ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGÔ DUY KHÁNH VY

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

Chuyên ngành: Khoa Học Máy Tính

Mã số: 60.48.01

LUẬN VĂN THẠC SĨ

PGS.TS DƯƠNG TUẤN ANH

TP. HỒ CHÍ MINH, 01/2016

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

Chương 1
GIỚI THIỆU

1.1. Giới thiệu bài toán

1.1.1. Bài toán tìm kiếm bất thường tổng quát

Tìm kiếm bất thường, một cách tổng quát, là bài toán tìm ra các đối tượng dữ liệu có hành vi khác với hành vi chung của các đối tượng trong một tập dữ liệu. Bài toán này có ý nghĩa quan trọng vì các đối tượng dữ liệu bất thường được qua tâm phân tích nhiều hơn các đối tượng tuân theo các hành vi chung [31]. Trong công nghiệp, chuỗi các tín hiệu bất thường được gởi đi từ một thiết bị cảm ứng theo dõi hoạt động của dây chuyền sản xuất có thể là dấu hiệu cho thấy có một hay một số bộ phận của dây chuyền đang bị hỏng. Trong y tế, một đoạn dữ liệu điện tâm đồ khác biệt của một bệnh nhân là chỉ dấu cho các vấn đề về sức khỏe của người này.

Để tìm kiếm bất thường trong một tập dữ liệu cách đơn giản là định nghĩa một *vùng* (region) biểu diễn hành vi bình thường của tập dữ liệu và các đối tượng không thuộc vùng này sẽ là các bất thường. Tuy nhiên theo V. Chandola và các cộng sự trong [31], cách làm này gặp phải các khó khăn sau:

- Rất khó xác định các hành vi bình thường của tập dữ liệu. Hơn nữa sự khác biệt giữa các hành vi bình thường so với hành vi bất thường có thể không rõ ràng.
- Trong nhiều lĩnh vực, các hành vi bình thường của tập dữ liệu có thể thay đổi theo thời gian
- Trong các lĩnh vực khác nhau, tiêu chí để đánh giá một đối tượng dữ liệu là bất thường rất khác nhau. Ví dụ trong y học, một độ lệch nhỏ so với

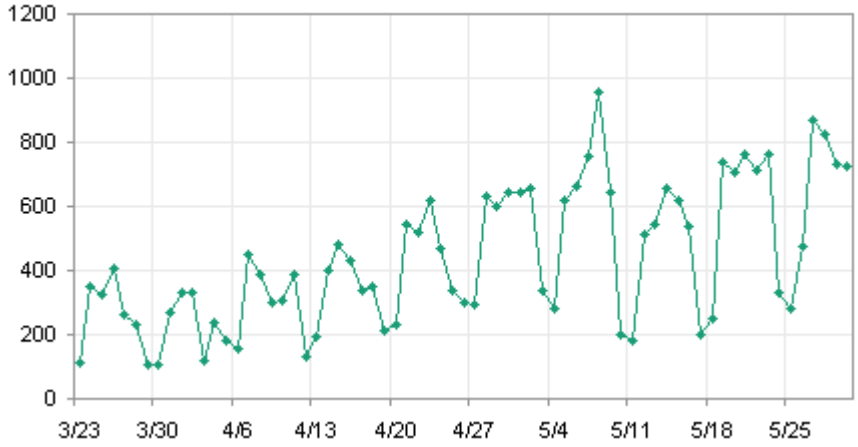
các hành vi bình thường cũng có thể xem là bất thường trong khi trong lĩnh vực chứng khoáng độ lệch như vậy vẫn được xem là bình thường.

Với các khó khăn trên, bài toán tìm kiếm bất thường là một bài toán không dễ giải quyết tổng quát và thực tế hầu hết mỗi kỹ thuật đã được xây dựng chỉ giải quyết được một số trường hợp đặc biệt của bài toán.

1.1.2. Bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian

Bài toán tìm kiếm chuỗi con bất thường trong dữ liệu chuỗi thời gian là một trường hợp riêng của bài toán tìm kiếm bất thường. Nhiệm vụ của bài toán là phát hiện được các đoạn con có hình dạng khác biệt so với các đoạn khác trong một chuỗi thời gian lớn. Việc xây dựng được một kỹ thuật hiệu quả để giải quyết bài toán này ngày càng được quan tâm do sự xuất hiện ngày càng nhiều của dữ liệu chuỗi thời gian trong nhiều lĩnh vực khác nhau 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. 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 đang xét. 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 có thể xem mỗi đoạn của một chuỗi thời gian là một đối tượng dữ liệu đa chiều. Số chiều của đối tượng dữ liệu có thể thay đổi từ vài chục như doanh số bán hàng theo ngày của một cửa hàng trong một quí 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 [5].



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. Việc phát hiện ra các chuỗi con bất thường trong dữ liệu chuỗi thời gian có 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 bệnh nhân 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 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 không đơn giản. Bản thân bài toán này ngoài những khó khăn vốn có của bài toán tìm kiếm bất thường còn chứa đựng những khó khăn của chính nó. Khó khăn thứ nhất là ta không biết trước được chiều dài của các chuỗi con bất thường do đó rất khó để tách chuỗi thời gian thành các đoạn con để so sánh. Thứ hai các chuỗi thời gian khác thuộc các *lĩnh vực* (domain) khác nhau thường có hành vi khác nhau, điều này gây khó cho việc tìm ra một kỹ thuật tổng quát có thể áp dụng cho nhiều lĩnh vực. Thứ ba hiện vẫn chưa có một tiêu chuẩn để đánh giá tính chính xác của một kỹ thuật, thông thường việc đánh giá phải dựa vào sự kiểm tra bằng mắt và hiểu biết của người quan sát về tập dữ liệu.

Nhiều nhà nghiên cứu đã quan tâm đến bài toán này và đưa ra nhiều giải thuật. Một số giải thuật dựa trên tính chu kỳ của chuỗi dữ liệu [28], một số khác dựa trên sự hiểu biết về bản chất dữ liệu để biết trước chiều dài của chuỗi con bất

thường như giải thuật HOT SAX [6] hay WAT [30][2]. M. Leng và các cộng sự đề xuất một phương pháp dựa trên việc phân đoạn chuỗi thời gian bằng cách dùng các đa thức bậc 2 để xấp xỉ chuỗi thời gian và so sánh các đoạn bằng *phương pháp chiều dài biến đổi* (variable length method) để tìm các chuỗi con bất thường có chiều dài khác nhau.

1.2. Mục tiêu của đề tài

Mục tiêu của đề tài là xây dựng một kỹ thuật tìm kiếm các chuỗi con bất thường có chiều dài khác nhau trong dữ liệu chuỗi thời gian mà không cần biết trước chiều dài của chuỗi con bất thường. Chúng tôi tiếp cận theo mô hình M. Leng và các cộng sự bởi vì mô hình này đáp ứng được yêu cầu trên 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 M.Leng và các cộng sự phải sử dụng độ đo *xoắn thời gian động* (Dynamid time warping) 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.

Chúng tôi đề xuất một cách tính khoảng cách mới dựa trên phép biến hình vị tự và công thức Euclid. Cách tính mới này có độ phức tạp tính toán tuyến tính do đó giảm được thời gian tìm kiếm mà vẫn giữ được các ưu điểm của mô hình. Chúng tôi cũng đề xuất một giải thuật phân đoạn mới dựa trên các *điểm cực trị quan trọng* (Significant extreme points). Phương pháp phân đoạn mới mày theo đánh giá của chúng tôi dễ ước lượng các tham số hơn phương pháp phân đoạn do M.Leng và các cộng sự đề xuất. Độ chính xác của giải thuật mới sẽ được so sánh bằng thực nghiệm với giải thuật HOT SAX.

1.3. Cấu trúc luận văn

Cấu trúc phần còn lại của luận văn sẽ gồm những chương sau

Chương 2: Các cơ sở lý thuyết. Trong chương này chúng tôi sẽ phân loại các bất thường, các tiêu chí đánh giá bất thường, trình bày các định nghĩa, các

phương pháp thu giảm số chiều và rời rạc hóa dữ liệu được sử dụng trong bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian.

Chương 3: Giới thiệu các công trình liên quan. Chương này chúng tôi sẽ trình bày các công trình liên quan đến phân đoạn chuỗi thời gian và các công trình về tìm kiếm chuỗi con bất thường. Sở dĩ chúng tôi đề cập đến các công trình liên quan đến phân đoạn là vì giải thuật phân đoạn có ảnh hưởng rất lớn đến tính hiệu quả của mô hình tìm kiếm chuỗi con bất thường trên dữ liệu chuỗi thời gian mà chúng tôi sử dụng.

Chương 4: Phương pháp giải quyết vấn đề. Phần này trình bày chi tiết các giải quyết vần đề của chúng tôi.

Chương 5: Thực nghiệm. Phần này chúng tôi trình bày kết quả thực nghiệm trên các bộ dữ liệu khác nhau mà chúng tôi đã thực hiện.

Chương 6: Kết luận. Phần này là một số kết luận về đóng góp, hạn chế và các hướng phát triển của đề tài

Chương 2
CƠ SỞ LÝ THUYẾT

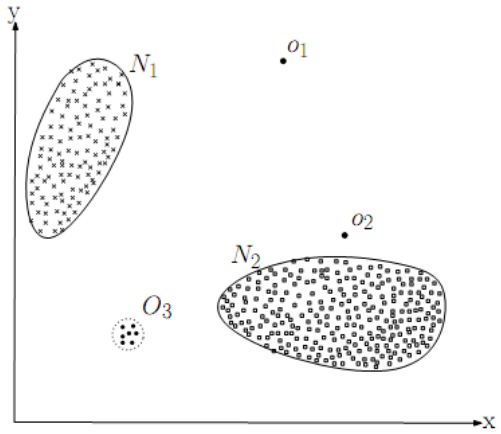
Chương này sẽ trình bày việc phân loại các bất thường, các tiêu chí đánh giá bất thường, các định nghĩa, các giải thuật phân đoạn, các phương pháp đo khoảng cách, các giải thuật *thu giảm số chiều* (dimensionality reduction) và *rời rạc hóa* (discretization) dữ liệu thường được sử dụng trong các công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

2.1. Các loại bất thường

Theo V.Chandola và các cộng sự các bất thường có thể phân thành ba loại chính [31]: *bất thường điểm* (point anomalies), *bất thường theo ngữ cảnh* (contextual anomalies), *bất thường tập thể* (collective anomalies).

2.1.1. Bất thường điểm.

Bất thường điểm là các đối tượng dữ liệu có giá trị khác biệt so với các đối tượng khác trong tập dữ liệu. Hình 2.1 là một ví dụ về loại bất thường này, các điểm o_1 và o_2 là 2 bất thường điểm. Bất thường điểm cũng thường hay được gặp trong thực tế, ví dụ khi khảo sát số tiền thanh toán trong các giao dịch trực tuyến của một cá nhân, nếu có một số giao dịch có số tiền thanh toán cao hơn mức bình thường thì các giao dịch này có thể là các bất thường điểm, hoặc khi quan sát số lượng truy cập hàng giờ của một website, nếu tại một thời điểm mà lượng truy cập cao hơn hay thấp hơn nhiều so với các thời điểm khác thì số lượng truy cập tại thời điểm này có thể xem là một bất thường.

