**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH**

****

**ĐỀ CƯƠNG LUẬN VĂN**

**TÌM CHUỖI CON BẤT THƯỜNG TRONG DỮ LIỆU CHUỖI THỜI GIAN BẰNG PHƯƠNG PHÁP ĐÁNH GIÁ HỆ SỐ BẤT THƯỜNG**

**GVHD: PGS.TS Dương Tuấn Anh**

**---o0o---**

***HVTH:* Ngô Duy Khánh Vy 13073042**

TP. HỒ CHÍ MINH, 05/2015

# MỤC LỤC

[MỤC LỤC ii](#_Toc419526331)

[DANH MỤC HÌNH iv](#_Toc419526332)

[DANH MỤC BẢNG vi](#_Toc419526333)

[Chương 1 GIỚI THIỆU 1](#_Toc419526334)

[1.1. Đặt vấn đề 1](#_Toc419526335)

[1.2. Mục tiêu của đề tài 2](#_Toc419526336)

[1.3. Cấu trúc đề cương 3](#_Toc419526337)

[Chương 2 CƠ SỞ LÝ THUYẾT 4](#_Toc419526338)

[2.1. Các định nghĩa 4](#_Toc419526339)

[2.2. Các giải thuật phân đoạn 5](#_Toc419526340)

[2.2.1. Giải thuật cửa sổ trượt 5](#_Toc419526341)

[2.2.2. Giải thuật từ trên xuống 7](#_Toc419526342)

[2.2.3. Giải thuật từ dưới lên 7](#_Toc419526343)

[2.3. Các phương pháp tính khoảng cách 9](#_Toc419526344)

[2.3.1. Công thức tính khoảng cách Euclid 9](#_Toc419526345)

[2.3.2. Phương pháp xoắn thời gian động 10](#_Toc419526346)

[2.4. Các phương pháp thu giảm số chiều và rời rạc hóa dữ liệu 13](#_Toc419526347)

[2.4.1. Phương pháp xấp xỉ PAA. 13](#_Toc419526348)

[2.4.2. Phương pháp biến đổi dạng sóng Haar 15](#_Toc419526349)

[2.4.3. Phương pháp biểu diễn SAX 17](#_Toc419526350)

[Chương 3 GiỚI THIỆU CÁC CÔNG TRÌNH LIÊN QUAN 20](#_Toc419526351)

[3.1. Các công trình liên quan đến phân đoạn 20](#_Toc419526352)

[3.1.1. Giải thuật phân đoạn từ trên xuống cải tiến của D. Lemire 20](#_Toc419526353)

[3.1.2. Giải thuật phân đoạn SWAB 22](#_Toc419526354)

[3.2. Các công trình về tìm kiếm chuỗi con bất thường 23](#_Toc419526355)

[3.2.1. Giải thuật HOT SAX 23](#_Toc419526356)

[3.2.2. Giải thuật WAT 27](#_Toc419526357)

[3.2.3. Giải thuật tìm các chuỗi con bất thường có độ dài khác nhau của Leng và các cộng sự 30](#_Toc419526358)

[Chương 4 NỘI DUNG NGHIÊN CỨU VÀ KẾ HOẠCH LÀM VIỆC 32](#_Toc419526359)

[4.1. Nội dung nghiên cứu 32](#_Toc419526360)

[4.2. Kế hoạch làm việc 34](#_Toc419526361)

[TÀI LIỆU THAM KHẢO 35](#_Toc419526362)

# DANH MỤC HÌNH

[Hình 1.1: Chuỗi thời gian biểu diễn trên mặt phẳng 1](#_Toc419526363)

[Hình 2.1. Giải thuật cửa sổ trượt. 6](#_Toc419526364)

[Hình 2.2. Giải thuật từ trên xuống. 7](#_Toc419526365)

[Hình 2.3. Giải thuật từ dưới lên. 8](#_Toc419526366)

[Hình 2.4. Hai chuỗi thời gian hình dạng giống nhau nhưng bị lệch theo trục tung 10](#_Toc419526367)

[Hình 2.5. (a) Đo khoảng cách bằng công thức Euclid. (b) Đo khoảng cách bằng phương pháp xoắn thời gian động. 12](#_Toc419526368)

[Hình 2.6. Ma trận xoắn thời gian và đường xoắn thời gian. 13](#_Toc419526369)

[Hình 2.7. Phương pháp xấp xỉ PAA thu giảm số chiều của một chuỗi thời gian. 14](#_Toc419526370)

[Hình 2.8 Biến đổi dạng sóng Haar cho hàm *f(x) =* (9 7 3 5). 15](#_Toc419526371)

[Hình 2.9. Hiện thực phương pháp biến đổi dạng sóng Haar bằng phép nhân ma trận. 16](#_Toc419526372)

[Hình 2.10. Giải thuật biến đổi dạng sóng Haar của Fu và các cộng sự 17](#_Toc419526373)

[Hình 2.11. Bảng các điểm chia với *a* từ 3 đến 10 18](#_Toc419526374)

[Hình 2.12. Chuỗi thời gian được biểu diễn thành chuỗi *cbccbaab* 18](#_Toc419526375)

[Hình 3.1. Mã giả cho giải thuật của D. Lemire. 21](#_Toc419526376)

[Hình 3.2. Giải thuật SWAB 23](#_Toc419526377)

[Hình 3.3. Giải thuật vét cạn tìm chuỗi con bất thường 25](#_Toc419526378)

[Hình 3.4. Giải thuật cải tiến từ giải thuật vét cạn. 25](#_Toc419526379)

[Hình 3.5 . Hai cấu trúc dữ liệu hỗ trợ cho việc sắp xếp thứ tự các chuỗi con trong hai vòng lặp. 26](#_Toc419526380)

[Hình 3.6. Chữ cái đầu tiên của từ được xem xét khi phân tách nút gốc của cây 28](#_Toc419526381)

[Hình 3.7. Chữ cái thứ hai được xem xét khi tiến hành phân tách các nút *a, b, c*. 29](#_Toc419526382)

# DANH MỤC BẢNG

[Bảng 2.1. Các ký hiệu sử dụng trong mục 2.2. 6](#_Toc419526383)

[Bảng 4.1. Bảng kế hoạch làm việc. 34](#_Toc419526384)

# GIỚI THIỆU

## Đặt vấn đề

Ngày nay dữ liệu chuỗi thời gian xuất hiện ngày càng nhiều trong nhiều lĩnh vực khác nhau trong cuộc sống như kinh tế, y khoa, thiên văn…Một chuỗi thời gian là một dãy các số thực, mỗi số biểu diễn giá trị của một đại lượng được xác định tại các điểm thời gian cách đều nhau. Một chuỗi thời gian thường được biểu diễn thành các điểm trên một mặt phẳng hai chiều với hoành độ là thời gian và tung độ là giá trị của đại lượng quan tâm tại thời điểm đó. Hình 1.1 bên dưới là biểu diễn của một chuỗi thời gian trong mặt phẳng. Thông thường khi nghiên cứu dữ liệu chuỗi thời gian người ta không quan tâm đến giá trị tại từng thời điểm mà quan tâm đến một đoạn gồm nhiều giá trị liên tục, vì vậy ta có thể xem một đoạn của một chuỗi thời gian là một đối tượng dữ liệu đa chiều. Số chiều của đối tượng dữ liệu này có thể có độ dài từ vài chục như doanh số bán hàng theo ngày của một cửa hàng trong một quí hay có độ dài đến vài trăm triệu như giá trị điện tim của một bệnh nhân. Hiện nay một máy cảm ứng có thể thu thập được hơn một triệu điểm dữ liệu chỉ trong vòng 3 phút [4].



Hình 1.1: Chuỗi thời gian biểu diễn trên mặt phẳng

Trong những năm gần đây, có rất nhiều công trình nghiên cứu về việc phát hiện ra các chuỗi con bất thường, tức là một đoạn trong một chuỗi thời gian khác biệt với phần còn lại của chuỗi. Việc phát hiện ra các chuỗi con bất thường như vậy có rất nhiều ứng dụng trong thực tiễn. Chẳng hạn các thiết bị theo dõi sức khỏe tự động có thể phát hiện ra các đoạn bất thường trong dữ liệu điện tim của người dùng và gởi đi các cảnh báo. Trong bài toán gom cụm trong dữ liệu chuỗi thời gian, giải thuật phát hiện các đoạn bất thường có thể dùng để loại bỏ các đoạn quá khác biệt mà ta có thể xem là các phần tử nhiễu, hay phần tử ngoại biên. Tuy nhiên việc phát hiện các chuỗi con bất thường trong dữ liệu chuỗi thời gian là không đơn giản. Khó khăn thứ nhất là ta không biết trước được chiều dài của các chuỗi con này, thứ hai làm sao ta có thể xác định một chuỗi con là khác biệt so với các chuỗi con khác trong chuỗi thời gian, thứ ba không thể duyệt để so sánh từng đoạn một các đoạn trong chuỗi dữ liệu thời gian vì chiều dài của chuỗi thường rất lớn.

Nhiều nhà nghiên cứu đã quan tâm đến bài toán này và đưa ra các giải thuật tìm kiếm hiệu quả như Eamonn Keogh và các cộng sự với giải thuật HOT SAX [5] hay Mingwei Leng và các cộng sự với phương pháp sử dụng *hệ số bất thường* (anomaly factor) [10].

