Then, it returns *bitsave* and a new group. The mechanism and calculations of *bitsave* are already given in Definitions 7-14.

The *bitsave* in line 4 means number of bits saved after performing a group construction (create). If such create operation cannot produce any more *bitsave*, this pair is more likely to be meaningless. Then, the loop is broken, and it goes to *L*+1 in the next iteration.

At lines 6 – 11, the *NextNeighbor* takes a center of the group as a reference to find another neighbor *C*. The center of a group is calculated by the average of a motif

pair *A* and *B*. Then, a neighbor *C* is added to the group if its *bitsave* > 0. This loop continues until *bs* ≤ 0 or there is no next neighbor left.

At line 12, as given in Figure 3.6, after building hypothesis and joining all neighbors to a group, it is considered as a new entry. The new entry either is added in the answer set or updates an existing answer over new *bitsave* value. This is an anytime algorithm since the update function always updates motif entities towards the solutions. Hence, users need not wait until completion; rather, they can query by the status quo of the best model class.

Line 14 checks for early termination. This can be easily checked the condition whether the answer set is modified within an entire iteration or not. If not, the algorithm will halt. Finally, every class of motifs has its own *bitsave* which was updated for each iteration.

CHAPTER IV

Experiments and Results

The parameter-free myths imply that a *practical* parameter-free algorithm be correct and maintain speed. In order to prove such properties, the experiment carries out in three orthogonal aspects. The first perspective – and priority—is *correctness*. It is imperative that the parameter-free algorithm outputs valid results before exploring any other aspects. The *correctness* is determined by an objective function and comparison to related works. Next, the proposed algorithm is compared with the related work in terms of running time analysis. Finally, the parameter-freeness is demonstrated via real world datasets compared with a state-of-the-art algorithm that requires a length of motif as a parameter.

4.1 Measurements and Definitions

In order to interpret the experimental results precisely, it is more intuitive to quantify goal of achieving correctness. First of all, basic definitions are clarified. Let P be a set of planted patterns. Let D be a set of discovered patterns. Let S be a tuple of (P,D). In other words,

P = { *planted patterns* }, and D = { *discovered patterns* } (4.1)

S = (P,D) (4.2)

There are several measurements used in this section. The first measurement is **Accuracy-on-Recall (AoR)**[15]. AoR is percentage of discovered patterns corresponding to planted patterns to total planted patterns, i.e.,

$$AoR(S) = \frac{|P \cap D|}{|P|} \tag{4.3}$$

The second measurement is **Accuracy-on-Detection (AoD)**[15]. The AoD is simply an overlapping percentage between discovered motifs and the corresponding planted patterns. Let d be a discovered pattern and p be a corresponding planted

pattern, where *l*, *w* are location and length of subsequence, respectively. The unit-AoD is an overlapping percentage measurement and defined as follows,

$$UnitAoD(p,d) = \frac{OverlapingPart(l_d, l_p, w_d, w_p)}{UnionPart(l_d, l_p, w_d, w_p)} \quad (4.4)$$

where,

$$OverlapingPart(l_1, l_2, w_1, w_2) = \min(l_1 + w_1, l_2 + w_2) - \max(l_1, l_2) + 1 \quad (4.5)$$

$$UnionPart(l_1, l_2, w_1, w_2) = \max(l_1 + w_1, l_2 + w_2) - \min(l_1, l_2) + 1 \quad (4.6)$$

Hence, AoD is an average of each *UnitAOD* in a motif class, and is defined as,

$$AoD(S) = \frac{\sum_{P \cap D} UnitAoD(p,d)}{|P \cap D|} \quad (4.7)$$

where $P \cap D = \{(p,d) \mid (d \in D, p \in P) \wedge d \quad and \quad p \quad are \quad overlapped\}$

Finally, correctness is determined as the average AoD of pattern classes weighted by AoR value, i.e., given a $\Psi = \{S\}$.

$$correctness = \frac{1}{|\Psi|} \sum_{S \in \Psi} AoR(S) * AoD(S) \quad (4.8)$$

Recalled that *P* = { *planted patterns* }, and *D* = { *discovered patterns* } and S = (*P*,*D*).

There are two reasons not to include precision into correctness measurement. First, the accuracy is already measured via *accuracy-on-detection (*AoD) for each class. Second, sometimes an extraneous discovered pattern is a subsequence of random walk whose shape is essentially appealing to be another element of a motif class (cf. experiment on single dataset section). Hence, it is not intuitive to discredit the correctness when the unexpected patterns are also able to be discovered.

4.2 Correctness Experiments

As for correctness measurement, the datasets per se should be measureable, i.e., the location of motifs in each dataset should be known in advance. In this regard, finding *planted* motifs is preferable since it is easy to objectively calculate accuracy of discovered motifs with respect to planted motif occurrences. The dataset creation is by

implanting motifs into a non-repeated random walk time series. The patterns to be planted are derived from the UCR classification/clustering archive [3].

The datasets are general, and not limited to a particular case. Roughly, there are two cases to do the experiments: single-pattern datasets, multi-pattern datasets.

4.2.1 Single-pattern datasets

There are 8 datasets in this section. Each dataset consists of a "single" class of motifs, i.e., a class of motifs has a fixed length. Datasets from spd1 to spd4 contain frequent occurrences (more than two). Others have only two occurrences. Table 4.1 shows the detail of all single-pattern datasets. All datasets are portrayed as in Figure 4.1 - 4.2.

Table 4.1: Single-pattern datasets.

dataset	length	pattern	w	#patterns	dataset	length	pattern	w	#patterns
spd1	6790	50words	270	7	spd2	5760	Adiac	176	10
spd3	6350	Beef	470	7	spd4	8220	Oliveoil	570	6
spd5	1900	Gun	150	2	spd6	2150	Trace	275	2
spd7	2454	OSUleaf	427	2	spd8	2740	Oliveoil	570	2

Results

The experimental results are shown in Table 4.2. The bottom line is the proposed algorithm manages to discover *proper* length motifs planted in the single-pattern datasets. The discovered motifs are largely accurate measured by AoD (for all datasets here, AoD > 95%). In addition, the number of planted pattern and number of discovered pattern are identical (for all datasets here, AoR = 100%).

The ranking method is based on definition 7 (K-compression motif). The 1^{st} rank means the hypothesis or a subsequence as a model can best compress the times series. As for multi-occurrence pattern (spd1 to spd4), the proposed algorithm ranks correctly the planted patterns as the 1^{st} rank with respect to planted patterns.