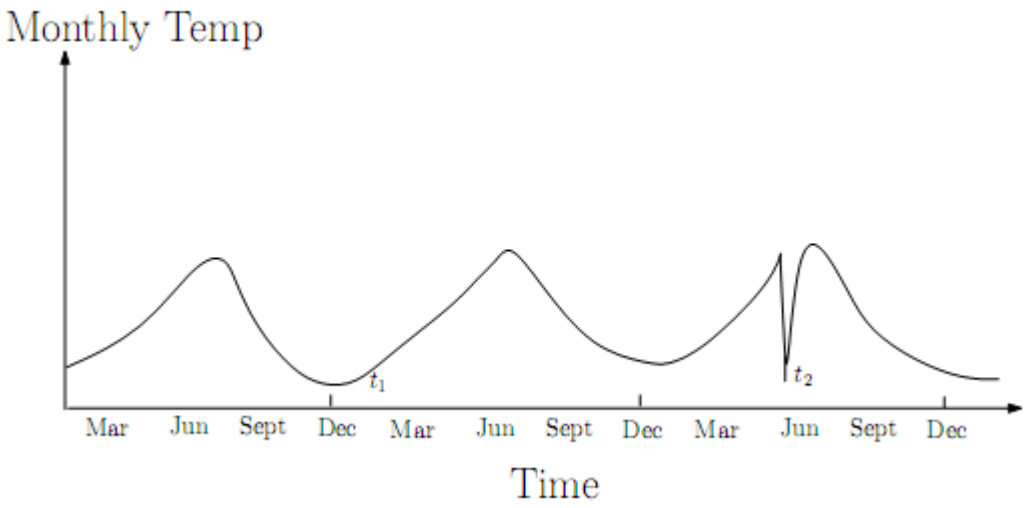


Hình 2.1. Ví dụ về bất thường điểm trong tập dữ liệu 2 chiều.

2.1.2. Bất thường theo ngữ cảnh

Trong thực tế có các đối tượng dữ liệu mà giá trị của chúng không khác biệt so với các giá trị khác trong tập dữ liệu nhưng nếu xét trong một ngữ cảnh tương ứng thì các đối tượng đó chính là các bất thường. Ví dụ khi theo dõi nhiệt độ của một thành phố theo tháng, vào các tháng mùa đông nếu nhiệt độ khoảng 10-15 độ C là bình thường nhưng cũng với nhiệt độ này nếu quan sát được vào các tháng mùa hè thì đó là bất thường. Trong bài toán tìm bất thường theo ngữ cảnh, một đối tượng dữ liệu chứa đựng hai loại thuộc tính: *các thuộc tính ngữ cảnh* (contextual attributes) và *các thuộc tính hành vi* (behavioral attributes). Hình 2.2, đối tượng t_2 có thuộc tính ngữ cảnh là tháng 6 và thuộc tính hành vi là giá trị nhiệt độ. Giá trị t_2 giống với giá trị t_1 nhưng do nó nằm trong ngữ cảnh khác nên được xem là một bất thường.

Bài toán xác định các đối tượng bất thường theo ngữ cảnh là một bài toán khó. Vì trong thực tế rất khó xác định được tất cả các ngữ cảnh của một đối tượng dữ liệu. Các ngữ cảnh này không cố định có thể thay đổi theo thời gian và theo từng loại dữ liệu.

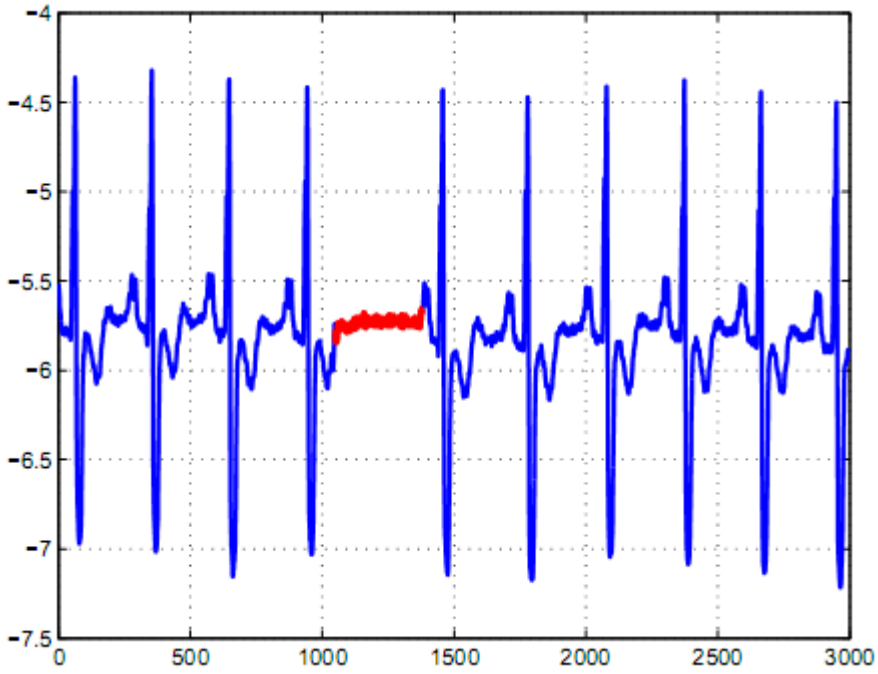


Hình 2.2. t_2 là bất thường theo ngữ cảnh trong chuỗi dữ liệu nhiệt độ theo tháng.

2.1.3. Bất thường tập thể

Đối với loại bất thường theo tập thể, mỗi một đối tượng có thể không phải là một bất thường nhưng khi chúng kết hợp với nhau sẽ tạo thành một chuỗi các đối tượng có hành vi khác vời toàn bộ tập dữ liệu. Chuỗi con bất thường trong dữ liệu chuỗi thời gian có thể xếp vào kiểu bất thường này. Trong đó mỗi điểm dữ liệu không phải khác thường nhưng khi chúng nối với nhau có thể tạo thành một chuỗi con có hình dạng khác biệt so với toàn bộ chuỗi thời gian. Hình 2.3 là một ví dụ về chuỗi con bất thường. Các điểm dữ liệu trong phần màu đỏ có giá trị bình thường nhưng chúng nối với nhau sẽ tạo thành một chuỗi con có hình dạng khác biệt.

Phương pháp thông thường để xác định các bất thường tập thể là biểu diễn các tập hợp điểm dữ liệu thành các đối tượng hình học và so sánh sự khác biệt về hình dạng của các đối tượng hình học này bằng một hàm khoảng cách. Các đối tượng hình học có hình dạng khác biệt nhiều so với các đối tượng hình học còn lại có thể là các bất thường.



Hình 2.3. Chuỗi con bất thường trong dữ liệu chuỗi thời gian.

2.2. Tiêu chí đánh giá chuỗi con bất thường trong dữ liệu chuỗi thời gian

Trong tập hợp các chuỗi con của một chuỗi thời gian, để đánh giá một chuỗi con có phải là bất thường hay không có thể dùng hai cách. Cách thứ nhất là tiến hành gom cụm các chuỗi con. Phương pháp này dựa trên giả định là các chuỗi con bình thường sẽ thuộc các cụm lớn, có nhiều phần tử trong khi các chuỗi con bất thường sẽ thuộc các cụm nhỏ, ít phần tử và các chuỗi con bình thường sẽ nằm gần *tâm* (centroid) của cụm gần nhất trong khi các chuỗi con bất thường sẽ nằm xa tâm của cụm gần nhất.

Cách thứ hai dựa trên *khoảng cách lân cận thứ k* (k nearest neighbor). Phương pháp này giả định rằng các chuỗi con bất thường sẽ có khoảng cách lân cận thứ k lớn hơn so với các chuỗi con bình thường. Cách dùng khoảng cách thứ k để xác định một chuỗi con là bất thường được xử dụng bởi nhiều tác giả [6][30][22]. Trong luận văn này, chúng tôi cũng sử dụng tiêu chí này để xác định một chuỗi con là bất thường.

Phần tiếp theo của chương này chúng tôi sẽ trình bày các định nghĩa dùng trong luận văn, các cách tính khoảng cách, các phương pháp thu giảm số chiều và rời rạc hóa dữ liệu chuỗi thời gian.

2.3. 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 $T = x_1, x_2, \ldots, x_m$ với x_i 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* ($m \leq n$) là một đoạn các giá trị liên tục nằm trong *T*. Ta ký hiệu $C = x_p, x_{p+1}, \ldots, x_{p+n-1}$, với $1 \leq p \leq m-n+1$. Đôi khi *C* được ký hiệu bằng (s_p, e_{p+n-1}), với $s_p = x_p$ và $e_{p+n-1} = x_{p+n-1}$.

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 giữa *P* và *Q* với *Q* thuộc *D* và thỏa mãn hai tính chất sau.

i) Tồn tại ít nhất *k* phần tử *Q'* thuộc *D* sao cho *Dist(D, Q') ≤ Dist(D,Q).*

ii) 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 [22]) nếu $Dist(P, Q) \geq e$ hoặc $|p - q| \geq n$, 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 [6] hay Y. Bu và các cộng sự trong [30] khẳng định hai chuỗi P, Q là khớp không tầm thường nếu $|p - q| \geq n$.

Sau đây là một số định nghĩa liên quan đến khái niệm điểm cực trị quan trọng do Pratt và Fink đề xuất năm 2002 [18].

Định nghĩa 7: Với một *tỉ số nén* (compression rate) $R > 1$, điểm t_m thuộc chuỗi thời gian T là điểm *cực tiểu quan trọng* (signification minimum) nếu tồn tại các chỉ số i và j, $i < m < j$ thỏa mãn hai điều kiện

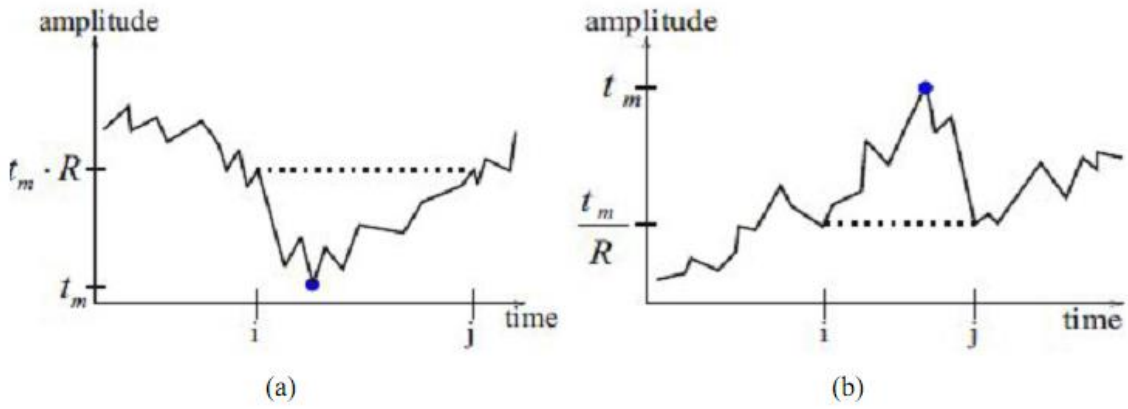
- t_m là giá trị bé nhất trong đoạn $t_i, \ldots, t_j$
- $t_i / t_m \geq R$ và $t_j / t_m \geq R$

Định nghĩa 8: Với một tỉ số nén $R > 1$, điểm t_m thuộc chuỗi thời gian T là điểm *cực đại quan trọng* (signification maximun) nếu tồn tại các chỉ số i và j, $i < m < j$ thỏa mãn hai điều kiện

- t_m là giá trị lớn nhất trong đoạn $t_i, \ldots, t_j$
- $t_m / t_i \geq R$ và $t_m / t_j \geq R$

Các điểm cực đại và cực tiểu quan trọng được gọi chung là điểm cực trị quan trọng. Hình 2.4 là một minh họa về các điểm cực trị quan trọng. Hình (a) biểu diễn một cực tiểu quan trọng và hình (b) biểu diễn một cực đại quan trọng.