## Mục tiêu của đề tài

Phương pháp sử dụng hệ số bất thường của Leng và các cộng sự là một phương pháp hiệu quả. Phương pháp này có khả năng phát hiện được các chuỗi con bất thường có độ dài khác nhau mà không cần biết trước chiều dài của các đoạn và có thể áp dụng được cho các chuỗi thời gian dạng luồng nên có khả năng áp dụng cao trong thực tế. Tuy nhiên giải thuật phải sử dụng độ đo xoắn thời gian động để đánh giá khoảng cách của các đoạn dữ liệu có độ dài khác nhau. Điều này làm cho giải thuật phải tốn nhiều thời gian thực thi và không hiệu quả đối với các chuỗi dữ liệu lớn.

Mục tiêu của đề tài này là cải tiến giải thuật của Leng để có thể giảm được thời gian tính toán mà vẫn giữ được các ưu điểm của giải thuật. Giải thuật mới sẽ được so sánh bằng thực nghiệm với một giải thuật phát hiện chuỗi con bất thường trong dữ liệu chuỗi thời gian nổi tiếng là giải thuật HOT SAX.

## Cấu trúc đề cương

Đề cương chia làm 5 chương:

**Chương 1:** Giới thiệu về bài toán và nhiệm vụ đề tài.

**Chương 2:** Các cơ sở lý thuyết.

**Chương 3:** Giới thiệu các công trình liên quan.

**Chương 4:** Nội dung nghiên cứu và kế hoạch là việc.

# CƠ SỞ LÝ THUYẾT

Chương này sẽ trình bày các định nghĩa, các giải thuật phân đoạn, các phương pháp đo khoảng cách, các giải thuật *thu giảm số chiều* (dimensionality reduction) và *rời rạc hóa* (discretization) dữ liệu thường được sử dụng trong các công trình liên quan đến bài toán tìm chuỗi con bất thường.

## Các định nghĩa

**Định nghĩa 1:** Một*chuỗi thời gian* (Time Series) chiều dài m là một tập hợp có thứ tự gồm *m* giá trị thực. Ta ký hiệu chuỗi thời gian là *T = x1, x2, …, xm* với *xi* là các số thực, *m* là một số nguyên.

**Định nghĩa 2:** *Chuỗi con* (subsequence) *C* có chiều dài *n* của một chuỗi thời gian *T* có chiều dài *m* (*m ≤ n*) là một đoạn các giá trị liên tục nằm trong *T*. Ta ký hiệu *C = xp, xp+1, …, xp+n-1*, với *1≤ p≤ m-n+1*. Đôi khi người ta còn ký hiệu *C* bằng *(sp, ep+n-1)*, với *sp = xp* và *ep+n-1 = xp+n-1*.

**Định nghĩa 3:** *Hàm khoảng cách* (distance function) *Dist()* của hai chuỗi thời gian *C* và *M* là một hàm số nhận *C* và *M* làm giá trị nhập và tạo ra một số thực dương *d* là khoảng cách của *C* và *M*. Để thuận tiện cho các hoạt động tính toán trên chuỗi thời gian thì hàm khoảng cách *Dist()* phải là một hàm số có tính chất đối xứng, nghĩa là *Dist(C,M) = Dist(M,C).*

**Định nghĩa 4:** Cho một số nguyên *k > 0*, một tập hợp *D* gồm tất cả các chuỗi con của chuỗi thời gian *T*, *P* là một phần tử thuộc *D*. Khoảng cách thứ *k* của *P*, ký hiệu *k-dist(P)* là khoảng cách của *P* và *Q* với *Q* thuộc *D* và thỏa mãn hai tính chất sau.

1. Tồn tại ít nhất *k* phần tử *Q’* thuộc *D* sao cho *Dist(D, Q’) ≤ Dist(D,Q).*
2. Tồn tại nhiều nhất *k-1* phần tử *Q’* thuộc *D \{Q}* sao cho *Dist(D, Q’) < Dist(D,Q).*

**Định nghĩa 5:** Cho một tập hợp *D* gồm các chuỗi con của một chuỗi thời gian *T*, ta kí hiệu *k-dist(D)* là tập hợp các khoảng cách thứ *k* của các chuỗi con trong *D*, *median(k-dist(D))* là trung vị của các giá trị trong *k-dist(D)*. *Hệ số bất thường* (Anomaly factor) theo khoảng cách thứ *k* của một chuỗi *P* thuộc *D* là tỉ số giữa *k-dist(P)* và *median(k-dist(D)).*

**Định nghĩa 6:** Hai chuỗi con *P* và *Q* có độ dài *n* của chuỗi thời gian *T* gọi là *khớp không tầm thường* (non-self match) (theo định nghĩa của Leng và các cộng sự trong [10]) nếu *Dist(P, Q) ≥ e* hoặc *|p - q| ≥ n*, với *e* là một số thực do người dùng quy định, *p* là vị trí bắt đầu của chuỗi *P* và *q* là vị trí bắt đầu của *Q* trong *T*.

Một số tác giả khác như E. Keogh và các cộng sự trong [5] hay Y. Bu và các cộng sự trong [15] khẳng định hai chuỗi *P*, *Q* là khớp không tầm thường nếu *|p - q|≥ n*.

## Các giải thuật phân đoạn

Để giải quyết bài toán tìm chuỗi con bất thường, hướng tiếp cận thường thấy là *phận đoạn* (segmentation) chuỗi thời gian ra nhiều chuỗi con và tiến hành so sánh các chuỗi con này để tìm ra các chuỗi con bất thường. Có nhiều giải thuật phân đoạn khác nhau nhưng E. Keogh và các cộng sự trong [7] đã gom chúng vào 3 lớp giải thuật cơ bản là *giải thuật cửa sổ trượt* (Sliding window algorthim), *giải thuật từ trên xuống* (Top-down algorithm) và *giải thuật từ dưới lên* (Botton-up algorithm).

### Giải thuật cửa sổ trượt

Giải thuật cửa sổ trượt sử dụng điểm đầu tiên của chuỗi thời gian làm *điểm mốc* (anchor) và xấp xỉ các điểm dữ liệu về bên phải bằng một đa thức, đến tại một điểm dữ liệu thứ *i* nào đó của chuỗi thời gian mà sai số xấp xỉ lớn hơn một giá trị mà người dùng quy định thì dừng, đoạn từ điểm mốc đến điểm *i-1* tạo thành một chuỗi con. Điểm mốc được dịch chuyển đến điểm *i* và giải thuật được lặp lai cho đến khi toàn bộ chuỗi thời gian đã được chia thành các đoạn nhỏ. Bảng 2.1 là bảng tóm tắt các ký hiệu dùng trong mục 2.2 này. Mã giả của giải thuật cửa sổ trượt được thể hiện dưới hình 2.1.

Giải thuật cửa sổ trượt là một giải thuật tối ưu cục bộ, nó không có khả năng nhìn thấy tổng thể toàn bộ chuỗi thời gian mà chỉ có thể xấp xỉ cục bộ một đoạn của chuỗi thời gian trong vùng cửa sổ với kích thước *n* cho trước. Tuy nhiên, giải thuật này là một giải thuật *trực tuyến* (online), nó có khả năng thích nghi tốt đối với *các chuỗi thời gian dạng luồng* (streaming time series). Khi các điểm dữ liệu mới được thêm vào, giải thuật không cần xấp xỉ lại toàn bộ chuỗi thời gian mà chỉ cần trượt và xấp xỉ trên các điểm dữ liệu mới và tạo thêm các chuỗi con mới.

|  |  |
| --- | --- |
| T | Chuỗi thời gian *T= t1, t2, t3,…, tn* |
| T[a:b] | Một chuỗi con của *T* bắt đầu từ điểm thứ *a* và kết thúc ở điểm thứ *b* |
| Seg\_TS | Tập hợp các đoạn xấp xỉ các chuỗi con của *T*, đoạn thứ *i* là Seg\_TS(*i*) |
| create\_segment(T) | Hàm số nhận một chuỗi thời gian làm tham số và tạo thành một đoạn xấp xỉ của của chuỗi thời gian đó |
| calculate\_error(T) | Hàm số tính toán sai số khi xấp xỉ một chuỗi thời gian |

Bảng 2.1. Các ký hiệu sử dụng trong mục 2.2.

**Algorithm** *Seg\_TS = Sliding\_Window(T , max\_error)*

*anchor = 1;*

**while** *not finished segmenting time series*

*i = 2;*

**while** *calculate\_error(T[anchor: anchor + i ]) < max\_error*

*i = i + 1;*

**end**;

*Seg\_TS = concat(Seg\_TS, create\_segment(T[anchor: anchor + (i-1)]);*

*anchor = anchor + i;*

**end**;

Hình 2.1. Giải thuật cửa sổ trượt.

### Giải thuật từ trên xuống

Giải thuật từ trên xuống bắt đầu bằng việc xem toàn bộ chuỗi thời gian là một đoạn duy nhất và tiến hành chia đoạn đó ra làm hai phần tại *điểm chia tốt nhất* (best location). Hai đoạn mới được tạo ra sẽ được kiểm tra xem sai số xấp xỉ của chúng có vượt quá sai số cho phép không, nếu có chúng sẽ được phân chia tiếp. Giải thuật lặp lại một cách đệ quy cho tới khi các đoạn con đều có sai số xấp xỉ nhỏ hơn ngưỡng cho phép. Hình 2.2 là mã giả cho giải thuật này. Hàm số *improve\_splitting\_here(T,i)* trả về sai số xấp nếu chia chuỗi *T* thành 2 đoạn tại điểm *i*.

Giải thuật từ trên xuống có khả năng nhìn được toàn bộ chuỗi thời gian và có thể đạt được tối ưu toàn cục. Tuy nhiên, khi có các điểm dữ liệu mới thêm vào, giải thuật phải chạy và phân đoạn lại cho toàn bộ chuỗi thời gian. Điều này hạn chế việc áp dụng giải thuật cho các chuỗi thời gian dạng luồng.