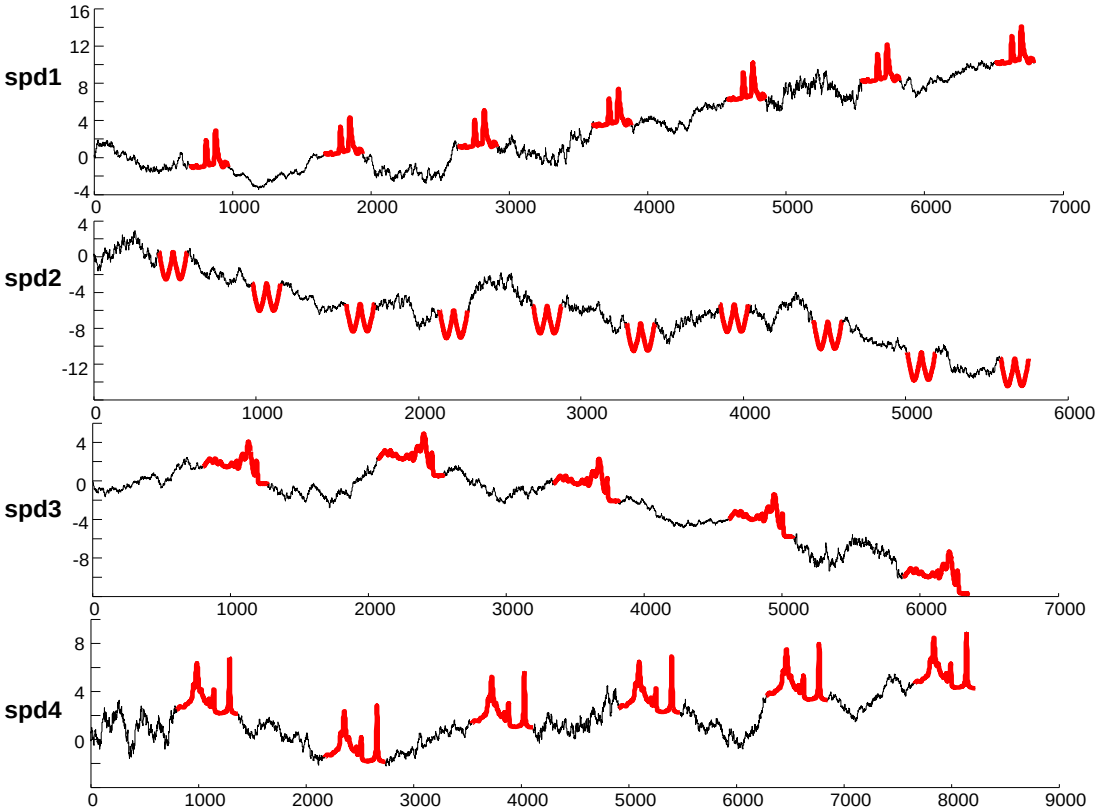

Figure 4.1: Datasets from spd1 to spd4. These contain multi-occurrences of a pattern.

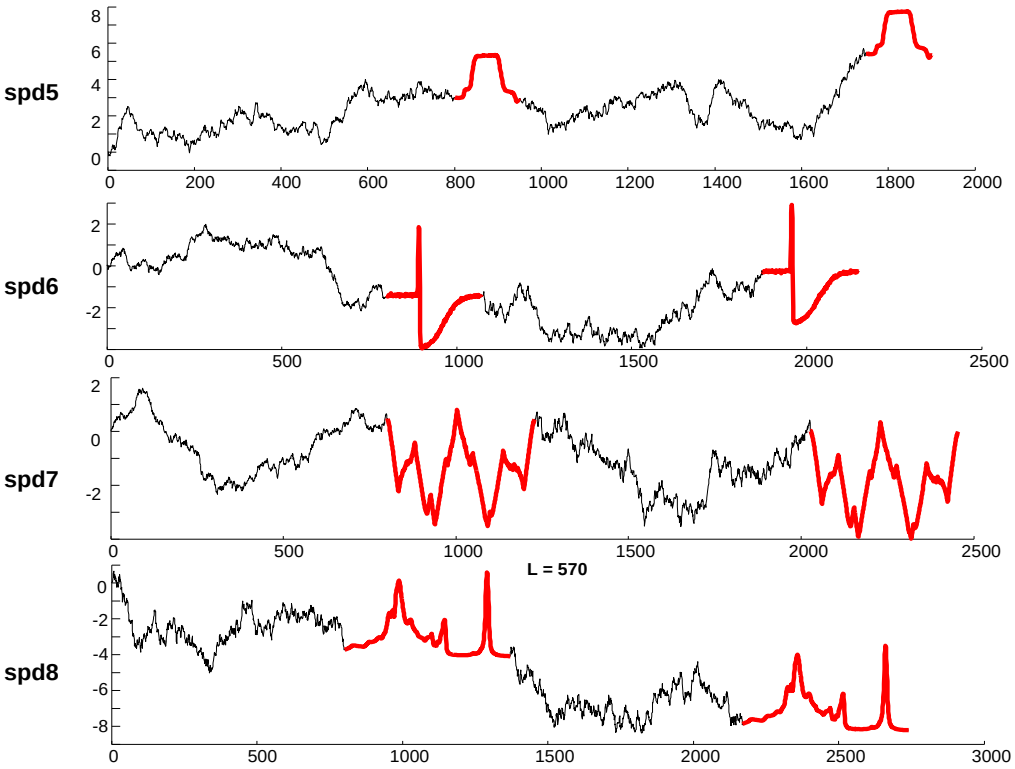

Figure 4.2: Datasets from spd5 to spd8. These data contain two occurrences of a pattern.

The interesting point is that the discovered pattern is not always the 1st rank by definition 7. There is a case that a subset of pattern can be combined with other outer subsequences, meaning that this sub-pattern is more frequent as in spd5 and spd6 (Figure 4.2). The first rank may lost some AoD since it is the frequent occurrence under sub-motifs (c.f. Figure 4.4 and Figure 4.5). Note that the *proper* length motifs of spd5 and spd6 are captured under rank 2nd and 4th respectively with AoD > 95%.

Table 4.2: Experimental Result for spd datasets.

Dataset	Pattern: #	Rank	AoR	AoD	AoR of the 1st rank	AoD of the 1st rank
spd1	50words: 7	1st	100%	98.53%	100%	98.53%
spd2	Adiac: 10	1st	100%	98.32%	100%	98.32%
spd3	Beef: 5	1st	100%	98.94%	100%	98.94%
spd4	Oliveoil: 6	1st	100%	98.95%	100%	98.95%
spd5	Gun: 2	2nd	100%	98.68%	100%	86.76%
spd6	Trace: 2	4th	100%	96.48%	100%	61.96%
spd7	OSULeaf: 2	1st	100%	99.53%	100%	99.53%
spd8*	Oliveoil: 2	1st	100%	99.65%	100%	99.65%
* Dataset spd8 has an interesting 2nd motif discovery.						

Selected proper length motif discovery results to discuss

The organization of this section is as follows. As the dataset from spd1 to spd4 yields the same results, only spd1 is selected to discuss in depth. Next, the spd5 and spd6 dataset will be discussed as its motif is not the 1st rank. Finally, the spd8 dataset will be discussed since 2nd rank motif is an unplanted motif.

The result of spd1 dataset is shown as Figure 4.3. The first motif is corresponding to the planted motif of length 270 with AoD = 98.53%. The motif discovered for each rank is highlighted as bold-face and red color. Note that 2nd motif contains two juxtaposed occurrences; they will be highlighted separately in order to be distinguishable between them. The second and third motifs are classes which occurred in random walk. As MDL principle concerns only compression ratio, it sometimes

ignores noise, whereas Euclidean distance counts exact noises. Therefore, the 2nd and 3rd motifs are share similarity in “shape” regardless of noises.