Các điểm cực trị quan trọng được sử dụng để nén các chuỗi thời gian nhằm phục vụ cho việc đánh chỉ mục [18] hay dùng để phân đoạn chuỗi thời gian [24][4].



Hình 2.4. Điểm cực trị quan trọng, (a) là điểm cực tiểu, (b) là điểm cực đại

2.4. 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 độ đo *xoắn thời gian động* (Dynamic time warping).

2.4.1. Công thức tính khoảng cách Euclid

a) Công thức tính khoảng cách thông thường

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

$$Dist(Q,C) = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \qquad (2.4.1)$$

với q_i thuộc Q và c_i thuộc C. Công thức tính khoảng cách này chỉ áp dụng được cho hai chuỗi thời gian có cùng chiều dài.

b) Công thức tính khoảng cách Euclid hiệu chỉnh

Trong một số trường hợp, các chuỗi thời gian có hình dạng giống nhau nhưng bị lệch nhau một khoảng theo trục tung. Ví dụ trong Hình 2.5 hai chuỗi thời gian rất giống nhau về hình dạng nhưng sẽ có khoảng cách lớn nếu tính theo công

thức (2.4.1). 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 [4] và Lee cùng các cộng sự trong [26] 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:

$$Dist(Q,C) = \sqrt{\sum_{i=1}^{n}(q_i - c_i - b)^2} \qquad (2.4.2)$$

Ở đây b là một số thực được tính bằng công thức

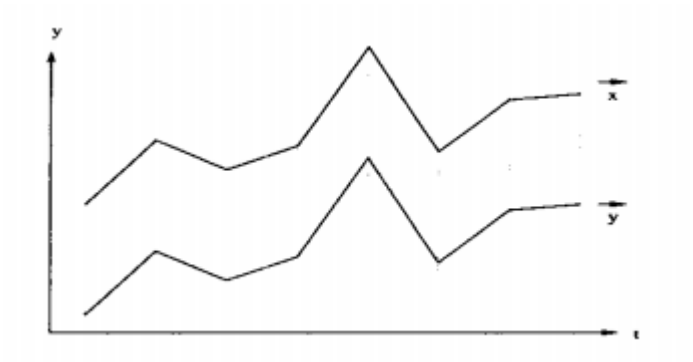
$$b = \frac{1}{n}\sum_{i=1}^{n}(q_i - c_i) \qquad (2.4.3)$$

K. Chan và các cộng sự trong [19] 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:

$$Dist(Q,C) = \sqrt{\left(\sum_{i=1}^{n}\left((q_i - q_A) - (c_i - c_A)\right)^2\right)} \qquad (2.4.4)$$

Với $q_A = \frac{1}{n}\sum_{i=1}^{n} q_i$ và $c_A = \frac{1}{n}\sum_{i=1}^{n} c_i$

Phương pháp tính khoảng cách bằng công thức Euclid đơn giản, dễ hiện thực và có 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 2.5. Hai chuỗi thời gian hình dạng giống nhau nhưng bị lệch theo trục tung

2.4.2. 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 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 2.6 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 độ đo xoắn thời gian động đối sánh các điểm sao cho tối ưu nhất. Dùng độ đo 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.

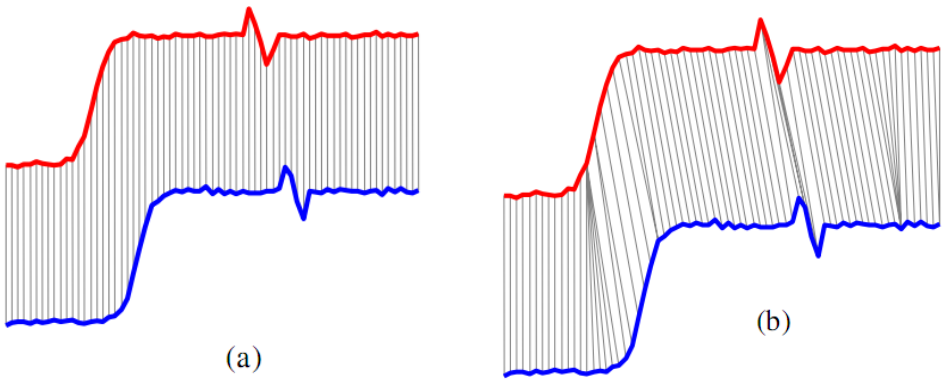
Độ đo xoắn thời gian động được hiện thực như sau. Gọi chuỗi thời gian thứ nhất là *A*, có chiều dài *m*, ký hiệu $A = a_1, a_2, ..., a_m$. Gọi *B* là chuỗi thời gian thứ 2 có chiều dài *n*, ký hiệu $B = b_1, b_2, ..., b_n$.Ta xây dựng một ma trận đường đi *M* với *m* hàng và *n* cột. Mỗi phần tử *(i,j)* của ma trận *M* tương ứng với khoảng cách của phần tử a_i và b_j . Trong các chuỗi thời gian thông thường, a_i, b_j là các số thực và khoảng cách của chúng được tính bằng $d(a_i,b_j) = |a_i - b_j|$. Đường xoắn thời gian *D* là một chuỗi $u_1, u_2, ..., u_k$ với u_i tương ứng với các một phần tử $(a_i\, b_j)$ 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 $u_1 = (a_1,b_1)$, $u_k = (a_m,b_n)$ [29].

ii) Ràng buộc tính liên tục (continuity constraint): nếu $u_k = (a_i, b_j)$, $u_{k+1} = (a_{i+1}, b_{j+1})$ thì $a_i - a_{i+1} \leq 1$ và $b_i - b_{i+1} \leq 1$ [29].

iii) Ràng buộc tính đơn điệu (Monotonicity constraint): nếu $u_k = (a_i, b_j)$, $u_{k+1} = (a_{i+1}, b_{j+1})$ thì $a_i \leq a_{i+1}$ và $b_i \leq b_{i+1}$ [29].

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 [3].



Hình 2.6. (a) Đo khoảng cách bằng công thức Euclid. (b) Đo khoảng cách bằng độ đo 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.

$$DTW_p(i,j) = \sqrt[p]{\gamma(i,j)} \quad (2.4.5)$$

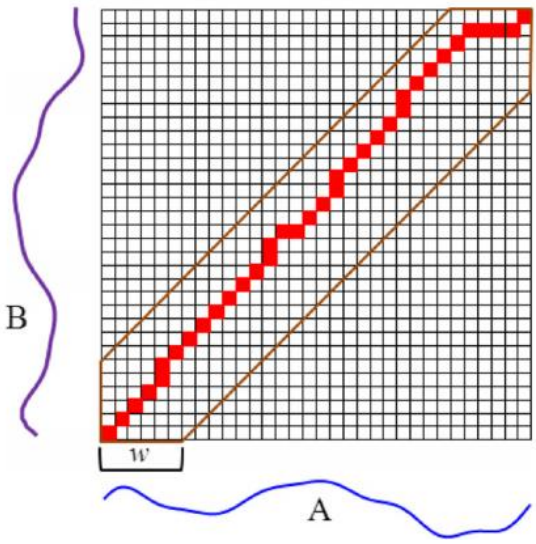
Ở đây $\gamma(i,j)$ là khoảng cách tích lũy được tính đệ quy như sau:

$$\gamma(i,j) = \left|a_i - b_j\right|^p + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\} \quad (2.4.6)$$

Với $\gamma(1,1) = \left|a_1 - b_1\right|^p$

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 2.7 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 2.7. Ma trận xoắn thời gian và đường xoắn thời gian.

Phương pháp tính khoảng cách bằng độ đo 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 phương pháp này khó có thể áp dụng đối với các chuỗi thời gian có kích thước lớn hay những chuỗi thời gian dạng luồng, khi mà dữ liệu mới liên tục cập nhập đòi hỏi các bước tính toán phải thực hiện nhanh.

Hình 2.8 dưới đây là mã giả cho phương pháp xoắn thời gian động. Giải thuật Hình 2.8 thực hiện hai vòng lặp lồng nhau. Tại mỗi bước lặp, giá trị *DTW[i,j]* được tính dựa vào các giá trị đã được tính trước đó *DTW[i-1,j]*, *DTW[i, j-1] và DTW[i-1, j-1]*.

Input: $Q: array\ [1 \dots n], C: array\ [1 \dots m], DTW : array\ [1 \dots n, 1 \dots m]$

Output: $DTW[n, m]$

for i = 1 : n

for j = 1 : m

$DTW[i, j]$

$= (Q[i] - C[j])^2$

$+ min(DTW[i-1, j], DTW[i, j-1], DTW[i-1, j-1])$

return $DTW[n, m]$

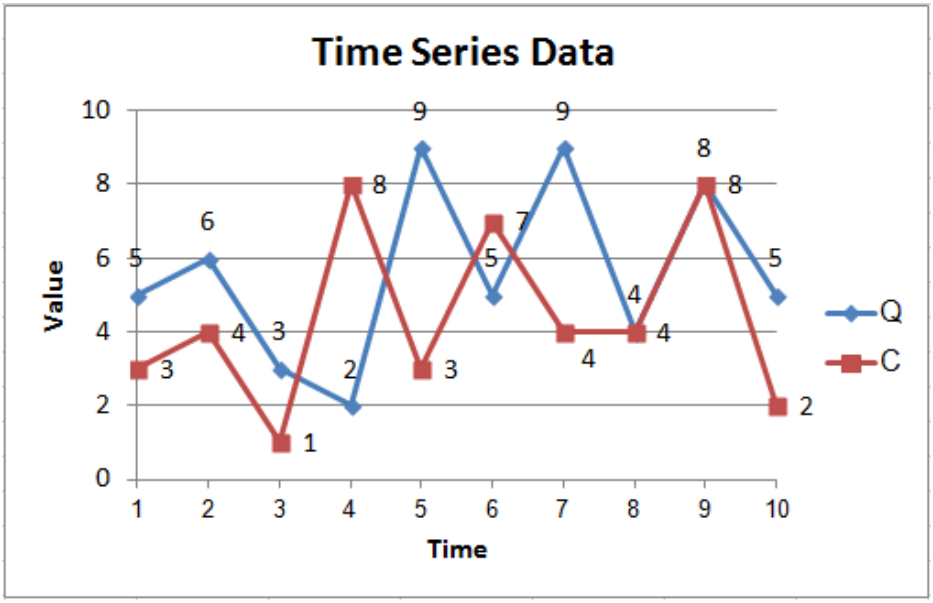
Hình 2.8. Mã giả cho giải thuật xoắn thời gian động.