**Algorithm** *Seg\_TS = Top\_Down(T , max\_error)*

*best\_so\_far = inf;*

**for** *i = 2* **to** *length(T) - 2 // Find best place to split the time series.*

*improvement\_in\_approximation = improvement\_splitting\_here(T,i);*

**if** *improvement\_in\_approximation < best\_so\_far*

*breakpoint = i;*

*best\_so\_far = improvement\_in\_approximation;*

**end**;

**end**;

*// Recursively split the left segment if necessary.*

**if** *calculate\_error(T[1:breakpoint]) > max\_error*

*Seg\_TS = Top\_Down(T[1: breakpoint]);*

**end**;

*// Recursively split the right segment if necessary.*

**if** *calculate\_error( T[breakpoint + 1:length(T)] ) > max\_error*

*Seg\_TS = Top\_Down(T[breakpoint + 1: length(T)]);*

**end**;

Hình 2.2. Giải thuật từ trên xuống.

### Giải thuật từ dưới lên

Giải thuật từ dưới lên ngược lại với giải thuật từ trên xuống. Giải thuật bắt đầu bằng việc xấp xỉ 2 điểm kế cận nhau vào một đoạn. Như vậy với một chuỗi thời gian có chiều dài *m* thì ta có *m/2* đoạn. Sau đó hai đoạn kề nhau sẽ được trộn lại nếu chi phí trộn hai đoạn là nhỏ nhất so với việc trộn các đoạn khác. Giải thuật được lặp lại cho đến khi chi phí trộn nhỏ nhất lớn hơn hay bằng ngưỡng cho phép. Trong quá trình tính toán giải thuật dùng một danh sách *merge\_cost* chứa tất cả các chi phí để trộn các đoạn liền kề với nhau. Sau khi trộn hai đoạn với nhau giải thuật sẽ tính toán lại chi phí khi trộn đoạn mới với hai đoạn ở hai bên và cập nhấp lại danh sách *merge\_cost*. Hình 2.3 là mã giả cho giải thuật này. Giải thuật từ dưới lên cũng có khả năng nhìn được toàn bộ chuỗi thời gian và có thể đạt được tối ưu toàn cục. Tuy nhiên, cũng giống giải thuật từ trên xuống khi có các điểm dữ liệu mới thêm vào, giải thuật phải chạy và phân đoạn lại cho toàn bộ chuỗi thời gian. Do đó giải thuật này khó áp dụng cho các chuỗi thời gian dạng luồng.

**Algorithm** *Seg\_TS = Bottom\_Up(T , max\_error)*

**for** *i = 1 : 2 : length(T) // Create initial fine approximation*.

*Seg\_TS = concat(Seg\_TS, create\_segment(T[i: i + 1 ]));*

**end**;

**for** *i = 1 : length(Seg\_TS) – 1 // Find cost of merging each pair of*

*segments.*

*merge\_cost(i) = calculate\_error([merge(Seg\_TS(i), Seg\_TS(i+1))]);*

**end;**

**while** *min(merge\_cost) < max\_error // While not finished.*

*index = min(merge\_cost); // Find “cheapest” pair to merge.*

*Seg\_TS(index) = merge(Seg\_TS(index), Seg\_TS(index+1))); // Merge them.*

*delete(Seg\_TS(index+1)); // Update records.*

*merge\_cost(index)=calculate\_error(merge(Seg\_TS(index), Seg\_TS(index+1)));*

*merge\_cost(index-1)=calculate\_error(merge(Seg\_TS(index-1), Seg\_TS(index)));*

**end**;

Hình 2.3. Giải thuật từ dưới lên.

## Các phương pháp tính khoảng cách

Các phương pháp tính khoảng cách được sử dụng để đánh giá mức độ khác biệt giữa hai chuỗi thời gian. Mục này sẽ trình bày hai phương pháp tính khoảng cách phổ biến là phương pháp dùng công thức tính *khoảng cách Euclid* (Euclidean distance) và phương pháp *xoắn thời gian động* (Dynamic time warping).

### Công thức tính khoảng cách Euclid

Công thức tính khoảng cách Euclid được phát biểu như sau: với hai chuỗi thời gian *Q* và *C* có cùng chiều dài *n* ta có *Dist(Q,C)* =  với *qi* thuộc *Q* và *ci* thuộc *C*. Công thức tính khoảng cách này chỉ áp dụng được cho hai chuỗi thời gian có cùng chiều dài.

Trong một số trường hợp, các chuỗi thời gian có hình dạng giống nhau nhưng bị lệch nhau một khoảng theo trục tung như trong hình 2.4, hai chuỗi thời gian rất giống nhau về hình dạng nhưng sẽ có khoảng cách lớn nếu tính theo công thức (2.3.1). Một số tác giả đã hiệu chỉnh công thức Euclid để loại bỏ sự lệch theo trục tung của hai chuỗi thời gian khỏi công thức tính khoảng cách. C.D Truong cùng các cộng sự trong [3] và Lee cùng các cộng sự trong [12] sử dụng công thức *khoảng cách Euclid cực tiểu* (Minimum Euclidean Distance) để loại ra sự khác biệt theo trục tung của hai chuỗi thời gian. Công thức này được tính như sau:

*Dist(Q,C) =*  , ở đây *b* là một số thực được tính bằng công thức

*b =* 

K. Chan và các cộng sự trong [9] sử dụng phương pháp lấy mỗi điểm của mỗi chuỗi thời gian trừ đi giá trị trung bình tương ứng của từng chuỗi trước khi áp dụng cộng thức Euclid. Cách tính này được cho bởi công thức sau:

*Dist(Q,C) =*

với  và 

Phương pháp tính khoảng cách bằng công thức Euclid đơn giản, dễ hiện thực và cố độ phức tạp tính toán tuyến tính *O(n)* nên được sử dụng nhiều trong các bài toán khai phá dữ liệu chuỗi thời gian. Tuy nhiên công thức này không thể áp dụng để tính khoảng cách của hai chuỗi thời gian có độ dài khác nhau.



Hình 2.4. Hai chuỗi thời gian hình dạng giống nhau nhưng bị lệch theo trục tung

### Phương pháp xoắn thời gian động

Phương pháp xoắn thời gian động là một phương pháp tính khoảng cách phổ biến được dùng nhiều trong các bài toán phân lớp và gom cụm dữ liệu chuỗi thời gian. Phương pháp này có ưu điểm là có thể tính được khoảng cách giữa hai chuỗi thời gian có độ dài khác nhau hay có biên độ dao động khác nhau. Ý tưởng chính của phương pháp này là nó cố gắng tìm một *đối sánh* (matching) tối ưu giữa các điểm của hai chuỗi thời gian để tìm ra khoảng cách nhỏ nhất giữa chúng. Hình 2.5 bên dưới minh họa cho sự khác nhau khi so sánh từng điểm của hai chuỗi thời gian để tính khoảng cách. Công thức tính khoảng cách Euclid đối sánh giá trị của các điểm có cùng hoành độ (cùng thời điểm) với nhau trong khi phương pháp xoắn thời gian động đối sánh các điểm sao cho tối ưu nhất. Dùng phương pháp xoắn thời gian động thì một điểm của chuỗi này có thể đối sánh với nhiều điểm của chuỗi kia nên có thể áp dụng để tính khoảng cách cho các chuỗi có độ dài khác nhau.

Phương pháp xoắn thời gian động được hiện thực như sau. Gọi chuỗi thời gian thứ nhất là *A*, có chiều dài *m*, ký hiệu *A = a1,a2,…,am*. Gọi *B* là chuỗi thời gian thứ 2 có chiều dài *n*, ký hiệu *B = b1,b2,…,bn*.Ta xây dựng một ma trận đường đi *M* với *m* hàng và *n* cột. Mỗi phần tử *(i,j)* của ma trận *M* tương ứng với khoảng cách của phần tử *ai* và *bj .* Trong các chuỗi thời gian thông thường, *ai, bj* là các số thực và khoảng cách của chúng được tính bằng *d(ai,bj) = |ai - bj|* . Đường xoắn thời gian *D* là một chuỗi *u1, u2,…,uk* với *ui* tương ứng với các một phần tử (*ai bj*) của ma trận *M*. Chuỗi *D* phải thỏa mãn các điều kiện sau:

*i)* *Ràng buộc điểm cuối* (endpoint constraint): điểm bắt đầu và kết thúc của đường đi D phải trùng với điểm đầu và cuối của ma trận *M*, nghĩa là *u1 = (a1,b1), uk = (am,bn)* [14].

*ii)* *Ràng buộc tính liên tục* (continuity constraint): nếu *uk = (ai, bj), uk+1 = (ai+1, bj+1)* thì *ai - ai+1 ≤ 1* và *bi - bi+1 ≤ 1* [14].

*iii)* *Ràng buộc tính đơn điệu* (Monotonicity constraint): nếu *uk = (ai, bj), uk+1 = (ai+1, bj+1)* thì *ai ≤ ai+1* và *bi ≤ bi+1* [14].

*iv)* *Ràng buộc về độ nghiêng* (slope constraint): đường xoắn thời gian không được quá dốc hay quá cạn, nghĩa là một điểm trên một chuỗi thời gian này không được đối sánh với quá nhiều điểm trên chuỗi thời gian khác. Điều này được thực hiện bằng hai tham số *x,y*. Nếu đã đi được *x* bước liên tục theo hướng ngang của ma trận *M* thì phải thực hiện một bước theo hướng dọc và ngược lại nếu đã đi *y* bước theo hướng dọc thì phải thực hiện một bước theo hướng ngang [2].