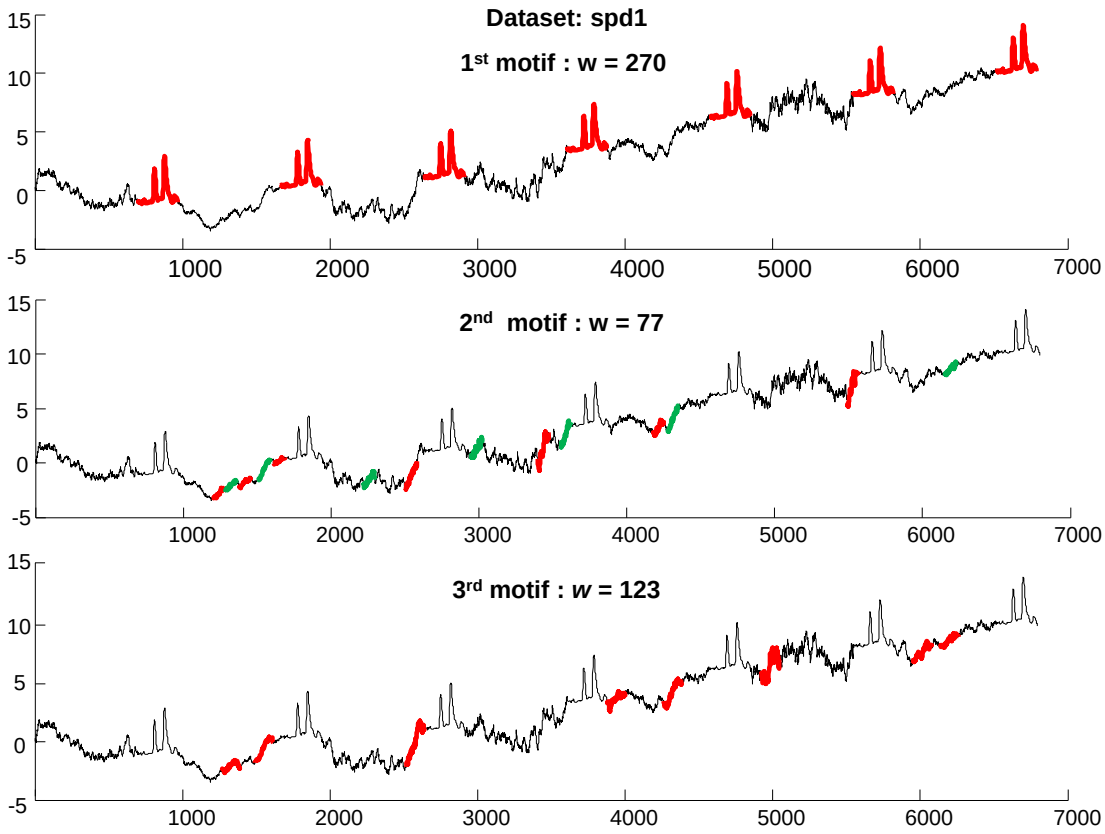

Figure 4.3: The ranked motifs from spd1 dataset. The 1st, 2nd and 3rd motif have length of 270, 77, and 123 respectively. All discovered motifs are highlighted in boldface.

The result of spd5 is shown in Figure 4.4. Notice that the planted patterns are only two occurrences; the proposed algorithm manages to discover them as 2nd rank motifs with AoD = 98.68%. The more interesting point is that the 1st motif includes a subsequence of random walk as it can compress as a neighbor of length 130. Even if user may prefer 2nd motif to 1st motif, the desirable result is still on top rank.

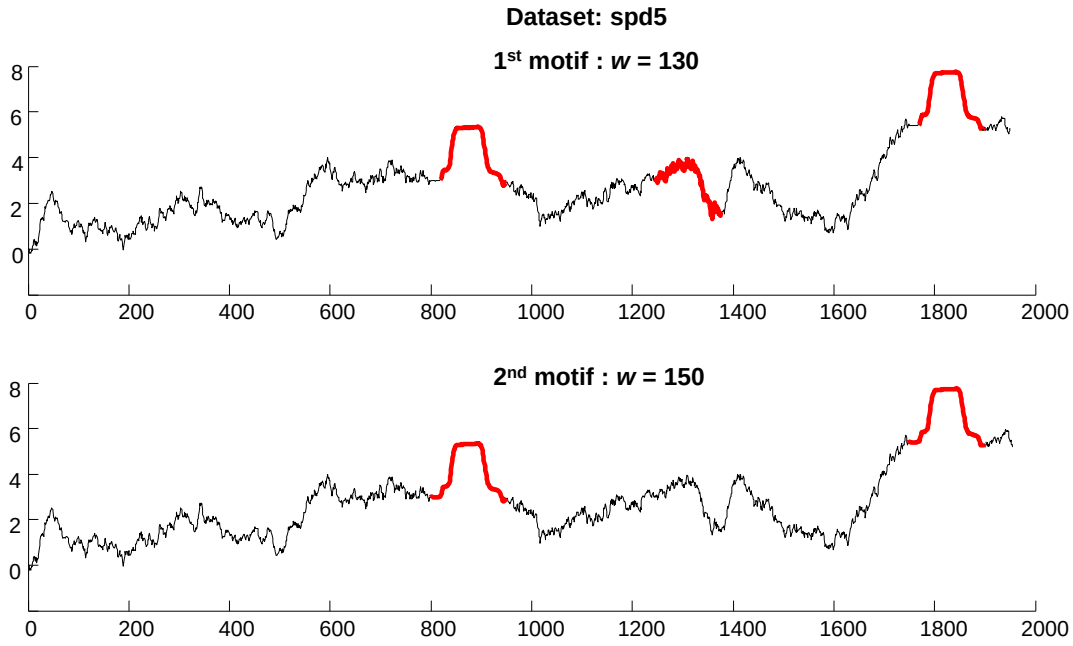

Figure 4.4: The ranked motifs from spd5 dataset. The 1st and 2nd motif have length of 130 and 150, respectively. All discovered motifs are highlighted in boldface.

Next, the result of spd6 is shown in Figure 4.5, there are four interesting ranks. The planted patterns are discovered as 4th rank in this dataset with AoD = 96.48%. The first, rank is discovered at w = 170. There are four occurrences as denoted in bold-faced red line. There are two noisy subsequences included in the 1st motif. Theses are included because their “shape” are similar. Without compression criterion, it is hard to notice these occurences. The 2nd motif is highlighted differently for contiguous subsequences for differentiability. Still, the motifs discovered are on the top rank.