Ví dụ sau đây sẽ minh họa cho giải thuật tính khoảng cách DTW. Giả sử chúng ta có 2 chuỗi thời gian:

$$Q = (5, 6, 3, 2, 9, 5, 9, 4, 8, 5)$$

$$C = (3, 4, 1, 8, 3, 7, 4, 4, 8, 2)$$

Hai chuỗi này được biểu diễn đồ thị bằng Hình 2.9.



Hình 2.9. Hai chuỗi thời gian minh họa cho phương pháp DTW.

Để tính khoảng cách DTW chúng ta xây dựng ma trận tính khoảng cách tích lũy của hai chuỗi trên như Hình 2.10. Mỗi ô trong ma trận sẽ chứa khoảng cách tích lũy tương ứng của cặp điểm đó. Giá trị trong các ô được tính như sau:

$$\gamma(1,1) = (Q_1 - C_1)^2$$

$$\gamma(1,j) = \gamma(1,j-1) + \left(Q_1 - C_j\right)^2$$

$$\gamma(i,1) = \gamma(i-1,1) + (Q_i - C_1)^2$$

$$\gamma(i,j) = min\{\gamma(i-1,j-1), \gamma(i,j-1), \gamma(i-1,j)\} + \left(Q_i - C_j\right)^2$$

		C									
		3	4	1	8	3	7	4	4	8	2
	5	4	5	21	30	34	38	39	40	49	58
	6	13	8	30	25	34	35	39	43	44	60
	3	13	9	12	37	25	41	36	37	62	45
	2	14	13	10	46	26	50	40	40	73	45
Q	9	50	38	74	11	47	30	55	65	41	90
	5	54	39	54	20	15	19	20	21	30	39
	9	90	64	103	21	51	19	44	45	22	71
	4	91	64	73	37	22	28	19	19	35	26
	8	116	80	113	37	47	23	35	35	19	55
	5	120	81	96	46	41	27	24	25	28	28

Hình 2.10. Ma trận xoắn DTW cho hai chuỗi Q và C.

Sau khi đã tính tất cả giá trị tích lũy cho các ô, chúng ta được một đường xoắn tối ưu (đường tô đậm trên Hình 2.10) bao gồm các ô tham gia tích lũy cho ô (10,10). Vậy khoảng cách DTW của hai chuỗi trên là $\sqrt{28} \approx 5{,}2915$.

Do độ phứa tạp tính toán cao của giải thuật xoắn thời gian động, một số tác giả đề xuất thêm các ràng buộc nhằm tăng tốc cho giải thuật. Phần sau đây sẽ trình bày hai ràng buộc quan trọng cho giải thuật xoắn thời gian động.

- **Ràng buộc dãi Sakoe-Chiba:**

Ràng buộc này được đề xuất bởi Sakoe và Chiba năm 1978 [13] định nghĩa đường xoắn hợp lệ: $W = w_1, w_2, \dots, w_k, \dots, w_K$ với $max(m,n) \leq K < m + n - 1$ và $w_k = (i,j)_k$ là tập các ô của ma trận xoắn của hai chuỗi thời gian với điều kiện $|i - j| \leq \omega$ với ω là một số nguyên dương cho trước gọi là cửa sổ xoắn.

- **Ràng buộc hình bình hành Itakura:**

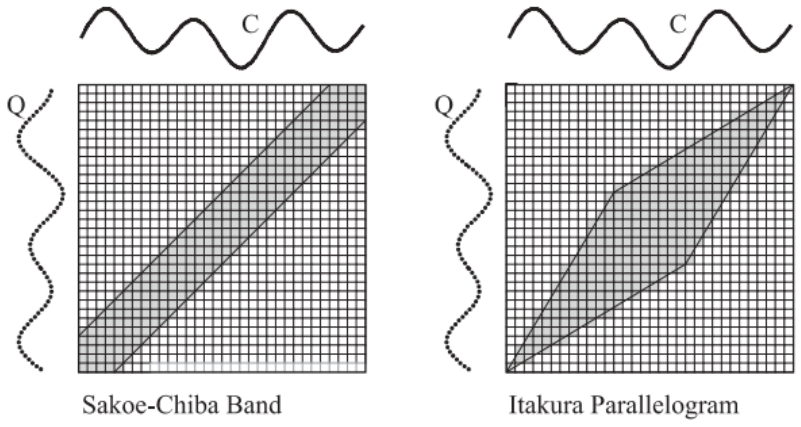
Ràng buộc này được đề xuất bởi Itakura năm 1975 [11] cũng định nghĩa đường xoắn hợp lệ được ràng buộc trong một tập con của ma trận xoắn của hai chuỗi thời gian theo hình dạng hình bình hành. Cho điểm i^{th} và điểm j^{th} tương ứng của hai chuỗi *Q*, *C* thì ràng buộc Itakura phát biểu rằng điểm j^{th} phải được định nghĩa bởi một hàm biến thiên thời gian theo i^{th}:

$$j^{th} = \omega(i^{th})$$

với các điều kiện ràng buộc biên: $\omega(1) = 1$, $\omega(n) = m$ và điều kiện liên tục:

$$\omega(i+1) - \omega(i) = 0,1,2 \quad (\omega(i) \neq \omega(i-1))$$
$$= 1,2 \quad (\omega(i) = \omega(i-1))$$

Hình 2.11 là minh họa cho cửa sổ xoắn của ràng buộc dãi Sakoe-Chiba và ràng buộc hình bình hành Itakura



Hình 2.11. Cửa sổ xoắn của ràng buộc dãi Sakoe-Chiba và ràng bược hình bình hành Itakura.

2.5. 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 công 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) và phương pháp *biểu diễn bit* (Bit representation).

2.5.1. Phương pháp xấp xỉ PAA.

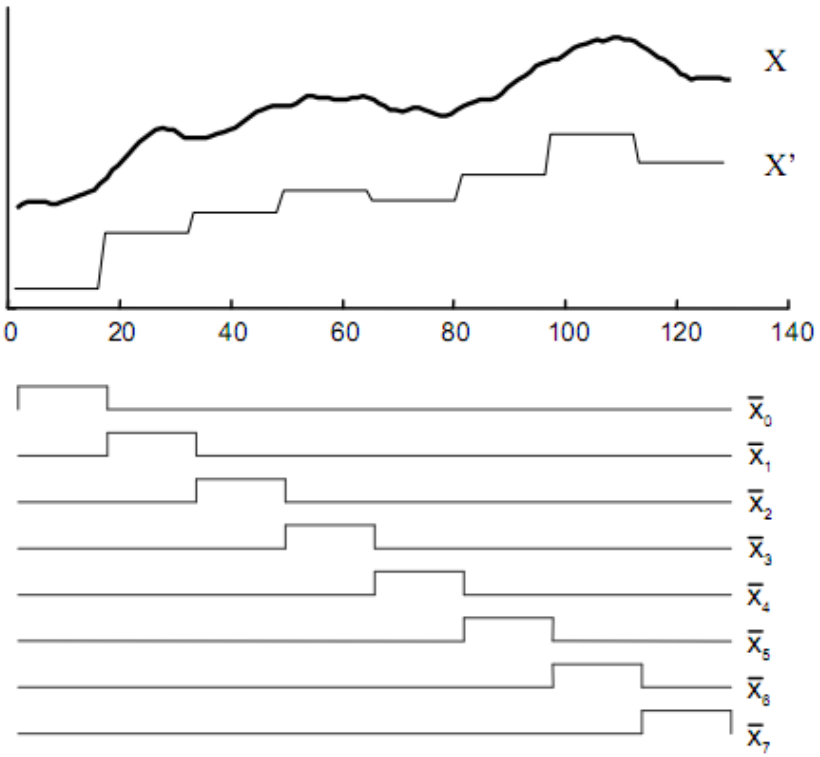
Phương pháp xấp xỉ PAA [7] được sử dụng để thu giảm số chiều của một chuỗi thời gian. Gọi một chuỗi thời gian $X = x_1, x_2, ..., x_n$, phương pháp xấp xỉ PAA biến đổi chuỗi X thành chuỗi $X' = x'_1, x'_2, ..., x'_N$ với $1 \leq N \leq n$. Thông thường, N là một thương số của n. Các điểm của chuỗi X' được tính bằng công thức sau:

$$x'_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{N}{n}i} x_j \qquad (2.5.1)$$

Từ công thức (2.5.1) 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 tạo thành một vector biểu diễn 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 [7][29][16][25]. Hình 2.12 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 2.12. 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 [16].

2.5.2. 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 [19]. 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 [19], (3) bảo toàn khoảng cách Euclid [19].

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 2.13 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.

Resolution	Averages	Coefficients
4	(9 7 3 5)	
2	(8 4)	(1 -1)
1	(6)	(2)

Hình 2.13 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

$$H(f(x)) = (c\, d_0^0\, d_0^1\, d_1^1) = (6\ 2\ 1\ -1)$$

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 2.13 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.

Biến 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 2.14.

$$\begin{bmatrix} x'_0 \\ d^1_0 \\ x'_1 \\ d^1_1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Hình 2.14. 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 d^1_0, d^1_1 và các giá trị trung bình x'_0, x'_1 được tìm thấy xen kẽ trong ma trận kết quả *w*. Các giá trị trung bình được đưa vào ma trận $x' = [x'_0\ x'_1\ 0\ 0]^T$ 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 [2] 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 2.15, 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.

Phương pháp biến đổi dạng sóng Haar thường được sử dụng để phục vụ cho việc *đánh chỉ mục* (indexing) các chuỗi thời gian hoặc dùng để thu giảm số chiều cho các chuỗi thời gian trong bài toán tìm kiếm chuỗi con bất thường nhờ ưu điểm có thể biểu diễn chuỗi thời gian dưới nhiều mức phân giải khác nhau và bảo toàn được khoản cách Euclid của các chuội thời gian trong quá trình biến đổi.

```
// 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])/√2
                dummy vector[i + w]= (input vector[2∗i]−input vector[2∗i+1])/√2
        end for
for i =0; i< (w ∗ 2); i++ do
        output vector[i]= dummy vector[i]
end for
end while
```

Hình 2.15. Giải thuật biến đổi dạng sóng Haar của Fu và các cộng sự

2.5.3. Phương pháp biểu diễn SAX

Phương pháp biểu diễn SAX, được đề xuất bởi J. Lin và các cộng sự trong bài báo [16], 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.