Hình 2.5. (a) Đo khoảng cách bằng công thức Euclid. (b) Đo khoảng cách bằng phương pháp xoắn thời gian động.

Khoảng cách giữa hai chuỗi thời gian *A*, *B* là độ dài ngắn nhất của đường xoắn thời gian *D* ký hiệu *DTW(A,B).* Mỗi điểm trên *D* là một đối sánh giữa một điểm trên chuỗi thời gian *A* và một điểm trên chuỗi thời gian *B*. Để tính độ dài ngắn nhất của *D*, người ta dùng phương pháp quy hoạch động.



Ở đây  là khoảng cách tích lũy được tính đệ quy như sau:



Với 

Giá trị *p* được chọn tùy vào ứng dụng. Đối với chuỗi thời gian bình thường có giá trị theo thời gian là các số thực thì *p* thường được chọn là 1 hoặc 2. Khoảng cách của hai chuỗi *A*, *B* là *Dist(A,B) = DTW(m,n)*.

Với cách tính như trên *DTW(m,n)* đã được chứng minh là khoảng cách tích lũy tối thiểu của các đường xoắn thời gian. Độ phức tạp của giải thuật là *O(m\*n)* do phải duyệt qua ma trận *M* có kích thước *m-n*. Hình 2.6 là một minh họa về cách tính *DTW*, các ô tô đậm là các điểm *(i,j)* mà đường xoắn thời gian đi qua.



Hình 2.6. Ma trận xoắn thời gian và đường xoắn thời gian.

Phương pháp tính khoảng cách bằng phương pháp xoắn thời gian động có độ phức tạp cao hơn phương pháp tính bằng khoảng cách Euclid (*O(m\*n)* so với *O(n)*). Do đó nó khó có thể áp dụng đối với các chuỗi thời gian có kích thước lớn hay những chuỗi dạng luồng, khi mà dữ liệu mới liên tục cập nhập đòi hỏi các bước tính toán phải thực hiện nhanh.

## Các phương pháp thu giảm số chiều và rời rạc hóa dữ liệu

Mục này sẽ trình bày các phương pháp thu giảm số chiều và rời rạc hóa dữ liệu thường dùng trong các cộng trình lên quan đến bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian: phương pháp *xấp xỉ PAA* (Piecewise Aggregate Approximation), phương pháp *biến đổi dạng sóng Haar* (Haar Wavelet Transform), phương pháp *biểu diễn SAX* (Sympolic Aggregate Approximation).

### Phương pháp xấp xỉ PAA.

Phương pháp xấp xỉ PAA được sử dụng để thu giảm số chiều của một chuỗi thời gian. Gọi một chuỗi thời gian *X = x1, x2, …, xn*, phương pháp xấp xỉ PAA biến đổi chuỗi *X* thành chuỗi *X’= x’1, x’2, …, x’N* với *1≤N≤n.* Thông thường, *N* là một thương số của *n*. Các điểm của chuỗi *X’* được tính bằng công thức sau:



Từ công thức (2.4.1) có thể thấy phương pháp xấp xỉ PAA thu giảm số chiều của chuỗi X bằng cách chia chuỗi này thành *N* đoạn bằng nhau và lấy trung bình các giá trị trong mỗi đoạn. Các giá trị trung bình này tạo thành một vector là một biểu diễn của chuỗi *X* với số chiều đã được thu giảm. Ví dụ cho một chuỗi thời gian *X* = -1, -2, -1, 0, 2, 1, 1, 0. Để thu giảm chuỗi này thành một chuỗi *X’* có độ dài 2, ta chia chuỗi *X* thành 2 đoạn, mỗi đoạn 4 phần tử và tính giá trị trung bình mỗi đoạn:

*X’* = *mean*(-1,-2,1,0), *mean*(2,1,1,0) = -1, 1.

Phương pháp xấp xỉ PAA dễ hiểu và dễ hiện thực nên thường được sử dụng nhiều trong các công trình liên quan đến bài toán khai phá dữ liệu chuỗi thời gian [6][14][8][11]. Hình 2.7 là một minh họa cho phương pháp xấp xỉ PAA, một chuỗi thời gian có được thu giảm về một vector có 8 chiều.



Hình 2.7. Phương pháp xấp xỉ PAA thu giảm số chiều của một chuỗi thời gian.

Thông thường trước khi áp dụng phương pháp xấp xỉ PAA, người ta thường chuẩn hóa chuỗi thời gian thành một chuỗi có trung bình bằng 0 và độ lệch chuẩn bằng 1 [8].

### Phương pháp biến đổi dạng sóng Haar

Phương pháp biến đổi dạng sóng Haar được đề xuất bởi K. Chan và các cộng sự trong bài báo [9] để phục vụ cho việc *đánh chỉ mục* (indexing) các chuỗi thời gian. Phương pháp này có 3 ưu điểm chính: (1) có thể xấp xỉ một chuỗi thời gian với nhiều mức *phân giải* (resolution) khác nhau,(2) có độ phức tạp tính toán *O(n)* với *n* là chiều dài của chuỗi thời gian [9], (3) bảo toàn khoảng cách Euclid [9].

Phương pháp biến đổi dạng sóng Haar thực hiện việc tính trung bình và *lấy hiệu* (diffrencing) trên các giá trị kề nhau của một *hàm thời gian rời rạc* (discrete time function). Hình 2.8 là minh họa cho quá trình thực hiện biến đổi Haar trên hàm *f(x) =* (9 7 3 5), cột *Resolution* chứa các mức phân giải, cột *Averages* chứa các giá trị trung bình cho từng mức phân giải, cột *Coefficients* chứa các hệ số cho từng mức phân giải.



Hình 2.8 Biến đổi dạng sóng Haar cho hàm *f(x) =* (9 7 3 5).

Mức phân giải thứ tư là mức phân giải đầy đủ của hàm *f(x)*. Ở mức phân giải thứ hai cặp giá trị trung bình (8 4) được tính lần lượt bằng cách tính trung bình hai cặp giá trị (9 7) và (3 5), cặp hệ số (1 -1) được tính lần lượt bằng cách lấy hiệu các cặp giá trị (9 7) và (3 5) rồi chia kết quả cho 2. Quá trình này được lặp lại cho đến khi đạt được mức phân giải thứ nhất. Khi đó biến đổi dạng sóng Haar của hàm *f(x)* là



Ở đây *c* chính là giá trị trung bình của các phần tử trong *f(x)*. Các giá trị trung bình ở các mức phân giải cao hơn có thể tính được bằng cách cộng hoặc trừ các hệ số vào các giá trị trung bình ở mức thấp. Ví dụ giá trị trung bình ở mức phân giải thứ hai trong hình 2.8 là (8 4) = (6+2 6-2) với 6 và 2 lần lượt là giá trị trung bình và hệ số ở mức phân giải thứ nhất.

Biển đổi dạng sóng Haar có thể được hiện thực bằng một chuỗi các phép nhân ma trận. Ví dụ để tìm biến đổi dạng sóng Haar của một chuỗi *x* có chiều dài 4 (được biểu diễn bằng một ma trận 1x4) đầu tiên ta nhân chuỗi *x* với ma trận *H* để tìm ma trận *w* như hình 2.9.



Hình 2.9. Hiện thực phương pháp biến đổi dạng sóng Haar bằng phép nhân ma trận.

Sau phép nhân đầu tiên các hệ số biến đổi dạng sóng Haar ,  và các giá trị trung bình ,  được tìm thấy xem kẽ trong ma trận kết quả *w*. Các giá trị trung bình được đưa vào ma trận  và tiếp tục thực hiện phép nhân giữa ma trận *H* với ma trận *x’*. Giải thuật được lặp lại cho đến khi chỉ có 1 giá trị khác 0 trong *x’*. Trong ví dụ này thì giải thuật sẽ dừng sau phép nhân thứ 2.

Các chuỗi thời gian có thể biến đổi về dạng sóng Haar bằng cách lấy trung bình và lấy hiệu các giá trị kề nhau. Hiện thực chi tiết phép biến đổi có thể khác nhau tùy vào các điều kiện chuẩn hóa khác nhau. A.W. Fu và các cộng sự trong bài báo [1] hiện thực giải thuật biến đổi dạng sóng Haar theo *điều kiện trực giao* (orthonormal condition). Mã giả của giải thuật được cho trong hình 2.10, giải thuật nhận một chuỗi thời gian có chiều dài *w* làm đầu vào và trả về một vector *output* chứa các hệ số biến đổi dạng sóng Haar của chuỗi này.

*// Initializtion*

*w = size of input vector*

*output vector = all zero*

*dummy vector = all zero*

*//start the conversion*

**while** *w> 1* **do**

*w = w/2*

**for** *i =0; i<w; i++* **do**

*dummy vector[i]= (input vector[2∗i]+input vector[2∗i+1])/*

*dummy vector[i + w]= (input vector[2∗i]−input vector[2∗i+1])/*

**end for**

**for** *i =0; i< (w ∗ 2); i++* **do**

*output vector[i]= dummy vector[i]*

**end for**

**end while**

Hình 2.10. Giải thuật biến đổi dạng sóng Haar của Fu và các cộng sự

### Phương pháp biểu diễn SAX

Phương pháp biểu diễn SAX, được đề xuất bởi J. Lin và các cộng sự trong bài báo [8], là một phương pháp biểu diễn chuỗi thời gian thành các chữ cái rời rạc.

Để xây dựng biểu diễn SAX của một chuỗi thời gian *C* có độ dài *n* ta cần thực hiện 2 bước.