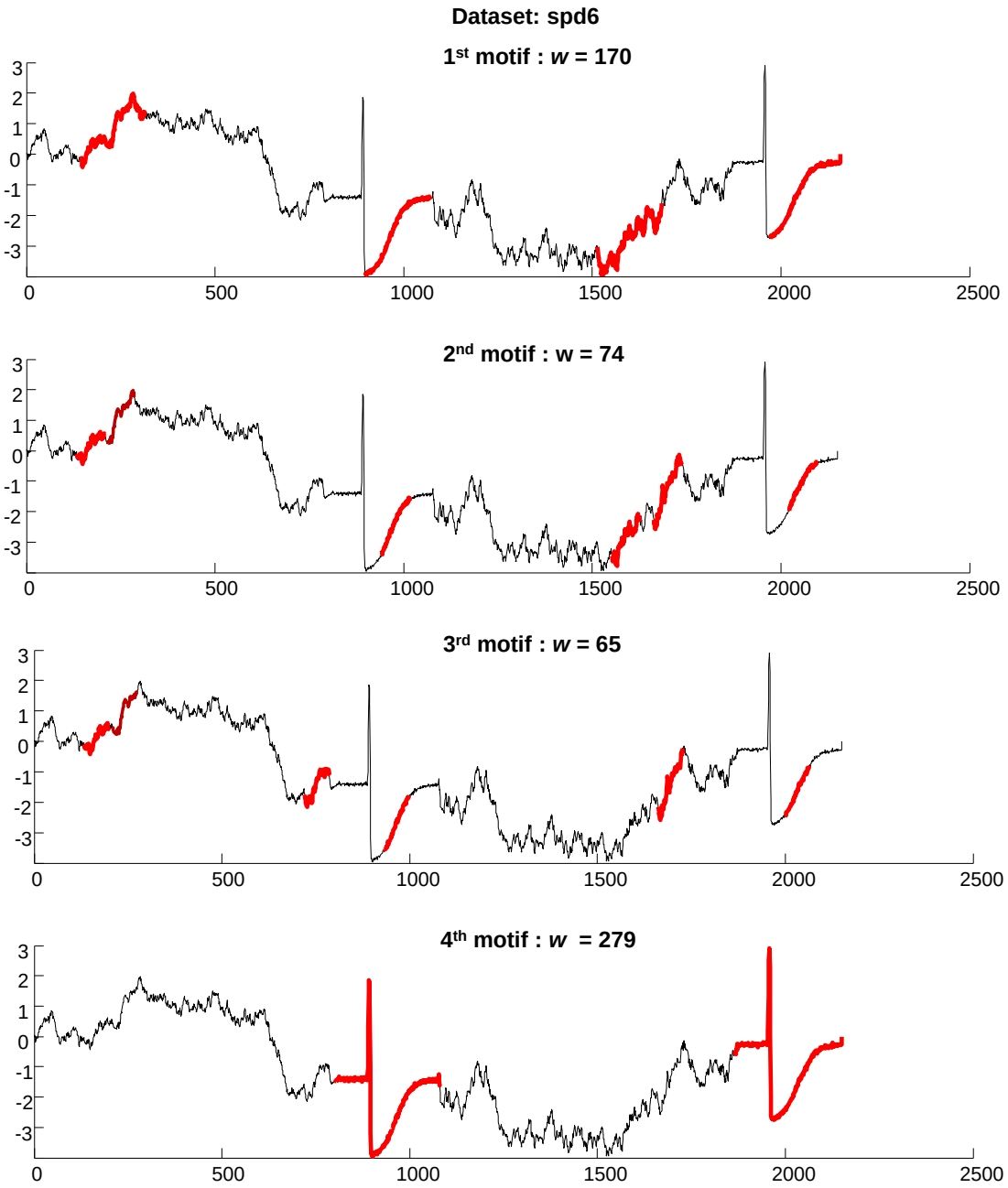

Figure 4.5 : The ranked motifs from spd6 dataset. The planted motif is discovered as 4th rank at length of 279.

Finally, the proposed algorithm manages to discover motifs in spd8 dataset correctly as the first rank with AoD = 99.65%. In addition, the second rank motif is discovered at length of 347. The closest pair of subsequences is denoted as red. The additional subsequence is a noisy one. Notice that this motif occurred unintentionally; in "shapewise," it is visually similar to others in the same class, see Figure 4.6.

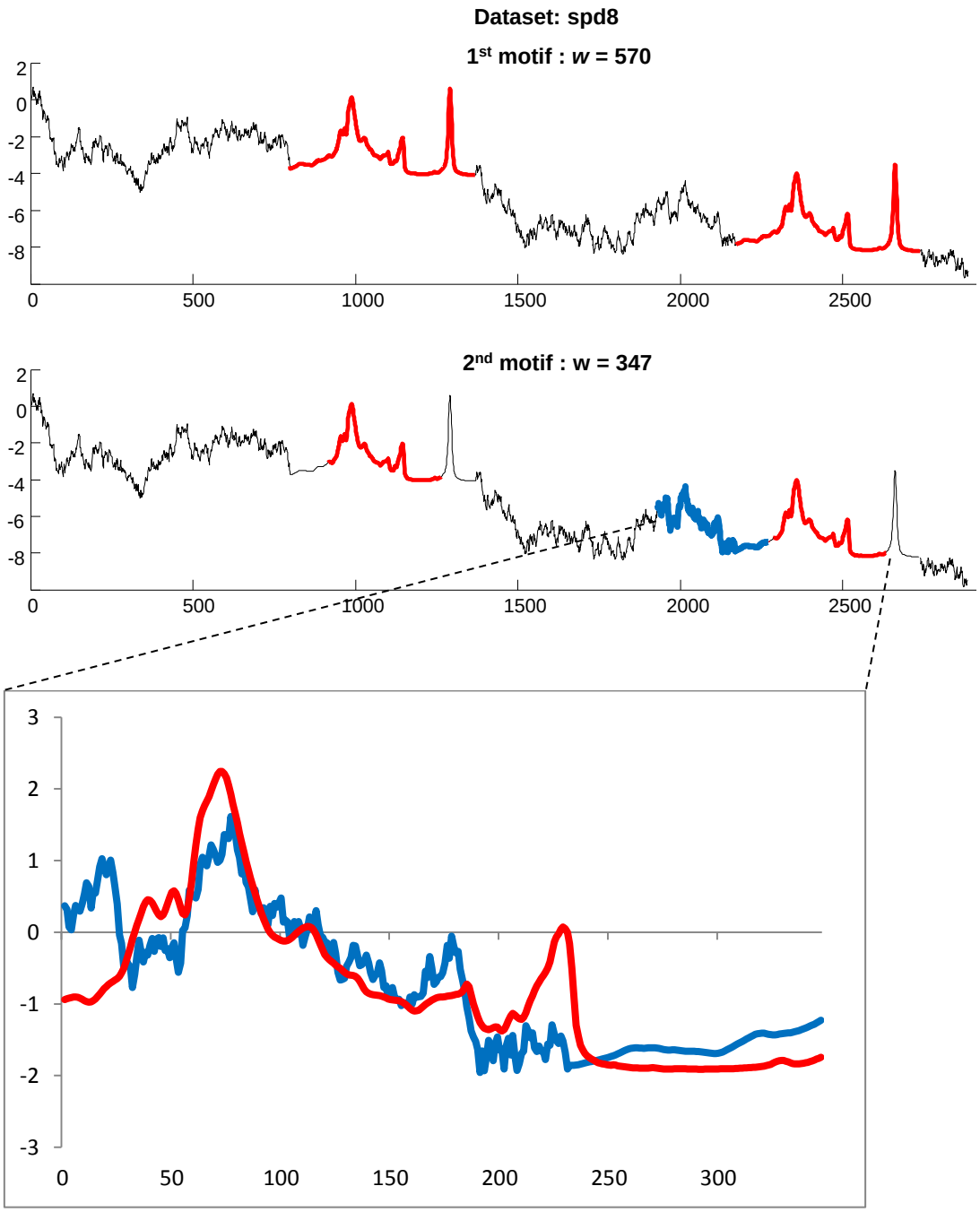

Figure 4.6: The ranked motifs from spd8 dataset. The planted motif is discovered as 1th rank at length of 570. The more interesting motif is the 2nd motif. (Bottom) the superposition of two similar motifs one of which is a subsequence of random walk.