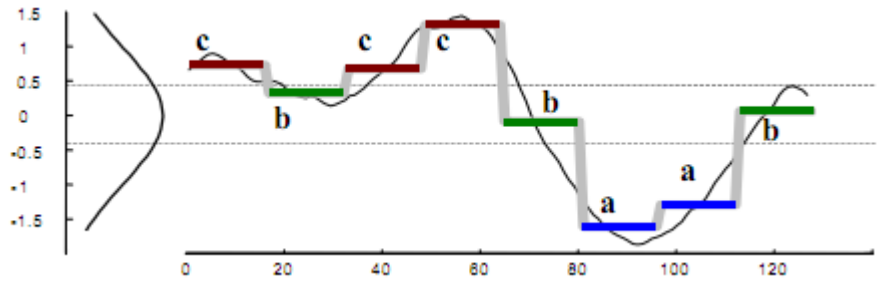
Bước thứ hai: xác định các *điểm chia* (breakpoints). Các điểm chia là một danh sách các điểm có thứ thự $B = B_1, B_2, ..., B_{a-1}$ sau cho phần diện tích dưới đường cong $N(0,1)$ Gauss từ B_i đến B_{i+1} bằng $1/a$ với a là một số nguyên. Các điểm chia này thường được xác định trong một bảng thống kê. Hình 2.16 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 $alpha_i$ là chữ cái thứ i trong bảng chữ cái tiếng Anh ($alpha_1 = a$, $alpha_2 = b$,…) thì một phần tử t_i của vector T sẽ được ánh xạ thành $alpha_j$ nếu và chỉ nếu $B_{j-1} \leq t_i < B_j$, với $B_0 = -\infty$ và $B_a = +\infty$.

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 2.17 bên dưới là minh họa cho biểu diễn SAX của một chuỗi thời gian với $a = 3$ và 2 điểm chia $B_1 = -0.43$, $B_2 = 0.43$.

β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Hình 2.16. Bảng các điểm chia với a từ 3 đến 10



Hình 2.17. Chuỗi thời gian được biểu diễn thành chuỗi *cbccbaab*

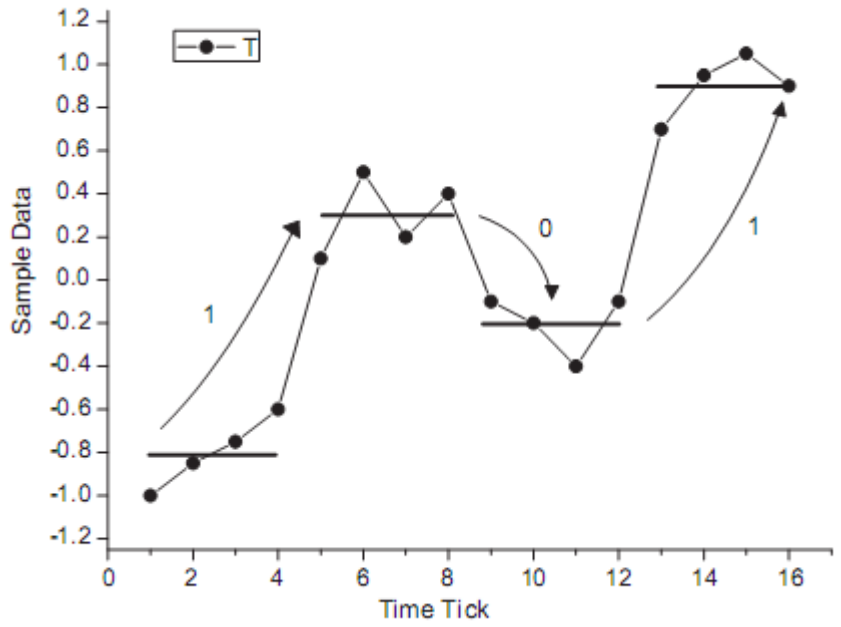
Trong bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian, phương pháp biểu diễn SAX được dùng để xây dựng các cấu trúc chỉ mục hỗ trợ cho việc tìm kiếm [16] hay dùng để rút ra các *luật văn phạm* (grammar rule) hỗ trợ cho việc xác định các chuỗi con bất thường [25].

2.5.4. Phương pháp biểu diễn bit bằng PAA

Phương pháp biểu diễn bit bằng PAA được G. Li và các cộng sự sử dụng để thu giảm số chiều và rời rạc hóa dữ liệu trong bài toán tìm kiếm chuỗi con bất thường [12]. Trong phương pháp này chuỗi thời gian $T = x_1, x_2, \ldots, x_m$ được thu giảm thành các chuỗi gồm các giá trọ 0 và 1 (chuỗi bit) qua hai bước sau. Bước thứ nhất, T được thu giảm về chuỗi $T' = x'_1, x'_2, \ldots, x'_w$, với $1 \leq w \leq m$ bằng phương pháp xấp xỉ PAA. Bước thứ hai chuỗi T' được thu giảm về chuỗi $B = b_1, b_2, \ldots, b_{w-1}$ với b_i thuộc tập $\{0, 1\}$, $1 \leq i \leq w-1$ theo công thức (2.5.2)

$$b_i = \begin{cases} 1 & \text{if } x'_{i+1} > x'_i \\ 0 & \text{otherwise} \end{cases} \qquad (2.5.2)$$

Hình 2.18 là một minh họa cho việc thu giảm một chuỗi thời gian thành một chuỗi bit. Ở đây chuỗi ban đầu là T = -10, -0.85, -0.75, -0.6, 0.1, 0.5, 0.2, 0.4, -0.1, -0.2, -0.4, -0.1, 0.7, 0.95, 1.05, 0.9 được thu giảm số chiều bằng phương pháp PAA thành chuỗi T' = -0.8, 0.3, -0.2, 0.9. Áp dụng công thức (2.5.2) cho chuỗi T' sẽ thu được chuỗi B = 1, 0, 1.



Hình 2.18. Minh họa cho việc thu giảm một chuỗi thời gian về thành một chuỗi bit bằng PAA.

Phương pháp biểu diễn bit không những có khả năng loại bỏ nhiễu trên chuỗi thời giàn mà còn giữ được các *xu hướng* (trend) chính của chuỗi ban đầu.

Cho hai chuỗi thời gian S và T, gọi $s = \{s_1, s_2, \ldots, s_{w-1}\}$ và $t = \{t_1, t_2, \ldots, t_{w-1}\}$ lần lượt là biểu diễn bit của S và T . Độ tương tự của hai chuỗi s và t (ký hiệu *BitSeries_Dist(s, t)*)được tính bằng công thức:

$$BitSeries\_Dist(s,t) = \frac{1}{w-1}\sum_{1}^{w-1} BDist(s_i, t_i) \qquad (2.5.3)$$

Với $BDist(s_i, t_i) = \begin{cases} 1 \text{ if } s_i \neq t_i \\ 0 \quad otherwise \end{cases}$ (2.5.4)

Dễ thấy *BitSeries_Dist(s, t)* luôn không amh, có tính đối xứng và thỏa mãn bất đẳng thức tam giác.

Chương 3
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. 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 [6][30][2][25]. 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 [22]. 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.

3.1. Các công trình liên quan đến phân đoạn chuỗi thời gian

Trong mô hình tìm kiếm chuỗi con bất thường mà chúng tôi hiện thực trong luận văn này, bước *phân đoạn* (segmentation) chuỗi thời gian thành những đoạn con giữ vai trò rất quan trọng. Nếu giải thuật phân đoạn hiệu quả có thể sẽ tách được các chuỗi con bất thường khỏi các chuỗi con bình thường và giúp cho thủ tục tìm kiếm chính xác hơn. Vì vậy chúng tôi dành mục này để trình bày các công trình liên quan đến phân đoạn chuỗi thời gian thành các đoạn con.

3.1.1. Cách phân loại các kỹ thuật phân đoạn chuỗi thời gian của E. Keogh và các cộng sự

Có nhiều giải thuật phân đoạn khác nhau nhưng E. Keogh và các cộng sự trong [8] 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).

3.1.1.1 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 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 3.1 là bảng tóm tắt các ký hiệu dùng trong mục 3.1.1.1 này. Mã giả của giải thuật cửa sổ trượt được thể hiện dưới Hình 3.1.

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 $T= t_1, t_2, t_3, \ldots, t_n$
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 3.1. Các ký hiệu sử dụng trong mục 3.1.1

```
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 3.1. Giải thuật cửa sổ trượt.

3.1.1.2 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 3.2 là mã giả cho giải thuật này. Hàm số *improve_splitting_here(T,i)* trả về sai số xấp xỉ 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 3.2. Giải thuật từ trên xuống.

3.1.1.3 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 3.3 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 3.3. Giải thuật từ dưới lên.

3.1.2. 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 dưới lên. Mỗi loại đều có ưu nhược điểm riêng. D. Lemire trong bài báo [5] 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 $y = a_1x + b_1$ và $y = a_2x + b_2$ 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 3.4 là mã giả cho giải thuật.

INPUT: *Time Series (x_i, y_i) 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 [x_p, x_q)*
S empty list
$d \leftarrow N - 1$; $S \leftarrow (0, n, d, E(0, n, d))$; $b \leftarrow k - d$
while *$b - d \geq 0$* **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
    $b \leftarrow b - d$
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 $0 \leq d' \leq q - 1$
    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 3.4. 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.

3.1.3. Giải thuật phân đoạn SWAB

E. Keogh và các cộng sự trong bài báo [8] 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 3.5 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 3.5. Giải thuật SWAB

3.1.4. Giải thuật phân đoạn dựa vào điểm cực trị quan trọng

Các điểm cực trị quan trọng được sử dụng để phân đoạn chuỗi thời gian thành các đoạn con trong giải thuật EP-C [4]. Ban đầu các điểm cực trị quan trọng của chuỗi thời gian $EP = (ep_1, ep_2, ..., ep_l)$ được xác định. Các chuỗi con được rút trích ra từ chuỗi thời gian bằng một cửa sổ trượt qua các điểm ep_i, với $1 \leq i \leq l-2$.

Tại mỗi điểm một chuỗi con có chiều dài từ ep_i đến ep_{i+2} sẽ được rút ra từ chuỗi ban đầu. Như vậy sau khi kết thúc bước lặp ta đã phân đoạn chuỗi thời gian thành các chuỗi con có chiều dài khác nhau. Giải thuật Hình 3.6 tìm các điểm cực trị được quan trong của chuỗi $T = t_1, t_2, \ldots, t_n$. Giải thuật bắt đầu bằng việc tìm hai điểm cực trị ban đầu sau đó sẽ tìm kiếm lần lượt các điểm cực đại và cực tiểu xen kẽ nhau. Dễ thấy giải thuật có độ phức tạp tuyến tính vì chỉ duyệt qua chuỗi thời gian một lần.