Bước thứ nhất: thực hiện chuẩn hóa chuỗi ban đầu thành một chuỗi thời gian có trung bình bằng 0 và độ lệch chuẩn bằng 1 sau đó thu giảm số chiều chuỗi thời gian này thành một vector *T* có chiều dài *w < n* bằng phương pháp xấp xỉ PAA.

Bước thứ hai: xác định các *điểm chia* (breakpoints). Các điểm chia là một danh sách các điểm có thứ thự *B = B1, B2, …, Ba-1* sau cho phần diện tích dưới đường cong *N(0,1)* Gauss từ *Bi* đến *Bi+1* bằng *1/a* với *a* là một số nguyên. Các điểm chia này thường được xác định trong một bảng thống kê. Hình 2.11 là bảng các điểm chia cho giá trị *a* từ 3 đến 10. Các giá trị của vector *T* sẽ được ánh xạ thành các chữ cái theo luật sau:

Gọi *alphai* là chữ cái thứ *i* trong bảng chữ cái tiếng Anh (*alpha1 = a, alpha2 = b*,…) thì một phần tử *ti* của vector *T* sẽ được ánh xạ thành *alphaj* nếu và chỉ nếu *Bj-1* ≤ *ti* < *Bj*, với *B0 =* -∞ *và Ba =* +∞.

Dãy các chữ cái tạo thành từ vector *T* theo luật trên được gọi là một *từ* (word) và nó là biểu diễn SAX của chuỗi thời gian C. Hình 2.12 bên dưới là minh họa cho biểu diễn SAX của một chuỗi thời gian với *a* = 3 và 2 điểm chia *B1* = -0.43, *B2* = 0.43.



Hình 2.11. Bảng các điểm chia với *a* từ 3 đến 10



Hình 2.12. Chuỗi thời gian được biểu diễn thành chuỗi *cbccbaab*

Trong bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian, phương pháp biểu diễn SAX được dùng để xây dựng các cấu trúc chỉ mục hỗ trợ cho việc tìm kiếm [8] hay dùng để rút ra các luật văn phạm (grammar rule) hỗ trợ cho việc xác định các chuỗi con bất thường [11].

# GiỚI THIỆU CÁC CÔNG TRÌNH LIÊN QUAN

Bài toán tìm kiếm chuỗi con bất thường trong chuỗi thời gian có nhiều ứng dụng trong thực tế nên rất được quan tâm bởi các nhà nghiên cứu. Có hai hướng tiếp cận thường thấy ở bài toán này. Hướng thứ nhất là cải tiến giải thuật *vét cạn* (brute force) bằng cách xây dựng các cấu trúc hỗ trợ cho việc sắp xếp thứ tự các chuỗi con trong hai vòng lặp nhằm tăng tốc độ cho giải thuật [5][15][1][11]. Cách tiếp cận thứ hai là thực hiện phân đoạn chuỗi thời gian thành các chuỗi con rồi tiến hành đánh giá khoảng cách từng cặp chuỗi con để tìm ra các chuỗi con bất thường [10]. Chương này sẽ trình bày các công trình liên quan đến phân đoạn chuỗi thời gian và các công trình tìm kiếm chuỗi con bất thường thuộc hai hướng tiếp cận nói trên.

## Các công trình liên quan đến phân đoạn

### Giải thuật phân đoạn từ trên xuống cải tiến của D. Lemire

Các giải thuật phân đoạn được phân loại thành ba loại chính: phương pháp cửa số trượt, phương pháp từ trên xuống hay phương pháp từ dười lên. Mỗi loại đều có ưu nhược điểm riêng. D. Lemire trong bài báo [4] đã cải thiện giải thuật từ trên xuống để tạo nên một giải thuật hiệu quả hơn. Tác giả gọi tổng số các *hệ số hồi quy độc lập* (regressor) của các đa thức dùng để xấp xỉ trên mỗi đoạn là *độ phức tạp mô hình* (model complexity). Ví dụ nếu một chuỗi thời gian được phân thành 3 đoạn, đoạn thứ nhất xấp xỉ bằng một đoạn thẳng *y = a* với *a* là hằng số, đoạn thứ hai và thứ ba được xấp xỉ lần lược bằng hai đoạn thẳng *y = a1x + b1* và *y = a2x + b2* thì độ phức tạp mô hình của cách phân đoạn này là 1 + 2 + 2 = 5. Độ phức tạp mô hình càng lớn thì phương pháp phân đoạn càng phức tạp. Tác giả nhận ra rằng đôi khi nếu có thể chia một đoạn được xấp xỉ bằng một đa thức bậc cao thành hai đoạn được xấp xỉ bằng đa thức bậc nhỏ hơn miễn là độ phức tạp mô hình không đổi thì có khả năng giảm sai số xấp xỉ. Giải thuật của Lemire như sau: ban đầu áp dụng phương pháp từ trên xuống sau đó với mỗi đoạn, tìm một điểm chia mà có thể chia đoạn đó thành hai đoạn con được xấp xỉ bằng đa thức có bậc nhỏ hơn với điều kiện không làm tăng độ phức tạp mô hình và tổng sai số xấp xỉ của hai đoạn con phải nhỏ hơn sai số xấp xỉ của đoạn cha. Hình 3.1 là mã giả cho giải thuật.

**INPUT**: *Time Series (xi, yi) of length n*

**INPUT**: *Bound on Polynomial degree N and model complexity k*

**INPUT**: *Function E(p, q, d) computing ﬁt error with poly.in range [xp, xq)*

*S empty list*

*d ← N − 1*

*S ← (0, n, d,E(0, n, d))*

*b ← k − d*

**while** *b − d ≥ 0* **do**

*ﬁnd tuple (i, j, d, e) in S with maximum last entry*

*ﬁnd minimum of E(i, l, d) + E(l, j, d) for l = i +1, . . . , j*

*remove tuple (i, j, e) from S*

*insert tuples (i, l, d,E(i, l, d)) and (l, j, d,E(l, j, d)) in S*

*b ← b − d*

**for** *tuple (i, j, q, e)* **in** S **do**

*ﬁnd minimum m of E(i, l, d′) + E(l, j, q − d′− 1) for*

*l = i + 1, . . . , j and 0 ≤ d′≤ q − 1*

**if** *m < e* **then**

*remove tuple (i, j, q, e) from S*

*insert tuples (i, l, d′,E(i, l, d′)) and (l, j, q − d′−1,E(l, j, q − d′− 1)) in S*

*S contains the segmentation*

Hình 3.1. Mã giả cho giải thuật của D. Lemire.

Giải thuật trả về một tập hợp *S* chứa các đoạn của chuỗi thời gian. Mỗi đoạn trong giải thuật được biểu diễn bằng một bộ *(i, j, d, e)*, với *i* là điểm bắt đầu của đoạn, *j* là điểm kết thúc, *d* là bậc của đa thức xấp xỉ và *e* là sai số xấp xỉ. Giải thuật của D. Lemire được kiểm tra bằng thực nghiệm và cho kết quả phân đoạn khá tốt. Tuy nhiên giải thuật này vẫn còn mang nhược điểm của phương pháp từ trên xuống là không thích nghi được với các chuỗi thời gian mà dữ liệu mới được thêm vào liên tục. Mỗi khi có dữ liệu mới thì giải thuật phải chạy và phận đoạn lại từ đầu.

### Giải thuật phân đoạn SWAB

E. Keogh và các cộng sự trong bài báo [7] đề ra một giải thuật phân đoạn mới kết hợp phương pháp từ dưới lên và phưng pháp cửa sổ trượt với mục đích tận dụng khả năng tối ưu toàn cục của phương pháp từ dưới lên và khả năng thích nghi với các chuỗi thời gian động của phương pháp cửa sổ trượt. Các tác giả gọi giải thuật mới là *SWAB* (**S**liding **W**indow **a**nd **B**ottom-up). Ý tưởng chính giải thuật là sử dụng một *vùng đệm* (buffer) có kích thước *w*. Kích thước của vùng đệm này được khởi tạo đủ lớn để có thể chứa được khoảng 5, 6 đoạn. Giải thuật từ dưới lên sẽ áp dụng để phân đoạn phần chuỗi thời gian trong vùng đệm. Sau khi phân đoạn xong, chuỗi con bên trái nhất của vùng đệm được chọn làm một chuỗi con của chuỗi thời gian và bị xóa khỏi vùng đệm. Dữ liệu mới được đọc vào vùng đệm bằng cách xấp xỉ dần các điểm ở vùng ngoài bên phải của vùng đệm bằng một đa thức giống như trong phương pháp cửa sổ trượt. Giải thuật được lặp lại cho đến khi chuỗi thời gian được phân đoạn xong. Khi có dữ liệu mới thì giải thuật chỉ cần chạy lại trên các điểm dữ liệu mới nên giải thuật có thể áp dụng cho các chuỗi thời gian động. Hình 3.2 là mã giả cho giải thuật này. Hàm *BEST\_LINE* là hàm số xấp xỉ các điểm của chuỗi thời gian bằng đa thức.