4.2.2 Multi-pattern datasets

Having comprehended the proposed algorithm's behavior from simple datasets in previous section, this section will experiment on more complicated datasets; the datasets contain patterns in more than one class, as shown in Table 4.3. In addition, this work will be further evaluated by comparing with Parameter-free Motif Discovery for time series data [7] (we thereafter call kBM – k-Best Motif) via AoR and AoD.

There is a disparity between the proposed method and kBM. While proposed method is able to find a set of motifs—which is usually more than two subsequences, kBM can discover only a pair of motif –which in fact, some true motifs may be disregarded since they consider only a pair (not a set).

Table 4.3 : Multi-pattern datasets

Datasets	Length	Pattern	w	# pattern
mpd1	7486	Gun	150	5
		50Words	270	3
		Fish	463	2
mpd2	8842	CBF	128	8
		Yoga	426	2
		Fish	463	2
		OliveOil	570	2
mpd3	7609	FaceAll	131	9
		Adiac	176	5
		50Words	270	3
		Beef	470	2
mpd4	10979	FaceAll	131	7
		Gun	150	6
		Adiac	176	8
		OSULeaf	427	2

Result

The comparison of output is carried out in terms of AoR, AoD, and Correctness (cf. Section 4.1). Note that kBM algorithm discovers only a set of motif pair, kBM's AoR measurement will use a union relevant set of same classes to give benefit to their work. In addition, to be fair, kBM's AoD will use the maximum motif pair's AoD of the motif set.

The results are portrayed in Table 4.4. Proposed method consistently dominates kBM by higher AoD (denoted in boldface). Proposed algorithm's AoD is also relatively invariant to the size of pattern while kBM slightly deteriorates by an amount of AoD due to larger size of the pattern. This can be explained. The ranking method requires pruning off the larger size of pattern before performing ranking function. They also choose median of whole population as threshold to cut off "too large" motif. Sometimes potential motif is cut off at the first place.

Table 4.4 : Comparison of four datasets in terms of AoR, AoD, and Correctness.

Input					Output					
Datasets	Length	Pattern	Pattern Length	Number of pattern	AoR		AoD		Correctness	
					Proposed %	kBM %	Proposed %	kBM %	Proposed %	kBM %
mpd1	7486	Gun	150	5	**100**	80	98.69	99.34	**99.03**	92.22
		50Words	270	3	100	100	99.27	98.89		
		Fish	463	2	100	100	99.14	98.29		
mpd2	8842	CBF	128	8	**100**	75	98.1	98.47	**99.16**	88.73
		Yoga	426	2	100	100	**99.6**	96.39		
		Fish	463	2	100	100	**99.7**	94.31		
		OliveOil	570	2	100	100	**99**	90.35		
mpd3	7609	FaceAll	131	9	**88.89**	66.67	98.51	100	**96.19**	70.25
		Adiac	176	5	**100**	60	98.33	97.74		
		50Words	270	3	**100**	66.67	**99.27**	85.09		
		Beef	470	2	100	100	99.58	98.95		
mpd4	10979	FaceAll	131	7	**100**	57.14	97.78	98.51	**98.74**	68.31
		Gun	150	6	**100**	50	99.34	98.69		
		Adiac	176	8	**100**	75	98.31	99.44		
		OSULeaf	427	2	100	100	**99.53**	93.45		

As for AoR, the proposed method discovers virtually all of planted patterns for each dataset. However, kBM manages to discover partial set of them. kBM's AoR is uncertain when the frequency of the patterns' occurrences is more than two. When number of patterns is more than two, kBM manages to discover all of them.

Here are details of each dataset. In mpd1, the result of proposed method and kBM is comparable both in terms of AoR and AoD. In mpd2, the kBM's correctness slightly decreases since the overall AoDs deteriorate, especially in Oliveoil's pattern when ten percent of portion is missing. In mpd3, kBM's AoR is evidently lower than previous dataset. This is the case when the number of patterns is higher than two since kBM focus on finding only a pair of motif. This can be redundant since there are $m(m-1)/2$ pairs possible when m represents number of patterns. This also happens in mpd4.

4.2.3 Does parameter-freeness reduce correctness?

The datasets are from planted patterns with random-walk data. Also, there are various patterns within the datasets. Note that the experimental results suggest that the proposed parameter-free algorithm manages to discover variable-length motifs with high recall rate ($AoR > 88\%$) and accuracy ($AoD > 95\%$). In other words, the proposed method manages to discover all of motifs with less than 5% missing *correctness*. In addition, the discovered motifs are on the top rank overall possible candidate motifs, which has at most $O(n^3)$ subsequences. $O(n^3)$ is calculated as follows:

Given a time series of length n, the space of all possible pairs of subsequences can be calculated via simple counting. For fixed length w, there are $\binom{n-w}{2}$ pairs. Counting for $w = 2$ to $n/2$, the total number of possible pairs are,

$$\sum_{w=2}^{n/2}\binom{n-w}{2} \le \sum_{w=2}^{n/2}\binom{n}{2} \le n\binom{n}{2} = O(n^3) \tag{4.9}$$

These results strongly suggest that the statement "parameter-free algorithm's correctness is sacrificed in exchange for parameter-freeness" is simply a myth.

4.3 Speed Experiments: Scalability

4.3.1 Running time analysis

Given sliding windows of length *m*, and time series of length *n* finding motifs of length *m* takes O(mn^2). In addition, running from all lengths *m* = 2 to *n*/2, the complexity becomes,

$$T_1(n) \le \sum_{m=2}^{n/2} mn^2 = n^2 \sum_{m=2}^{n/2} m \le n^2 \frac{(n(n-2))}{8} \tag{4.10}$$

$$\therefore T_1(n) = O(n^4) \tag{4.11}$$

Thus, it spends $O(n^4)$ to discover all possible lengths of motifs. Even if time complexity is high, the proposed algorithm applies additional techniques. Two major novel techniques are early termination and making to be an anytime algorithm. Time complexity can be asymptotically reduced by applying a technique called Early Termination. Consider the largest motifs in time series of length *M*, where $M << n$. Instead of running all possible lengths, the proposed algorithm runs only to at most length *M*. That is,

$$T_2(n) \le \sum_{m=2}^{M} mn^2 = n^2 \sum_{m=2}^{M} m \le n^2 \frac{(M(M-1))}{2} \tag{4.12}$$

$$\therefore T_2(n) = O(M^2 n^2) \tag{4.13}$$

Therefore, with Early Termination, the proposed algorithm spends $O(M^2n^2)$, where $M << n$, while original work done on single length requires $O(mn^2)$. That is, the proposed algorithm's running is bounded by length of patterns. In addition, the proposed algorithm is an anytime algorithm, it can provide best-so-far motifs while running as it is an anytime algorithm. This also gives us an advantage when users want to determine roughly the current motifs to use in another subroutine or process.