```
i = FIND-FIRST-TWO
if i < n and T[i] > T[1] then i = FIND-MIN(i)
while i < n do
  i = FIND-MAX(i);  i = FIND-MIN(i)
-------------------------------------------------------------------------------
FIND-FIRST-TWO
iMin = 1; iMax = 1; i = 2
while i < n and T[i] / T[iMin] < R and T[iMax] / T[i] < R do
  if T[i] < T[iMin] then iMin = i
  if T[i] > T[iMax] then iMax = i
if iMin < iMax then  output(T[iMin], iMin); output(T[iMax], iMax)
else  output(T[iMax], iMax); output(T[iMin], iMin)
return i
-------------------------------------------------------------------------------
FIND-MIN(i)
iMin = i
while i < n and T[i]/T[iMin] < R do
  if T[i] < T[iMin] then iMin = i
  i = i + 1; output(T[iMin], iMin)
return i
-------------------------------------------------------------------------------
FIND-MAX(i)
Finding the first significant maximum after the i-th point
iMax = i
while i < n and T[iMax] / T[i] < R do
  if T[i] > T[iMax] then iMax = i
  i = i + 1; output(T[iMax], iMax); return i
```

Hình 3.6. Giải thuật tìm các điểm cực trị quan trọng

3.2. Các công trình về tìm kiếm bất thường trong dữ liệu chuỗi thời gian

Bài toán tìm chuỗi con bất thường trong dữ liệu chuỗi thời gian gần đây rất được quan tâm nghiên cứu, nhiều bài báo đã được công bố. Một số bài báo đề xuất giải thuật tìm kiếm các chuỗi con bất thường trong dữ liệu chuỗi thời gian thuộc một lĩnh vực cụ thể như bài báo của Chuah và các cộng sự năm 2007 [21] hay bài báo của Wei và các cộng sự năm 2008 [20]. Trong khi một số bài báo khác đề xuất các giải thuật tổng quát có thể áp dụng cho các chuỗi thời gian thuộc các lĩnh vực khác nhau. Một số giải thuật tiêu biểu có thể áp dụng cho các chuỗi thời gian thuộc nhiều lĩnh vực khác nhau:

- Giải thuật HOT SAX đề xuất bởi Keogh và các cộng sự [6] dựa vào biểu diễn SAX và cửa sổ trượt để tìm kiếm chuỗi con bất thường nhất.
- Giải thuật WAT đề xuất bởi Bu và các cộng sự [30] dựa vào biến đổi dạng sóng Harr và cửa sổ trược để tìm kiếm các chuỗi con bất thường nhất.
- Giải thuật TARZAN đề xuất bởi Keogh và các cộng sự [9] dựa vào *cây hậu tố* (suffix tree) và *mô hình Markov ẩn* (hidden Markov model) để tìm chuỗi con bất thường.
- Giải thuật đề xuất bởi Oliveira và các cộng sự [1] dùng mạng *neural nhân tạo* (artificial neural network) để tìm chuỗi con bất thường.
- Giải thuật dựa trên *SVM* (support vector machine) đề xuất bởi Ma và các cộng sự [17] để tìm chuỗi con bất thường.
- Giải thuật dựa vào *máy hữu hạn trạng thái* (finite state automaton) đề xuất bởi Salvador và các cộng sự [27] để tìm chuỗi con bất thường.
- Giải thuật đề xuất bởi Li và các cộng sự [12] dựa vào biểu diễn bit bằng PAA và gom cụm để tìm chuỗi con bất thường.
- Giải thuật đề xuất bởi Leng và các cộng sự [22] dùng phân đoạn và *phương pháp chiều dài biến đổi* (variable length methods) để tìm các chuỗi con bất thường có chiều dài khác nhau.

- Giải thuật dựa trên biểu diễn SAX và *luật văn phạm* (grammar rule) được đề xuất bởi Senin và các cộng sự [25] để tìm các chuỗi con bất thường.
- Giải thuật dựa trên phân đoạn và trung *bình cục bộ* (local means) và *độ lệch chuẩn* (standard deviations) được đề xuất bởi Dani và các cộng sự [23] để tìm chuỗi con bất thường.
- Giải thuật của Kha và các cộng sự [24] đề xuất dựa vào phân đoạn và gom cụm để tìm kiếm các chuỗi con bất thường.

Sau đây chúng tôi trình bày chi tiết một số giải thuật chính.

3.2.1. Giải thuật HOT SAX

E. Keogh và các cộng sự trong [6] 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 p_i là điểm bắt đầu của chuỗi con bất thường thứ i với $1 \leq i \leq k$ thì $|p-p_i| \geq n$.

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* (brute force). 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 giản nhưng độ phức tạp tính toán là $O(m^2)$ 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, không nhất thiết phải tính được khoảng cách

giữa nó đến chuỗi con khớp không tầm thường gần nó nhất mà chỉ cần 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 3.7 thì 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 cải tiến được minh họa ở hình 3.8. Ở vòng lặp ngoài 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 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 ]
```

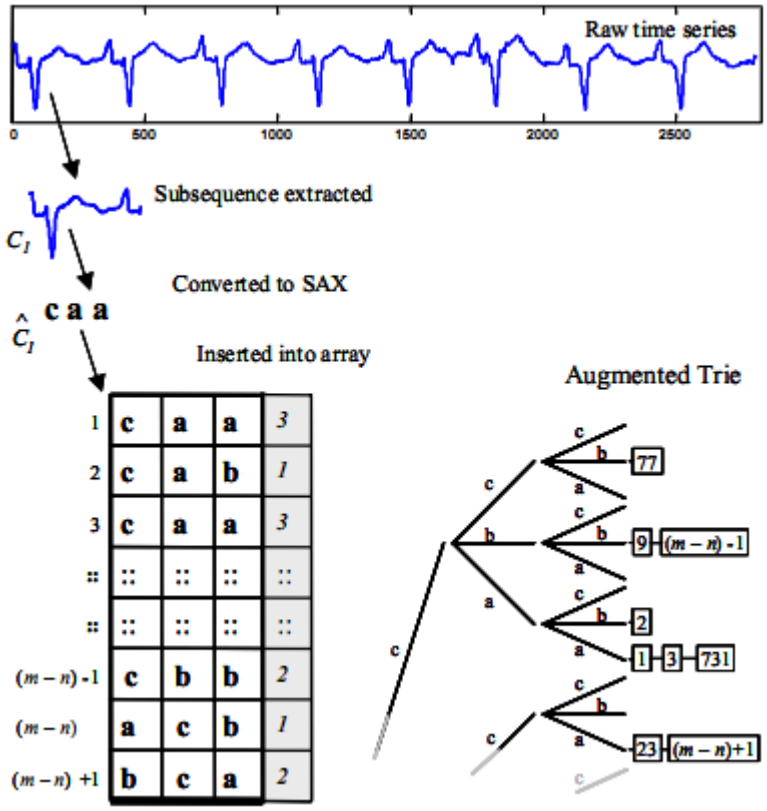
Hình 3.7. Giải thuật vét cạn tìm chuỗi con bất thường

```
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 ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)  < best_so_far_dist
            Break                                  // Break out of Inner Loop
         End
         IF  Dist ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)  < nearest_neighbor_dist
            nearest_neighbor_dist = Dist ((t_p,...,t_p+n-1), (t_q,...,t_q+n-1)
         End
      End                                           // End non-self match test
   End                                              // End Inner Loop
    IF nearest_neighbor_dist > best_so_far_dist
      best_so_far_dist = nearest_neighbor_dist
      best_so_far_loc  = p
    End
End                                                 // End Outer Loop
Return[ best_so_far_dist, best_so_far_loc ]
```

Hình 3.8. 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 3.8, 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 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 lá đó. Cấu trúc cây này giúp 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 3.9 . 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. Đôi khi một số tác giả dùng chiều dài của một phân đoạn PAA làm tham số thay cho *w*. Dễ thấy $w = \left\lceil \frac{n}{PAA\_LENGTH} \right\rceil$, với *PAA_LENGTH* là chiều dài của phân đoạn PAA.

. E. Keogh và các cộng sự cũng đã chứng minh trong [6] rằng 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.

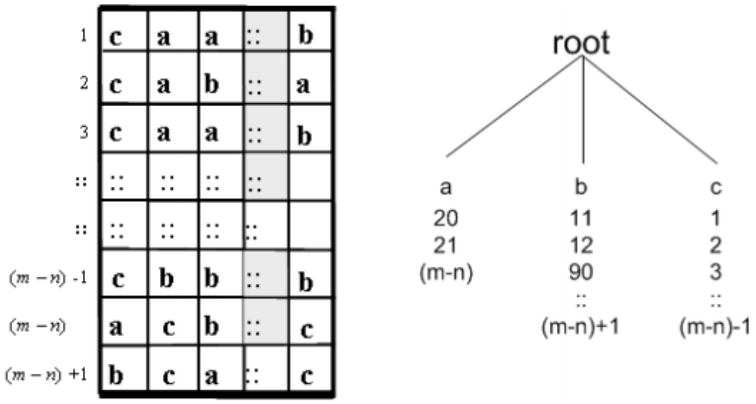
3.2.2. Giải thuật WAT

Giải thuật *WAT* (**w**avelet and **a**ugmented **t**rie) được đề xuất bởi Y. Bu và các cộng sự trong [30] 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.

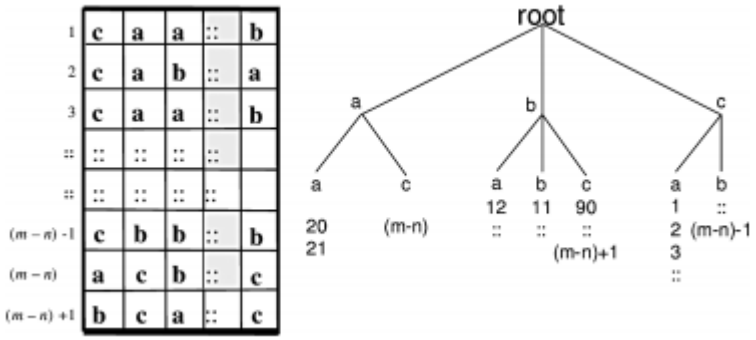
Các vector hệ số Haar của các chuỗi có con sẽ được chuẩn hóa thành các chuỗi có trung bình bằng 0 và độ lệch chuẩn bằng 1. Để giảm độ phức tạp trong việc so sánh, các chuỗi này sẽ được rời rạc hóa thành các từ SAX giống như trong giải thuật HOT SAX. Giả sử một chuỗi thời gian con *C* có kích thước *n* được biến đổi sang dạng sóng Haar thành chuỗi $C' = \{c'_1, c'_2, ..., c'_n\}$. Gọi $alpha_i$ là chữ cái thứ *i* trong bảng chữ cái tiếng Anh. Gọi $B = b_1, b_2, ..., b_{a-1}$ là chuỗi các điểm chia sau cho phần diện tích dưới đường cong sác xuất *N(0,1)* Gauss giữa b_i, b_{i+1} bằng *1/a* với

b_0 và b_a tương ứng bằng -∞ và +∞. Hệ số c'_i của biến đổi Haar sẽ được ánh xạ thành chữ cái $alpha_j$ nếu và chỉ nếu $b_{j-1} \leq c'_i \leq b_j$.