**Algorithm** *Seg\_TS = SWAB(max\_error, seg\_num) // seg\_num is integer 5 or 6*

*read in w number of data points // Enough to approximate seg\_num of segments.*

*lower\_bound = w / 2;*

*upper\_bound = 2 \* w;*

**while** *data at input*

*T = Bottom\_Up(w, max\_error) // Call the classic Bottom-Up algorithm*

*Seg\_TS = CONCAT(SEG\_TS, T(1)); // Sliding window to the right.*

*w = TAKEOUT(w, w’); // Deletes w’ points in T(1) from w.*

**if** *data at input // Add w” points from BEST\_LINE() to w.*

*w = CONCAT(w, BEST\_LINE(max\_error));*

*{check upper and lower bound, adjustment if necessary}*

**else**  *// flush approximated segments from buffer.*

*Seg\_TS = CONCAT(SEG\_TS, (T – T(1)))*

**end**;

**end**;

**Function** *S = BEST\_LINE(max\_error) //returns S points to approximate*

**while** *error ≤ max\_error // next potential segment.*

*read in one additional data point, d, into S*

*S = CONCAT(S, d);*

*error = approx\_segment(S);*

**end while**;

**return** S;

Hình 3.2. Giải thuật SWAB

## Các công trình về tìm kiếm chuỗi con bất thường

### Giải thuật HOT SAX

E. Keogh và các cộng sự trong [5] xây dựng một giải thuật để tìm kiếm *chuỗi con bất thường nhất* (discord) có kích thước *n* của một chuỗi thời gian. Giải thuật này được gọi là giải thuật HOT SAX vì nó sử dụng biểu diễn SAX của các chuỗi con để tăng tốc độ hội tụ của giải thuật.

Một chuỗi con gọi là chuỗi con bất thường nhất nếu khoảng cách của nó so với chuỗi con khớp không tầm thường gần nó nhất là lớn nhất. Tương tự, chuỗi con bất thường thứ *k* là chuỗi con có khoảng cách đến chuỗi con khớp không tầm thường gần nó nhất lớn thứ *k* và thỏa mãn điều kiện sau: gọi *p* là điểm bắt đầu của chuỗi đang xét và *pi* là điểm bắt đầu của chuỗi con bất thường thứ *i* với *1 ≤ i ≤ k* thì *|p-pi| ≤ n*.

Để tìm một chuỗi con bất thường có chiều dài *n* trong một chuỗi thời gian, cách đơn giản nhất là dùng giải thuật vét cạn. Giải thuật này dùng một cửa sổ trượt có kích thước *n* duyệt qua từng điểm của chuỗi thời gian để tìm chuỗi có khoảng cách chuỗi con khớp không tầm thường gần nó nhất là lớn nhất như hình 3.3.

Giải thuật vét cạn cho kết quả chính xác và đơn gian nhưng độ phức tạp tính toán là *O(m2)* với m là chiều dài của chuỗi thời gian. Do đó nó khó áp dụng được đối với các chuỗi thời gian có kích thước lớn. Tuy nhiên E. Keogh và các cộng sự nhận thấy rằng đối với mỗi chuỗi con, ta không nhất thiết phải tính được khoảng cách giữa nó đến chuỗi con khớp không tầm thường gần nó nhất mà chỉ cần ta phát hiện khoảng cách của nó đến một chuỗi con khớp không tầm thường với nó nhỏ hơn giá trị *best\_so\_far\_dist* trong giải thuật hình 3.3 thì ta lặp tức loại nó khỏi danh sách các chuỗi con bất thường ngay. Như vậy vòng lặp bên trong của giải thuật có thể kết thúc sớm. Lợi dụng nhận xét này, nếu thứ tự các chuỗi con ở vòng lặp ngoài và vòng lặp trong của giải thuật vét cạn được sắp xếp hợp lý có thể làm cho giải thuật sớm hội tụ. Giải thuật được minh họa ở hình 3.4. Ở vòng lặp ngoài nếu các chuỗi có khoảng cách đến chuỗi con khớp không tầm thường gần nhất với nó lớn được xếp các vị trí đầu và ở vòng lặp trong các chuỗi con có khoảng cách đến chuỗi con đang được chọn ở vòng lặp ngoài nhỏ được xếp ở các vị trí đầu thì ta có thể dừng vòng lặp trong chỉ sau một vài lần lặp. Trường hợp tốt nhất ta có thể đạt được độ phức tạp *O(m).*

**Function** *[dist, loc ]= Brute\_Force(T, n)*

*best\_so\_far\_dist = 0*

*best\_so\_far\_loc = NaN*

**For** *p = 1* ***to*** *|T | - n + 1 // Begin Outer Loop*

*nearest\_neighbor\_dist = infinity*

**For** *q = 1* ***to*** *|T | - n + 1 // Begin Inner Loop*

**IF** *| p – q | ≥ n // non-self match?*

**IF** *Dist ((tp,…,tp+n-1), (tq,…,tq+n-1)* *< nearest\_neighbor\_dist*

*nearest\_neighbor\_dist = Dist ((tp,…,tp+n-1), (tq,…,tq+n-1)*

**End**

**End** *// End non-self match test*

**End** *// End Inner Loop*

**IF** *nearest\_neighbor\_dist > best\_so\_far\_dist*

*best\_so\_far\_dist = nearest\_neighbor\_dist*

*best\_so\_far\_loc = p*

**End**

**End** *// End Outer Loop*

**Return***[ best\_so\_far\_dist, best\_so\_far\_loc ]*

Hình 3.3. Giải thuật vét cạn tìm chuỗi con bất thường

**Function** *[dist, loc ]= Heuristic\_Search(T, n, Outer, Inner )*

*best\_so\_far\_dist = 0*

*best\_so\_far\_loc = NaN*

**For** *Each p in T ordered by heuristic Outer // Begin Outer Loop*

*nearest\_neighbor\_dist = infinity*

**For** *Each q in T ordered by heuristic Inner // Begin Inner Loop*

**IF** *| p – q | ≥ n // non-self match?*

**IF** *Dist ((tp,…,tp+n-1), (tq,…,tq+n-1) < best\_so\_far\_dist*

**Break** *// Break out of Inner Loop*

**End**

**IF** *Dist ((tp,…,tp+n-1), (tq,…,tq+n-1) < nearest\_neighbor\_dist*

*nearest\_neighbor\_dist = Dist ((tp,…,tp+n-1), (tq,…,tq+n-1)*

**End**

**End** *// End non-self match test*

**End** *// End Inner Loop*

**IF** *nearest\_neighbor\_dist > best\_so\_far\_dist*

*best\_so\_far\_dist = nearest\_neighbor\_dist*

*best\_so\_far\_loc = p*

**End**

**End** *// End Outer Loop*

**Return***[ best\_so\_far\_dist, best\_so\_far\_loc ]*

Hình 3.4. Giải thuật cải tiến từ giải thuật vét cạn.

Để đạt được thứ tự hợp lý của các chuỗi con trong hai vòng lặp ở giải thuật trên hình 3.4, các tác giả dùng một cửa sổ trượt kích thước *n* quét qua từng điểm của chuỗi thời gian, lấy ra các chuỗi con có kích thước *n* và biểu diễn chúng thành các từ SAX. Các từ này sau đó được lưu vào hai cấu trúc dữ liệu riêng biệt. Cấu trúc thứ nhất là một *mảng* (array) mà *chỉ số* (index) của mỗi phần tử chính là vị trí của điểm đầu tiên của chuỗi con được biểu diễn bởi từ SAX lưu trong phần tử đó cùng với số lần xuất hiện của từ SAX đó. Mảng này giúp ta tìm được biểu diễn SAX của một chuỗi con và số chuỗi con có cùng biểu diễn SAX với nó nhanh chóng. Cấu trúc thứ hai là một cấu trúc *cây* (trie) mà mỗi cạnh của nó được gán một chữ cái. Nút lá của cây chứa danh sách các điểm bắt đầu của các chuỗi con có biểu diễn SAX khớp với đường duyệt cây từ gốc đến nốt đó. Cấu trúc cây này giúp ta có thể dễ dàng xác định một từ SAX biểu diễn các chuỗi con nào Hình 3.5 là một minh họa cho hai cấu trúc dữ liệu này.



Hình 3.5 . Hai cấu trúc dữ liệu hỗ trợ cho việc sắp xếp thứ tự các chuỗi con trong hai vòng lặp.

Các chuỗi con trong vòng lặp đầu của giải thuật trên hình 3.4 được sắp theo thứ tự tăng dần số lần xuất hiện của từ SAX biểu diễn chúng. Như vậy các chuỗi có biểu diễn SAX khác biệt với các chuỗi khác sẽ được duyệt đầu tiên, các chuỗi như vậy có khả năng cao là các chuỗi bất thường. Đối với thứ tự của vòng lặp bên trong, mỗi khi một chuỗi con ở vòng lặp ngoài được chọn, giải thuật sẽ duyệt qua cấu trúc cây để tìm các chuỗi con có biểu diễn SAX khớp nhiều nhất với chuỗi được chọn để duyệt trước bởi vì những chuỗi như vậy có khoảng cách đến chuỗi đang xét có nhiều khả năng nhỏ hơn giá trị *best\_so\_far\_dist*. Điều này giúp cho giải thuật nhanh chóng dừng.

Các thông số đầu vào của giải thuật bao gồm số chiều dài *w* của các từ SAX, số các chữ cái *a* và độ dài *n* của chuỗi con bất thường. E. Keogh và các cộng sự cũng đã chứng minh trong [5] rằng các thông số *a* ít ảnh hưởng đến độ chính xác và tốc độ của giải thuật. Như vậy chỉ cần xác định thông số *w* và *n* chính xác thì giải thuật HOT SAX rất hiệu quả nhưng đáng tiếc giá trị này không phải lúc nào cũng xác định được và chúng thường khác nhau đối với các bộ dữ liệu khác nhau.