4.3.2 Empirical observations

Empirical scalability experiments are shown in Figure 4.7. The datasets are the same as those in multi-pattern dataset section. All algorithms are executed on Intel-i7 3GHz server. As in Figure 4.7, the proposed method is approximately a few magnitudes

faster than kBM. Notice that *kBM* and Proposed without Early Termination is essentially the same order of magnitude. With Early Termination, the algorithm is a few magnitudes faster than kBM. Therefore, it is not dead-end when worst-case time complexity is high as speedup over average case is possible.

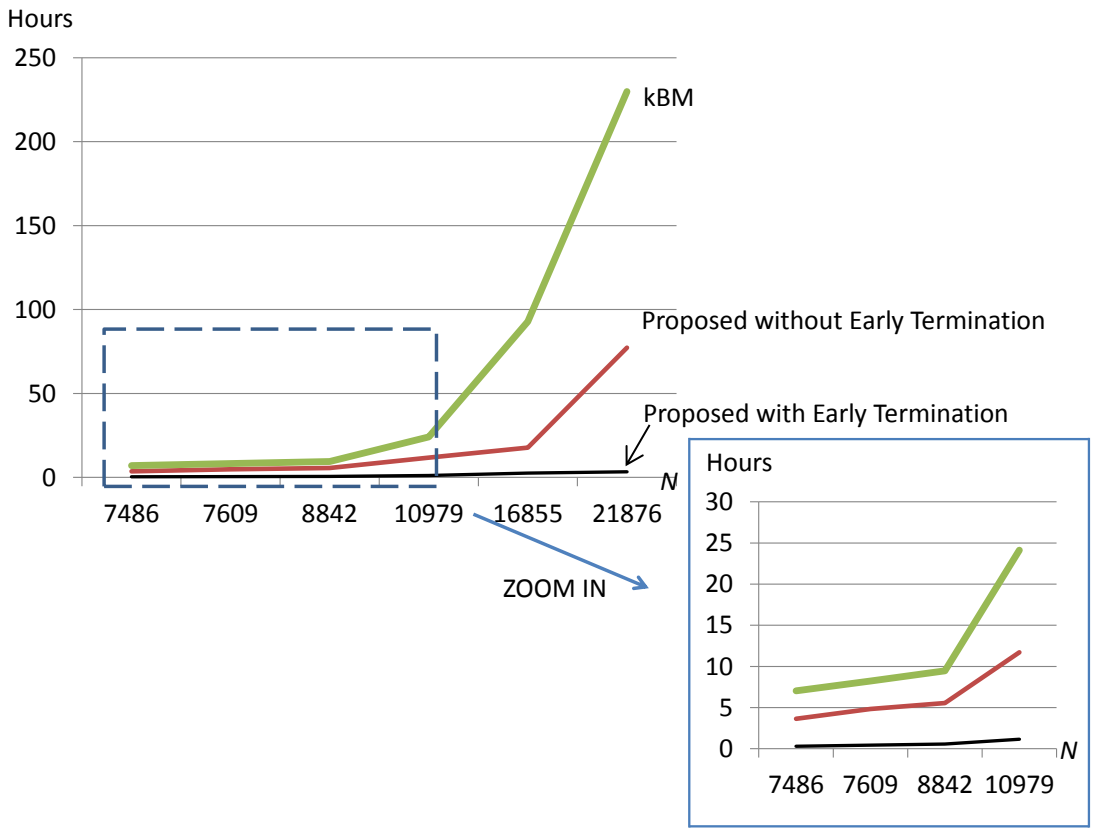

Figure 4.7 : Comparison among three methods. The y-axis represents running time in hours. The x-axis represents length of time series. The proposed method is a few magnitudes faster than kBM.

4.3.3 Anytime proper length time series motif discovery

In this section, two desirable properties of anytime algorithm are evaluated: *monotonicity* and *interruptibility* properties (cf. Section 2.1.4). In order for quality of the algorithm to be measurable, this work use *correctness* (cf. section 4.1) as a measurement. The datasets are from mpd1 to mpd4. The results are shown as Figure 4.8.

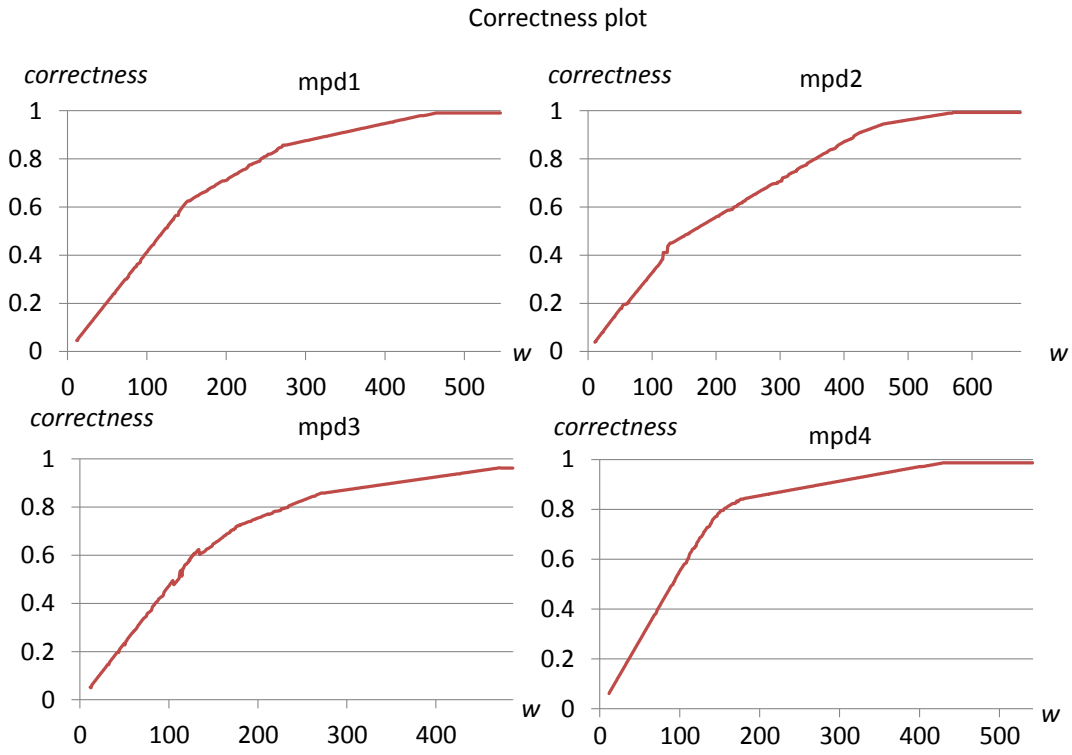

Figure 4.8 : Correctness plot for mpd1 to mpd4 datasets.