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 độ 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 mỗi từ 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 3.10. Chữ cái đầu tiên của từ được xem xét khi phân tách nút gốc của cây



Hình 3.11. Chữ cái thứ hai được xem xét khi tiến hành phân tách các nút *a, b, c*.

Hình 3.10 và 3.11 là một ví dụ là minh họa cho cách thức xây dựng cấu trúc cây này. Ở hình 3.10, khi phân tách nút gốc, 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 3.11 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 được 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 3.8 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 3.11 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 3.8 *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.

3.2.3. Giải thuật tìm kiếm chuỗi con bất thường dựa trên gom cụm các biểu diễn bit bằng PAA

G. Li và các cộng sự đề xuất một giải thuật tìm kiếm chuỗi con bất thường nhất có chiều dài *n* dựa trên việc gom cụm các biểu diễn bit bằng PAA của các chuỗi con trong bài báo năm 2013 [12]. Giải thuật bao gồm hai bước. Bước thứ nhất, các chuỗi con có chiều dài *n* của chuỗi thời gian sẽ được biểu diễn thành các chuỗi bit theo giải thuật trình bày ở mục 2.5.4 và gom cụm. Các chuỗi con có hành vi biến đổi giống nhau sẽ được gom vào cùng một cụm. Bước thứ hai, dựa vào kết quả gom cụm, giải thuật tiến hành sắp xếp thứ tự duyệt của các chuỗi con trong vòng lặp trong và vòng lặp ngoài của giải thuật vét cạn để tìm kiếm chuỗi con bất thường nhất.

3.2.3.1 Gom cụm các chuỗi bit

Tất cả các chuỗi con có chiều dài *n* của chuỗi thời gian được thu giảm thành các chuỗi bit và gom cụm bằng giải thuật K-Medoids cải tiến. Trong bài báo [12], giải thuật gom cụm được gọi là BitCluster. Giải thuật cần hai thông số: tập hợp các biểu diễn bit của các chuỗi con *BD* và số cụm *k*. Ban đầu *k* chuỗi bit được chọn ngẫu nhiên làm trung tâm của *k* cụm. Sau đó giải thuật gán các chuỗi bit còn lại vào

các cụm gần nhất dựa trên công thức tính khoảng cách giữa các chuỗi bit (2.5.3). Mã giả cho giải thuật *BitCluster* cho dưới hình 3.12

```
Algorithm. Clustering algorithm BitCluster based on PAA bit
serialization
INPUT: Bit series dataset BD, clustering parameter k
OUTPUT: set of k clusters bCluster, cluster identification tags
PatternCTag
1       for i=1 to k do
2               RandInitial(bCluster[i].Center);
3               bCluster[i].NumMembers=1;
4       end of for
5       for j=1 to |BD|do
6               if (!BelongtoCenter(bCluster, j, k))
7                       p=1;
8                       for i=1 to k do
9                               if (BitSeries_Dist(BD[j],
                                BD[i])<BitSeries_Dist(BD[j],BD[bCluster[p].Center]))
10                                      p=i;
11                      end of for
12                      PatternCTag[j]=p;
13                      bCluster[p].Member[bCluster[p].NumMembers]=j;
14                      bCluster[p].NumMembers++;
15      end of for
16      for i=1 to k do
17              bCluster[i].Radius=CalClusterRadius(bCluster[i]);
18      end of for
```

Hình 3.12. Giải thuật BitCluster

Trong hình 3.12, *bCluster* là mảng các đối tượng cụm. Mỗi đối tượng cụm có các thuộc tính: tâm của cụng, bán kính cụm, số lượng thành viên thuộc cụm và các thành viên của cụm. Dòng 1-4 thực hiện việc khởi tạo ngẫu nhiên *k* trung tâm cụm. Dòng 5-15 thực hiện việc gom cụm các chuỗi bit. Mỗi chuỗi bit chưa được gom cụm sẽ được gán vào cụm có tâm gần nó nhất. Dòng 16-18 tính bán kính cho từng

cụm. Bán kính cụm là khoảng cách lớn nhất từ một thành viên đến tâm của cụm. Để thuận tiện cho việc tìm kiếm chuỗi con bất thường ở bước thứ hai, công thức Euclid được sử dụng để tính bán kính cụm.

Độ phức tạp của giải thuật BitCluster là $O(k*|BD|)$, $|BD|$ là số lượng các chuỗi bit được gom cụm. Tham số k được chọn dựa vào bản chất của tập dữ liệu và được kiểm tra bằng thực nghiệm.

3.2.3.2 Chiến thuật tăng tốc cho giải thuật vét cạn dựa trên kết quả gom cụm

Để tăng tốc cho giả thuật vét cạn, G. Li và các cộng sự [12] đề xuất giải thuật BitClusterDiscord với hai heuristic sau:

Heuristic 1: Trong vòng lặp ngoài của giải thuật vét cạn (hình 3.8), các chuỗi con có biểu diễn bit thuộc cụm có số phần tử ít nhất sẽ được duyệt trước. Với mỗi chuỗi con *p* được chọn ở vòng lặp ngoài, các chuỗi con có biểu diễn bit thuộc cùng một cụm với chuỗi biểu diễn bit của *p* sẽ được duyệt trước ở vóng lặp trong.

Heuristic 2: Gọi *C1* và *C2* là hai cụm có tâm và bán kính lần lượt là *O1, O2* và *r1, r2*. Gọi *best_so_far_dist* là một ngưỡng cho trước. Nếu *Dist(O1, O2) – r1 – r2 > best_so_far_dist* thì không tồn tại các cặp lân cận gần nhất mà một phần tử thuộc *C1* và phần tử còn lại thuộc *C2*

Hình 3.13 là mã giả của giải thuật. Dòng 3-6, giải thuật thu giảm các chuỗi con thành các chuỗi bit và tiến hành gom cụm các chuỗi này theo giải thuật BitCluster. Dòng 7-8, hàm *Pr oduceOuterOrder(), Pr oduceInnerOrder()* sắp xếp thứ tự duyệt các chuỗi con theo heuristic 1. Dòng 13-14 dùng heuristic 2 để dừng sớm vòng lặp trong.

```
Algorithm . Bit representation clustering based discord
discovery BitClusterDiscord
INPUT: Time series T with length n, discord length m,
compression ratio by PAA cr, clustering parameter k
OUTPUT: Discord starting position best_so_far_loc, the largest
distance from discord to nearest neighbor best_so_far_dist
1       best_so_far_loc = -1;
2       best_so_far_dist =0;
3       for i=1 to n-m+1
4               for subsequence t_i,...t_i+m-1, use PAA bit serialization;
5       end of for
6       BitCluster(BD, k);
7       Pr oduceOuterOrder();
8       Pr oduceInnerOrder();
9        for p in T by outer loop order
10              nearest_neighbor_dist = inf inity;
11               for q in T by inner loop order
12                      if (|p - q| ≥ m)
13                              if (distance between clusters which p and q belong
                                        to>=best_so_far_dist)
14                                      break;
15                      if(Dist(t_p...t_p+m-1, t_q..t_q+m-1)< nearest_neighbor_dist)
16                              nearest_neighbor_dist = Dist(t_p...t_p+m-1, t_q..t_q+m-1);
17                      if (Dist(t_p...t_p+m-1, t_q..t_q+m-1)< best_so_far_dist)
18                              break;
19              end of for
20              if (nearest_neighbor_dist > best_so_far_dist)
21                       best_so_far_dist = nearest_neighbor_dist;
22                      best_so_far_loc = p;
23      end of for
```

Hình 3.13. Giải thuật BitClusterDiscord.

Giải thuật BitClusterDiscord được chứng minh bằng thực nghiệm có thể tìm kiếm được chuỗi con bất thường hiệu quả và có tốc độ hội tụ nhanh hơn giải thuật vét cạn. Tuy nhiên giải thuật này cần phải xác định trước chiều dài *m* của chuỗi con

bất thường. Việc gom cụm các chuỗi bit cần khởi tạo ngẫu nhiên *k* trung tâm cụm nên giải thuật cần được chạy nhiều lần để đạt kết quả đáng tin cậy.

3.2.4. 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 năm 2008 [22] 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 $f(t) = b_0 + b_1t + b_2t^2$ cho đến khi sai số lớn hơn một giá trị ε_1 mà người dùng chọn. Chuỗi con thứ *i* tìm được biểu diễn bằng ký hiệu (s_i, e_i) với s_i là vị trí bắt đầu và e_i 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 $s_{i+1}=s_i+j$ với *j* là số nguyên bé nhất sau cho khoảng cách từ hai chuỗi (s_i, e_i) và $(s_i + j, e_i + j)$ lớn hơn ε_2 mà người dùng chọn hoặc $j \geq (e_i - s_i)$. 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 l_{min} và l_{max} là chiều dài lớn nhất và nhỏ nhất của các chuỗi con và xây dựng một ma trận khoảng cách $D = (d_{ij})_{mxm}$ với *m* là số chuỗi con. Giá trị d_{ij} là khoảng cách của chuỗi con thứ *i* và thứ *j*, được tính bằng công thức:

$$d_{ij} = \min_{l_{\min} \leq l \leq l_{\max}} Dist((s_i, e_i), (s_j, s_j + l)) \qquad (3.2.1).$$

Các tác giả khẳng định trong [22] 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 (anomaly factor) 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 α_0 thì chuỗi đó là chuỗi con bất thường. Hai chuỗi con bất thường (s_i, e_i) và (s_j, e_j) nếu $s_i \leq s_j \leq e_i$ thì trộn chúng lại thành một chuỗi con (s_i, e_j). Ngược lại *nếu* $s_j \leq s_i \leq e_j$ thì trộn chúng lại thành một chuỗi con (s_j, e_i).

Trong bước phân đoạn, giải thuật có hai tham số đầu vào là ε_1 và ε_2. Tham số ε_1 quyết định chiều dài của các chuỗi con. Nếu ε_1 càng lớn thì các chuỗi con càng dài và ngược lại. Tham số ε_2 quy định số lượng chuỗi con được phân đoạn và có thể loại bỏ những chuỗi con ít có khả năng là chuỗi con bất thường. Các tham số này được ước lượng chính xác sẽ giúp tăng tốc độ và độ chính xác của giải thuật.

Ở bước tính toán hệ số bất thường, giải thuật cần hai tham số là *k* và *a*. Tham số *k* thường là một số nguyên dương nhỏ thể hiện việc khoảng cách đến lân cận thứ *k* của chuỗi con sẽ được dùng để tính hệ số bất thường. Tham số *a* là ngưỡng bất thường, dùng để đánh giá một chuỗi con là bất thường. *a* thường là một số thực lớn hơn 0.

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 độ đo 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 độ đo xoắn thời gian động là $O(m*n)$ với *m*, *n* là chiều dài của hai chuỗi thời gian.