### Giải thuật WAT

Giải thuật *WAT* (**w**avelet and **a**ugmented **t**rie) được đề xuất bởi Y. Bu và các cộng sự trong [15] nhằm tìm kiếm *k* chuỗi con bất thường nhất trong chuỗi thời gian. Cũng giống như giải thuật HOT SAX, giải thuật này cố gắng tăng tốc cho giải thuật hình 3.4 bằng cách sắp xếp hợp lý thứ tự các chuỗi con trong vòng lặp ngoài và vòng lặp trong. Tuy nhiên giải thuật WAT không dùng phương pháp xấp xỉ PAA mà dùng biến đổi dạng sóng Haar để biểu diễn các chuỗi con dưới dạng đa mức phân giải.

Các vector hệ số Haar của các chuỗi có con sẽ được chuẩn hóa thành các chuỗi có trung bình bằng 0 và độ lệch chuẩn bằng 1. Để giảm độ phức tạp trong việc so sánh, các chuỗi này sẽ được rời rạc hóa thành các từ SAX giống như trong giải thuật HOT SAX. Giả sử một chuỗi thời gian con *C* có kích thước *n* được biến đổi sang dạng sóng Haar thành chuỗi *C’ = {c’1, c’2, …, c’n}*. Gọi *alphai* là chữ cái thứ *i* trong bảng chữ cái tiếng Anh. Gọi *B= b1, b2, …, ba-1* là chuỗi các điểm chia sau cho phần diện tích dưới đường cong sác xuất *N(0,1)* Gauss giữa *bi, bi+1* bằng *1/a* với *b0* và *ba* tương ứng bằng -∞ và +∞. Hệ số *c’i* của biến đổi Haar sẽ được ánh xạ thành chữ cái *alphaj* nếu và chỉ nếu *bj-1 ≤ c’i≤bj.*

Khi các chuỗi con đã được ánh xạ thành các từ, các từ này sẽ được lưu trong một mảng có các *con trỏ* (pointer) trỏ về vị trí đầu tiên của các chuỗi con ban đầu. Tiếp theo giải thuật tiến hành xây dựng một cấu trúc cây với nút lá chứa các chỉ mục đến vị trí đầu tiên của các chuỗi con ban đầu. Nút gốc của cây được tạo ra gồm tất cả các từ trong mảng. Sau đó, giải thuật tiến hành lặp để *phân tách* (split) các nút lá của cây và tăng chiều cao của cây theo luật sau: để tăng dộ cao của cây lên *(h+1)*, với h là chiều cao của cây trước khi phân tách các nút lá, chữ cái thứ *(h+1)* của mỗi từ trong nút đang xét sẽ được dùng làm điều kiện rẽ nhánh để xây dựng các nhánh con mới của nút đó. Bước xây dựng cấu trúc cây sẽ dừng nếu phát hiện có tồn tại nút lá chỉ chứa 1 từ duy nhất hoặc toàn bộ chữ cái trong ký tự đã được chọn. Như vậy giải thuật WAT có thể tự xác định kích thước của các từ dựa vào bản chất của dữ liệu.



Hình 3.6. Chữ cái đầu tiên của từ được xem xét khi phân tách nút gốc của cây



Hình 3.7. Chữ cái thứ hai được xem xét khi tiến hành phân tách các nút *a, b, c*.

Hình 3.6 và 3.7 là một ví dụ là minh họa cho cách thức xây dựng cấu trúc cây này. Ở hình 3.6, khi phân tách nút gốc, ta xét chữ cái đầu tiên của các từ trong mảng bên trái, có 3 chữ cái khác nhau là *a,b,c* nên nút gốc được phân ra 3 nhánh, mỗi nhánh có cạnh tương ứng với *a, b* và *c*. Chỉ số của các từ trong mảng có chữ cái đầu tiên là *a, b, c* sẽ lưu tương ứng trong nốt lá bên trái, nốt lá giữa và nốt lá bên phải. Hình 3.7 minh họa cho bước lặp tiếp theo của giải thuật. Để phân tách các nốt lá *a, b, c*, chữ cái thứ hai của các từ có chỉ số tương ứng chứa trong các nốt này sẽ đực xem xét. Đến đây việc xây dựng cấu trúc cây sẽ dừng vì chỉ có một từ duy nhất được chứa trong nốt lá thứ hai từ trái sang (nút được nối với nút gốc bằng cạnh *ac*). Vậy chiều dài của một từ sẽ là 2. Bởi vì các chữ cái biểu diễn các mức phân giải khác nhau của các chuỗi con nên khi có nhiều từ được xét và cấu trúc cây càng cao thì điều đó có nghĩa là giải thuật cần mức phân giải cao hơn để xác định được chuỗi con bất thường.

Thứ tự các chuỗi con trong vòng lặp ngoài và vòng lặp trong của giải thuật hình 3.4 sẽ được sắp xếp như sau: các chuỗi con nằm trong các nút lá có ít số chỉ mục đến các từ nhất sẽ được duyệt trước trong vòng lặp ngoài. Ví dụ trong hình 3.7 chuỗi con có vị trí bắt đầu là *m-n* sẽ được duyệt trước. Khi duyệt qua một chuỗi con *p* ở vòng lặp ngoài các chuỗi con nằm trong các nút có đường đi đến nút gốc khớp nhiều nhất với đường đi từ nút chứa *p* đến nút gốc sẽ được duyệt trước. Giải thuật WAT có thể tìm được *k* chuỗi con bất thường nhất trong chuỗi thời gian bằng cách chạy lại giải thuật hình 3.4 *k* lần, mỗi khi tìm được chuỗi con bất thường thứ *i* thì xóa nó ra khỏi tập các chuỗi con cần xét và chạy lại giải thuật để tìm chuỗi con bất thường thứ *i+*1.

Giải thuật WAT có khả năng xác định chiều dài của các từ một cách tự động dựa vào biểu diễn đa mức phân giải của các chuỗi con. Điều này giúp giải thuật WAT dễ sử dụng hơn so với giải thuật HOT SAX. Tuy nhiên cũng giống như giải thuật HOT SAX, giải thuật WAT cũng cần biết trước kích thước *n* của chuỗi con bất thường. Điều này hạn chế khả năng ứng dụng của WAT trong thực tế vì giá trị *n* rất khó được xác định chính xác và thay đổi tùy loại dữ liệu.

### Giải thuật tìm các chuỗi con bất thường có độ dài khác nhau của Leng và các cộng sự

M. Leng và các cộng sự trong bài báo [10] đã xây dựng một giải thuật có khả năng tìm ra các chuỗi con bất thường có chiều dài khác nhau mà không cần biết trước chiều dài của chúng. Giải thuật của các tác giả được chia làm hai bước.

Bước thứ nhất các tác giả tiến hành phân đoạn chuỗi thời gian bằng phương pháp cửa số trượt. Các điểm của chuỗi thời gian được xấp xỉ bằng một đa thức bậc 2 *f(t) = b0 + b1t + b2t2* cho đếnkhi sai số lớn hơn một giá trị *ε1­* mà người dùng chọn. Chuỗi con thứ *i* tìm được biểu diễn bằng ký hiệu *(si, ei)* với *si* là vị trí bắt đầu và *ei* là vị trí kết thúc của chuỗi. Các tác giả không chọn đa thức bậc 1 như thông thường vì cho rằng nó quá đơn giản, không thể phát hiện được các chuỗi con phức tạp. Sau khi tìm được chuỗi con thứ *i*, các tác giả tìm chuỗi con thứ *i+1* bắt đầu tại vị trí *si+1=ei+j* với *j* là số nguyên bé nhất sau cho khoảng cách từ hai chuỗi *(si, ei)* và *(si + j, ei + j)* nhỏ hơn *ε2* mà người dùng chọn hoặc *j ≥ (ei - si)*. Các bước trên được lặp lại cho đến khi toàn bộ chuỗi thời gian ban đầu đã được phân đoạn hoàn toàn thành các chuỗi con.

Bước thứ hai các tác giả sẽ tìm hai giá trị *lmin* và *lmax* là chiều dài lớn nhất và nhỏ nhất của các chuỗi con và xây dựng một ma trận khoảng cách *D = (dij)mxm* với *m* là số chuỗi con. Giá trị *dij* là khoảng cách của chuỗi con thứ *i* và thứ *j*, được tính bằng công thức:

.

Các tác giả khẳng định trong [10] là tính khoảng cách như trên hiệu quả hơn cách tính thông thường. Từ ma trận khoảng cách các tác giả tính được hệ số bất thường của các chuỗi con theo khoảng cách thứ *k* (*k-dist*). Những chuỗi nào có hệ số bất thường lớn hơn một giá trị được người dùng chọn *α0* thì chuỗi đó là chuỗi con bất thường. Hai chuỗi con bất thường *(si, ei)* và *(sj, ej)* nếu *si ≤ sj ≤ ei* thì trộn chúng lại thành một chuỗi con *(si, ej).* Ngược lại *nếu sj ≤ si ≤ ej* thì trộn chúng lại thành một chuỗi con *(sj, ei)*.

Trong bài báo [10], Leng và các cộng sự chọn k bằng là số nguyên nhỏ nhất lớn hơn hay bằng 0.05*m* (m là số chuỗi con). Hệ số *α0­* được chọn bằng 3 vì các tác giả cho rằng trong thống kê một đối tượng được coi là phần tử ngoại biên nếu khoảng cách của nó đến giá trị trung bình lớn hơn độ lệch chuẩn 3 lần [10] . Như vậy giải thuật chỉ cần 2 thông số *ε1* và *ε2* nhưng bằng thực nghiệm các tác giả cũng chứng minh được sự thay đổi của chúng không ảnh hưởng nhiều đến kết quả của giải thuật.