As for *monotonicity*, the correctness value for each dataset is non-decreasing. In addition, a user can interrupt the algorithm at anytime while running. For example, a user wants the partial result from the mpd4 dataset while running at w = 200. He or she can obtain the valid result at correctness more than 80 percent since most of patterns are of length less than 200. Consequently, the algorithm still has a high percentage of correctness while running at w = 200. This demonstrates the *interruptibility* of the algorithm in cases that user may request the premature answer from the running algorithm.

4.3.4 Is parameter-free algorithm always slow?

With inefficient implementation, the running time is bounded by brute force search as $O(n^4)$, where n = time series length. This is too slow and impractical. On the other hand, with additional optimization via Early Termination, it is possible to speedup since the algorithm will run on need-only basis.

According to the running time analysis, the proposed algorithm runs at most $O(M^2n^2)$, where M is the largest patterns and $M << n$, while original work done on single length requires $O(mn^2)$. That is, the amount of pruning power depends on the pattern size. In addition, Figure 4.7 does strongly suggest that the proposed algorithm with Early Termination allows a very impressive speedup over brute force search (kBM).

To summarize, parameter-free algorithm with an efficient implementation allows a speedup over brute force. It also can prune off search space at run time. Hence, both theoretical and experimental evident strongly suggest that the statement "parameter-free algorithm is always not faster non-parameter-free algorithm" is also a myth.

4.4 Parameter-freeness Test: comparison with state-of-the-art algorithm

There are many existing works that requires many predefined parameters. Exact Motif Discovery [2] (called thereafter MK – Mueen-Keogh) is selected as a baseline because it requires only length of motif to be specified. The objective in this section is to show that the proposed algorithm manages to find proper length of motif for each class without indicating any parameter while state-of-the-art method [2] requires a user to select window size. To be fair to the MK method, benefit is given to them by manually selecting the "best" length for them.

The real-world datasets are used. This dataset is ECG— abnormal ECGs with features of PVCs and ventricular couplets on bradycardia, and WPW [14]. The record is 30 minutes long. Abnormal ECG is more preferable because it exhibits anomaly in some beats.

The proposed method shows a result in Figure 4.9. The first rank motif is motif of length 194, which has myriad neighbors. In fact, this first motif is a regular heartbeat. The more interesting discovery is the second motif of length 247. Although the best motif lengths of those classes are 195 and 245, respectively, obtained by "picking" the best length for running MK, the differences are not significant. In addition, the proposed method is quite effortless to the user because it can discover the commensurate size of each class without letting the user "manually" select a length as a parameter.

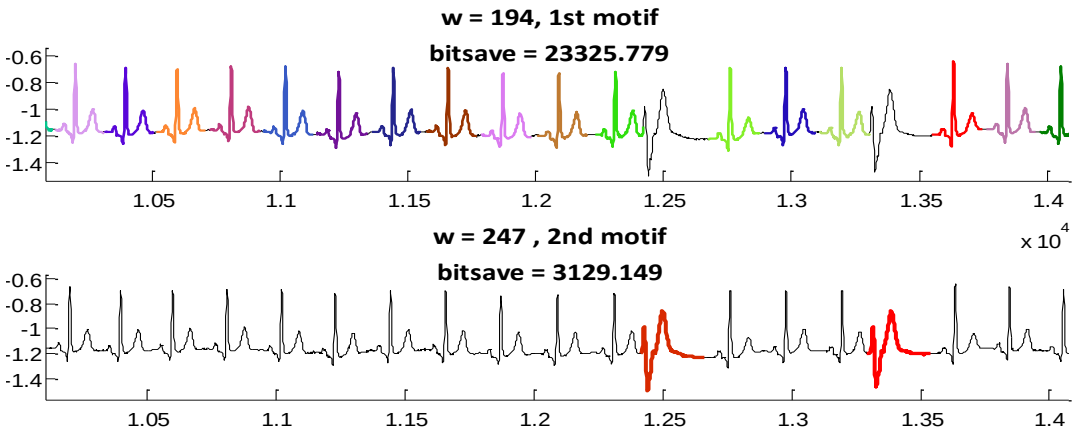

Figure 4.9 : A snippet of ECG dataset. Total length is 15420. The highlight colors are motif discovered at length w=194 and 247, respectively. The proposed method is able to find two different classes of motif. Each length is 194 and 247, respectively, while state-of-the-art (MK) method uses “best size” of 195 and 245, respectively

4.5 Experiment Summary

The experiments have been designed to correspond to the objectives: parameter-freeness, correctness, and speed. The metrics used for this work are (Accuracy-on-Recall) AoR, and (Accuracy-on-Detection) AoD. The datasets are from both planted patterns with random walk and real-world datasets.

As for correctness, the datasets are planted patterns with random walk since the exact position of patterns are known, and it is easy to evaluate objectively. There are two subsections: single-pattern datasets and multi-pattern datasets. The single-pattern is a simple time series data. The purpose of using single-pattern datasets is to evaluate the proposed algorithm’s behavior with AoR and AoD. The next datasets are multi-pattern datasets. In this datasets, the proposed algorithm is compared to kBM via results of multi-pattern datasets. The main finding is that the proposed algorithm manages to discover proper length of motifs with consistently high k-AoD and AoR while kBM’s k-AoD deteriorates in some cases, especially in larger patterns.

As for speed, the datasets are the same as above. The proposed algorithm is compared to kBM in terms of speed analysis. Theoretically, According to the running time analysis, the proposed algorithm runs at most $O(M^2n^2)$, where M is the largest patterns and $M << n$, while original work done on single length requires $O(mn^2)$. On the

other hand, kBM requires $O(n^4)$. Experimentally, the proposed algorithm is a few magnitudes faster than kBM.

Finally, as for parameter-freeness, the dataset is real-world data from ECG—abnormal ECGs with features of PVCs and ventricular couplets on bradycardia, and WPW. This work is compared to MK motif discovery. The benefit has given to MK via selecting the “best” parameter for them. The result is that the proposed algorithm effortlessly discovered the “proper” length of two classes of motifs: regular beats and anomaly beats while the lengths of these two classes are commensurate with “best” length from MK.

CHAPTER V

CONCLUSION AND RECOMMENDATIONS

4.1 General Conclusion

Time series motif discovery has been actively explored in the past decade. The problem of finding proper predefined parameters is untenable and still unsolved. Reducing parameters to be zero is difficult since the motifs are highly sensitive to choices of length as a parameter. In addition, even if the motif discovery is run at various parameters, it is still arduous to rank all of them objectively, not to mention possible redundancy of sub-motifs. Ultimately, the running all possible ranges of parameter is computationally expensive.

There are some attempts to deal with a parameter issue. However, they cannot truly eliminate parameters. The first parameter-free motif discovery(*kBM*) has addressed this problem, but it is not as efficient due to scalability problem and solution quality. Hence, these reasons inspire this work to introduce a proper length time series motif discovery with parameter-freeness. The objectives of this work are parameter-freeness, correctness, and speed.