Giải thuật của Leng và các cộng sự có thể tìm thấy các chuỗi con bất thường có chiều dài khác nhau mà không cần biết trước độ dài của chúng. Đây là một ưu điểm so với các giải thuật tìm kiếm chuỗi con bất thường khác như HOT SAX, WAT. Tuy nhiên vì phải tính khoảng cách của các chuỗi có chiều dài khác nhau nên các tác giả dùng phương pháp xoắn thời gian động. Điều này làm cho giải thuật chậm hội tụ vì bản thân giải thuật phải thực hiện việc tính khoảng cách nhiều lần mà độ phức tạp tính toán của phương pháp xoắn thời gian động là *O(m\*n)* với *m, n* là chiều dài của hai chuỗi thời gian.

# NỘI DUNG NGHIÊN CỨU VÀ KẾ HOẠCH LÀM VIỆC

## Nội dung nghiên cứu

Mục đích của của đề tài luận văn này là xây dựng một phương pháp hiệu quả để tìm kiếm các chuỗi con bất thường trong dữ liệu chuỗi thời gian. Ở đây tôi sẽ tiếp cận theo phương pháp của M. Leng và các cộng sự trong [10] bởi vì nó có 3 ưu điểm chính. Thứ nhất, phương pháp của các tác giả có thể tìm được các chuỗi con bất thường có chiều dài khác nhau mà không cần biết trước độ dài của chúng. Thứ hai, giải thuật sử dụng cửa số trượt để phân đoạn chuỗi thời gian nên có thể thích nghi được với các chuỗi thời gian dạng luồng. Thứ ba, ý tưởng của giải thuật đơn giản, dễ hiện thực. Điểm hạn chế trong phương pháp của Leng và các cộng sự là phải sử dụng phương pháp xoắn thời gian động để tính khoảng cách giữa các chuỗi thời gian có chiều dài khác nhau. Như đã nói ở chương 2, phương pháp tính khoảng cách xoắn thời gian động có độ phức tạp tính toán cao mà việc tính khoảng cách giữa hai chuỗi thời gian được thực hiện nhiều lần trong việc phân đoạn và xây dựng ma trận khoảng cách. Điều này làm chậm tốc độ hội tụ của giải thuật.

Để giải quyết khó khăn này tôi sẽ sử dụng phương pháp tính khoảng cách Euclid, một phương pháp có tốc độ tính toán nhanh hơn nhiều phương pháp xoắn thời gian động nhằm tăng tốc độ tính toán. Tuy nhiên, công thức tính khoảng cách Euclid chỉ áp dụng cho hai chuỗi có cùng độ dài. Do đó trước tiên, tôi sẽ dùng phép *biến hình vị tự* (homothetic) để biến đổi hai chuỗi thời gian về cùng một độ dài. Sở dĩ có thể làm được điều này vì phép biến hình vị tự có thể thay đổi kích thước của một đối tượng hình học trong không gian affine mà không làm thay đổi hình dạng của nó. Một pháp biến hình vị tự tâm *O*, tỉ số *k* biến một điểm *M* thành điểm *M’* sao cho  = *k*\*. Việc sử dụng phép biến hình vị tự để biến đổi các chuỗi thời gian đã được C.D. Truong và các cộng sự sử dụng trong [3].

Phép vị tự để biến đổi một chuỗi thời gian *T* có chiều dài *n* (*T={y1, y2, …, yn})* thành *n’* được thực hiện theo các bước như sau. Thứ nhất gọi *Y\_MAX* = *MAX{y1, …, yn}, Y\_MIN = MIN{y1, …,yn }*. Thứ hai tìm *I* là tâm của chuỗi *X\_C = n/2, Y\_C = (Y\_MAX + Y\_MIN)/2*. Cuối cùng thực hiện phép biến hình vị tự với tâm *I* và tỉ số *k* = .

Để tính khoảng cách giữa hai chuỗi thời gian, trước hết ta sẽ kiểm tra chiều dài của hai chuỗi nhập, nếu chúng dài bằng nhau thì áp dụng công thức Euclid để tính khoảng cách của chúng, nếu không thì sẽ dùng phép biến hình vị tự để đưa chúng về cùng độ dài và sau đó áp dụng công thức Euclid. Vì độ phức tạp tính toán của phép biến hình vị tự là *O(n)* và của công thức tính Euclicd *là O(n)* nên phương pháp mới có độ phức tạp là *O(n)*, nhanh hơn nhiều so với phương pháp xoắn thời gian động (có độ phức tạp tính toán *O(m\*n)*).

Để tìm chuỗi con bất thường của một chuỗi thời gian, trước tiên tôi sẽ phân đoạn chuỗi thời gian thành những chuỗi con bằng phương pháp cửa sổ trượt và đa thức xấp xỉ là đa thức bật hai *f(t) = b0 + b1t + b2t2*. Tiếp theo xây dựng một ma trận khoảng cách cho các chuỗi con và từ đó rút ra hệ số bất thường của các chuỗi này. Các chuỗi có hệ số bất thường lớn hơn một ngưỡng cho trước sẽ được đưa vào danh sách các chuỗi con bất thường. Hai chuỗi con trong danh sách chuỗi con bất thường có điểm đầu và điểm cuối *phủ lên nhau* (overlapping) sẽ được trộn lại thành một chuỗi con bất thường mới.

Tôi cũng sẽ tìm hiểu và hiện thực giải thuật HOT SAX để làm cơ sở đánh giá tính chính xác cho phương pháp mới. Các kết quả của phương pháp mới sẽ được so sánh với kết quả của HOT SAX để đánh giá hiệu quả.

## Kế hoạch làm việc

Luận văn sẽ được thực hiện trong 24 tuần. Bảng 5.1 bên dưới là bảng kế hoạch của đề tài.

|  |  |  |  |
| --- | --- | --- | --- |
| Thứ tự | Nội dung | Thời gian bắt đầu | Thời gian kết thúc |
| 1 | Tìm hiểu bài toán tìm chuỗi con bất thường, các giải thuật phân đoạn, các công thức tính khoảng cách và các tiêu chuẩn đánh giá độ bất thường. Các công trình liên quan đến bài toán. | Tuần 1 | Tuần 3 |
| 2 | Tìm hiểu giải thuật HOT SAX và giải thuật được M. Leng cùng các cộng sự đề xuất. | Tuần 4 | Tuần 7 |
| 3 | Hiện thực phương pháp đề xuất và giải thuật HOT SAX | Tuần 8 | Tuần 18 |
| 4 | Tạo dữ liệu kiểm tra và chạy thực nghiệm so sánh phương pháp đề xuất với giải thuật HOT SAX | Tuần 16 | Tuần 20 |
| 5 | Viết luận văn | Tuần 4 | Tuần 24 |

Bảng 4.1. Bảng kế hoạch làm việc.

# TÀI LIỆU THAM KHẢO

[1] A.W. Fu, O.T. Leung, E. Keogh, J. Lin. *Finding Time Series Discords Based on Haar Transform*. Advanced Data Mining and Applications – ADMA, pp. 31-41, 2006.

[2] C.A. Ratanamahatana, E. Keogh. *Making Time-series Classification More Accurate Using Learned Constraints*. In Proceedings of SIAM International Conference on Data Mining, 2004.

[3] C.D. Truong, H.N. Tin, D.T Anh. *Combining motif information and neural network for time series prediction****.*** Int. J. Business Intelligence and Data Mining, vol. 7, no. 4, pp. 318-339, 2012.

[4] D. Lemire***.*** *A Better Alternative to Piecewise Linear Time Series Segmentation*. Proceedings of the 7th SIAM International Conference on Data Mining, pp. 545-550, 2007.

[5] E. Keogh, J. Lin, A. Fu. *HOT SAX: Finding the Most UnusualTime Series Subsequence: Algorithms and Applications****.*** Proceedings of the 5th IEEE International Conference on Data Mining, pp. 226-233, 2005.

[6] E. Keogh, K.Chakrabarti, M. Pazzani, S. Mehrotra. *Dimensionality Reducion for Fast Similarity Search in Large Time Series Database*. Knowledge and Information Systems 3, pp. 263-286, 2001.

[7] E. Keogh, S. Chu, D. Hart, M. Pazzani.*An Online Algorithm for Segmenting Time Series****.*** Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 289-296, 2001.

[8] J. Lin, E Keogh, S. Lonardi, B. Chiu. *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*. Proceedings of the 8th ACM SIGMOD, pp. 2-11, 2003.

[9] K. Chan, A.W. Fu. *Efficient Time Series Matching by Wavelets*. Proceedings of the 15th International Conference on Data Engineering, pp. 126-133, 1999.

[10] M. Leng, X. Chen, L. Li. *Variable Length Methods for Detecting Anomaly Patterns in Time Series*. International Symposium on Computational Intelligence and Design, pp. 52-56, 2008.

[11] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi. *Times Series anomaly discovery with grammar-based compression.* 18th International Conference on Extending Database Technology, pp. 481-492, 1015.

[12] S. Lee, D. Kwon, S. Lee. *Efficient Pattern Matching of Time Series Data.* InProceedings of the 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE, pp. 586-595, 2002.

[13] W. Luo, M. Gallagher, J. Wiles. *Parameter-Free Search of Time-Series Discord*. Journal of Computer Science and Technology, vol. 28, no. 2, pp. 300-310, 2013.

[14] Y.S. Jeong, M.K. Jeong, O.A Omitaomu. *Weighted dynamic time warping for time series classification*. In Pattern Recognition 44, pp. 2231-2240, 2011.

[15] Y. Bu, T. Leung, A.W. Fu, E. Keogh, J. Pei, S. Meshkin. *WAT: Finding Top-K Discords in Time Series Database*. Proceedings of the 7th SIAM International Conference on Data Mining, pp. 449-454, 2007.