One of most quintessential of time series motif discovery is a definition of motifs. A good motif definition reflects a good algorithm design. It is possible that motifs be defined in various ways depending on domain applications. In fact, there are two main bases of those definitions: similarity basis or frequency basis. These definition, however, assume that length of motifs to be discovered is known by users. Unfortunately, this information is barely available to users, even for domain experts. Since variable length of motif is required to be defined in order to evaluate precisely, kBM has defined motifs as K-best motifs. In short, K-best motifs mean location-similarity supported motifs. That is, a good motif should have a lot of its supported subset. In this work's perspective, the definition of K-best motifs is biased toward larger size of motifs. Therefore, this work introduces alternative definition of variable-length motifs via K-compression motifs.

There are misconceptions regarding parameter-freeness of an algorithm. This work clarifies via introducing three parameter-free myths. These myths are confirmed by experimental evidences.

The proposed algorithm is parameter-free. The principle behind the algorithm is that a proper length motif is a hypothesis that can best compress the time series. This can be realized via MDL principle. In addition, this work utilizes compression score function as heuristic information in both Update Answer Sets and Early Termination Techniques. The gist is that by using compression score function, the algorithm manages to run on need-only basis of search space.

The experiments have been designed to correspond to the objectives: parameter-freeness, correctness, and speed. The metrics used for this work are RR, k-AoD, and AoR. The datasets are from both planted patterns with random walk and real-world datasets. As for correctness, the main finding is that the proposed algorithm manages to discover proper length of motifs with consistently high k-AoD and AoR while kBM's k-AoD deteriorates in some cases especially in larger patterns. As for speed, the proposed algorithm is faster than kBM, theoretically and experimentally.

4.2 Recommendations

Even though the proposed algorithm manages to discover motifs with high accuracy, the outputs sometimes include noises as a subsequence (false positive). This error may affect the ranking of that class of motifs. In the worst case, some potential motifs are missing (this is a subtle point regarding how to select a criterion or create/add members). This happens because the choice of compression is MDL principle implemented via entropy basis. Entropy is regard as a "coarse" grain MDL. Therefore, a further refined version of MDL can enhance the accuracy. Furthermore, Euclidean distance does not allow warping distance in the time series. This can be improved by using alternative distances that allow warping such as Dynamic Time Warping (DTW).

As for speed, one weakness of this work is at the very small size of motifs there are myriads candidates. So, it is slightly slow at the beginning since most of small size motifs are likely to be similar. One way to mitigate this issue is to provide an optional parameter to begin with any length other than 2. Typically, this optional parameter is trivial since some small motifs are meaningless.

Even if the proposed algorithm has speedup over brute force search, a question still remains: can it be faster? For example, why runs from bottom to top incrementally only one length. Why not run on every other two lengths? By doing this, it is possible to incorporate Apriori-like algorithm to discover proper length at a very fast speed.

REFERENCES

[1] A. Mueen, E. Keogh, and N. B. Shamlo. Finding Time Series Motifs in Disk-Resident Data. In Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09), pp. 367-376, Miami, Florida, USA, December 6 - 9 2009.

[2] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact Discovery of Time Series Motifs. In Proceedings of the 9th SIAM International Conference on Data Mining (SDM'09), pp. 473-484, Sparks, Nevada. April 30 – May 2 2009.

[3] E. Keogh, X. Xi, L. Wei, C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage. [Online]. 2008. Available from: http://www.cs.ucr.edu/?eamonn/time_series_data/, 2012.

[4] H. Li and N. Abe. Clustering Words with the MDL Principle. In Proceedings of the 16th Conference on Computational Linguistics (COLING'96), pp. 5-9, Copenhagen, Denmark, August 5-9, 1996.

[5] H. Tang and S. S. Liao. Discovering Original Motifs with Different Lengths from Time Series. Knowledge-Based Systems 21(2008): 666-671.

[6] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding Motifs in Time Series. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02), pp. 53-68, Edmonton Alberta, Canada. July 23-26 2002.

[7] P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana. Parameter-free Motif Discovery for Time Series Data. In Proceedings of the 9th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI-CON'12), pp. 1 – 4, Hua Hin, Prachuap Khiri Khan, Thailand, May 17-19 2012.

[8] T. Rakthanmanon, E. Keogh, S. Lonardi, S. Evans. Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data. In Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11), pp. 547-556, Vancouver, Canada, December 11 – 14 2011.

[9] S. Rodpongpun, V. Niennattrakul, and C.A. Ratanamahatana. Selective Subsequence Time Series Clustering. Knowledge-Based Systems 35(2012): 361 - 368.

[10] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. Machine Learning 58 (2005): 269-300.

[11] Y. Li, J. Lin. Approximate variable-length time series motif discovery using grammar inference. In Proceedings of the 10th International Workshop on Multimedia Data Mining (MDMKDD'10), pp. 1–9, Washington DC, USA, July 25 – 28 2010.

[12] B. Hu, T. Rakthanmanon,Y. Hao et al. Discovering the Intrinsic Cardinality and Dimensionality of Time Series Using MDL. In Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11), pp.1086-1091, Vancouver, Canada, December 11 – 14 2011.

[13] H. T. Lam, T. Calders, and N. Pham. Online Discovery of Top-k Similar Motifs in Time Series Data. In Proceedings of the 11th SIAM International Conference on Data Mining (SDM'11), pp. 1004-1015, Mesa, Arizona, USA, April 28 – 30 2011.

[14] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23)(2000), pp. e215-e220.

[15] V. Niennattrakul, D. Wanichsan, and C. A. Ratanamahatana. Accurate Subsequence Matching on Data Stream under Time Warping Distance. In

Proceedings of Workshops of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09), pp. 752-755, Pattaya, Thailand.

[16] P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana. Discovery of Variable-Length Time Series Motif. In Proceedings of the 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON'11), pp. 472 – 475, Khon Kean, Thailand, May 17 – 20 2011.

[17] Q. Zhu, G. Batista, T. Rakthanmanon, and E. Keogh. A Novel Approximation to Dynamic Time Warping allows Anytime Clustering of Massive Time Series Datasets. In Proceedings of the 12th SIAM International Conference on Data Mining (SDM'12), pp. 999 – 1010, Anaheim, California, USA, April 26-28 2012.

Biography

Sorrachai Yingchareonthawornchai was born in Bangkok, Thailand, on October 16, 1989. In 2008, he graduated from Watsuthivararam School. His Bachelor's degree had been under supervision of Associate Professor Dr. Somchai Prasitjutrakul. During senior, his senior project regarding quantum computing had been under supervision of Professor Dr. Prabhas Chongstitvatana. He received his B.Eng. in Computer Engineering from Chulalongkorn University in 2012. His Master's degree at Chulalongkorn University (2012 – 2013) has been under supervision of Assistant Professor Dr. Chotirat Ann Ratanamahatana. During his M.Eng study, he was granted Returning Student Scholarship of academic year 2012 from the Department of Computer Engineering, Chulalongkorn University. He was also an external reviewer of Pacific Asia Knowledge Discovery and Data Mining (PAKDD). His research interests include but not limited to time series data mining, machine learning, and quantum computing.