3.3. Kết luận.

Các giải thuật HOT SAX, WAT và giải thuật dựa trên gom cụm và biểu diễn bit bằng PAA đều có nhược điểm chung là đòi hỏi phải biết trước chiều dài của chuỗi con bất thường. Giải thuật đề xuất bởi Leng và các cộng sự [22] có khả năng

tìm kiếm được các chuỗi con bất thường có chiều dài khác nhau mà không cần xác định trước chiều dài của các chuỗi con bất thường. Tuy nhiên, phương pháp này phải sử dụng giải thuật xoắn thời gian động để tính khoảng cách giữa hai chuỗi thời gian có chiều dài khác nhau. Điều này làm tăng độ phức tạp tính toán của phương pháp và làm cho phương pháp khó có khả năng áp dụng cho các chuỗi thời gian có độ dài lớn. Chương sau, chúng tôi sẽ trình bày các phương pháp chúng tôi sử dụng để tăng tốc cho giải thuật của Leng và các cộng sự.

Chương 4
PHƯƠNG PHÁP GIẢI QUYẾT VẤN ĐỀ

Trong chương này, chúng tôi đề xuất một giải thuật tính khoảng cách giữa các chuỗi thời gian có độ dài khác nhau bằng công thức tính khoảng cách Euclid và phép *biến hình vị tự* (homothetic transformation) để cải thiện tốc độ cho mô hình tìm kiếm chuỗi con bất thường của M. Leng và các cộng sự [22] . Chúng tôi cũng đề xuất thêm việc sử dụng phương pháp phân đoạn bằng các điểm cực trị quan trọng vào bước phân đoạn của giải thuật trên để giúp việc ước lượng các tham số của mô hình đơn giản hơn.

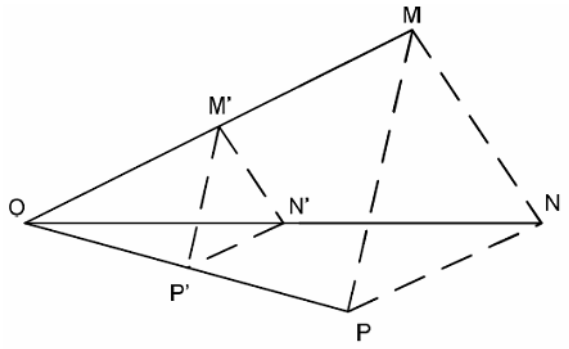
4.1. Tính khoảng cách giữa hai chuỗi thời gian có độ dài khác nhau

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 chúng tôi sẽ tiếp cận theo phương pháp của M. Leng và các cộng sự bởi vì phương pháp này 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, hơn nữa bằng cách dùng phương pháp chiều dài biến đổi (xem công thức (3.2.1)) để tính khoảng cách của hai chuỗi con cần đến $(l_{max} - l_{min} + 1)$ lần tính khoảng cách xoắn thời gian động, với l_{max} và l_{min} lần lượt là chiều dài cùa chuỗi con dài nhất và chuỗi con ngắn nhất trong tập các chuỗi con được xem xét. Điều này làm giảm tốc độ tìm kiếm của giải thuật.

Để giải quyết khó khăn này chúng tôi đề xuất hai cải tiến. Thứ nhất, chúng tôi sử dụng pháp biến hình vị tự để đưa các chuỗi con về cùng chiều dài và áp dụng cộng thức Euclid để tính khoảng cách của các chuỗi con. Thứ hai, chúng tôi đề xuất thêm một tham số *r* để giảm số lần phải tính khoảng cách trong công thức (3.2.1).

4.1.1. Giải thuật tính khoảng cách dựa trên pháp biến hình vị tự và công thức Euclid.

Phép biến hình vị tự thay đổi kích thước của một đối tượng hình học trong không gian affine mà không làm thay đổi hình dạng của nó. Dó đó việc dùng phép biến hình vị tự không ảnh hưởng đến việc so sánh hình dạng của các chuỗi. Một pháp biến hình vị tự tâm *O*, tỉ số *k* biến một điểm *M* thành điểm *M'* sao cho $\overline{OM} = k*\overline{OM'}$. Phép biến hình vị tự đã được sử dụng trong bài toán tìm motif trong dữ liệu chuỗi thời gian ở công trình của C.D. Truong cùng các cộng sự [4] và công trình của Huỳnh Nguyễn Tín cùng các cộng sự [15] để tính khoảng cách của các chuỗi thời gian có chiều dài khác nhau. Phép vị tự để biến đổi một chuỗi thời gian *T* có chiều dài *n* (*T={y_1, y_2, ..., y_n})* thành *n'* được thực hiện theo các bước như sau. Thứ nhất gọi *Y_MAX = MAX{y_1, ..., y_n}, Y_MIN = MIN{y_1, ...,y_n }*. 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 $k = \frac{n'}{n}$. Hình 4.1 là ví dụ về phép biến hình vị tự, ba điểm M', P' và N' được biến thành ba điểm M, P , N. Tam giác MNP có diện tích lớn hơn tam giác M'N'P' nhưng cả hai có đều có hình dạng giống nhau.



Hình 4.1. Phép biến hình vị tự

Để tính khoảng cách giữa hai chuỗi thời gian, trước hết chúng tôi sẽ kiểm tra chiều dài của hai chuỗi nhập, nếu chúng dài bằng nhau thì áp dụng ngay công thức Euclid để tính khoảng cách của chúng, nếu không chúng tôi dùng phép biến hình vị tự để đưa chúng về cùng độ dài và sau đó áp dụng công thức Euclid. Độ dài được chọn là độ dài trung bình của hai chuỗi. Ví dụ để tính khoảng cách của hai chuỗi con A và B có chiều dài lần lượt là 8 và 10, chúng tôi dùng phép biến hình vị tự để biến chuỗi A thành chuỗi A' và chuỗi B thành chuỗi B' cùng có độ dài là 9 rồi áp dụng công thức Euclid để tính khoảng cách cho hai chuỗi mới này. Mã giả của của phương pháp này được cho trong hình 4.2 dưới đây.

```
Function Dist(X, Y)
if (X.length == Y.length)
        return Euclid(X,Y)
Else
        MEAN = |(Y.length + X.length)| / 2
        Y' = Homothetic(Y,MEAN)
        X' = Homothetic(X,MEAN)
        return Euclid(X',Y')
```

Hình 4.2 Mã giả giải thuật tính khoảng cách.

Công thức Euclid chúng tôi sử dụng là công thức tính khoảng cách Euclid cực tiểu (cộng thức 2.4.2) để loại ra sự khác biệt theo trục tung của hai chuỗi thời gian. Dễ thấy độ phức tạp tính toán của phép biến hình vị tự là $O((n+ m) / 2)$ và của công thức tính Euclicd *là* $O((n+m) / 2)$ nên phương pháp mới có độ phức tạp là $O((n+m) / 2)$, nhanh hơn nhiều so với phương pháp xoắn thời gian động (có độ phức tạp tính toán $O(m*n)$). Ở đây, m và n là chiều dài của hai chuỗi thời gian cần tính khoảng cách.

4.1.2. Giảm số lần tính khoảng cách bằng tham số *r*

Theo công thức tính khoảng cách dựa trên phương pháp chiều dài biến đổi do Leng và các cộng sự đề xuất [22] (công thức 3.2.1), để tính được khoảng các giữa hai chuỗi thời gian i và j bất kỳ, cần thực hiện $(l_{max} – l_{min} + 1)$ lần tính khoảng cách,

ở đây l_{min} và l_{max} lần lượt là chiều dài của chuỗi con dài nhất và chiều dài của chuỗi con ngắn nhất trong các chuỗi con được phân đoạn từ chuỗi con ban đầu. Nếu l_{max} lớn hơn nhiều so với l_{min}, giải thuật phải thực hiện rất nhiều lần tính khoảng cách để xác định được khoảng cách của hai chuỗi con.

Để giảm số lần tính khoảng cách, chúng tôi đề xuất thêm tham số *r*. Tham số này dùng để xác định cận trên l_{upper} và cận dưới l_{lower} thay cho l_{max} và l_{min} trong công thức (3.2.1). Các giá trị l_{upper} và l_{lower} được tính theo hai công thức sau:

$$l_{upper} = \lfloor l_{avg}(1+r) \rfloor \qquad (4.1.1)$$

$$l_{lower} = \lceil l_{avg}(1-r) \rceil \qquad (4.1.2)$$

Ở đây l_{avg} là chiều dài trung bình của các chuỗi con được phân đoạn ban đầu.

Thay l_{upper} và l_{lower} cho l_{max} và l_{min} trong công thức (3.2.1) ta được công thức tính khoảng cách giữa hai chuỗi con *i* và *j*.

$$d_{ij} = \min_{l_{upper} \le l \le l_{lower}} Dist((s_i, e_i), (s_j, s_j + l)) \qquad (4.1.3)$$

Tham số *r* được chọn là một số thực dương nhỏ để giá trị l_{upper} không quá lớn so với l_{lower} mà vẫn đảm báo tính chính xác của giải thuật . Điều này làm giảm số lần tính khoảng cách phải thực hiện. Dựa vào các kết quả thực nghiệm (sẽ trình bày ở chương sau), chúng tôi nhận thấy tham số *r* chỉ biến thiên trong khoảng 0.05 đến 0.25.

4.2. Dùng phương pháp phân đoạn bằng điểm cực trị quan trọng

Để tìm chuỗi con bất thường của một chuỗi thời gian, trước tiên chuỗi thời gian phải được phân đoạn thành các đoạn con. Chất lượng của khâu phân đoạn ảnh hưởng nhiều đến kết quả tìm kiếm chuỗi con bất thường của giải thuật. Như đã trình bày trong mục 3.2.3 của chương 3, chất lượng phân đoạn bằng phương pháp của Leng và các cộng sự phụ thuộc vào hai thông số ε_1 và ε_2. Trong quá trình thực nghiệm (sẽ trình bày ở chương sau), chúng tôi nhận thấy việc ước lượng hai tham số

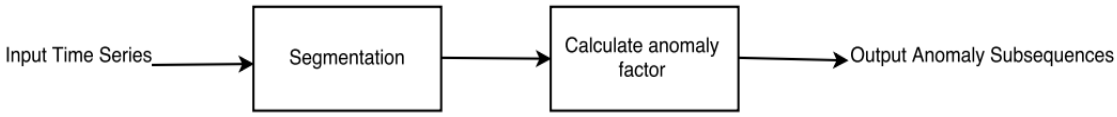
này là việc không dễ. Do đó chúng tôi đề xuất thêm phương pháp phân đoạn bằng các điểm cực trị quan trọng.

Để xác định các điểm cực trị quan trọng cần xác định tỉ số nén R . Tham số R sẽ quy định số chuỗi con được phân đoạn và chiều dài của chúng. Nếu R càng lớn thì số chuỗi con sẽ càng ít và chiều dài của chúng càng lớn. Nếu R càng nhỏ thì số chuỗi con được phân đoạn càng nhiều và chiều dài của chúng càng nhỏ. Để tạo một cận dưới cho chiều dài của các chuỗi con, chúng tôi đưa vào tham số *MIN_LENGH* quy định khoảng cách tối thiểu giữa hai điểm cực trị quan trọng. Khi đó một chuỗi con được phân đoạn sẽ có chiều dài tối thiểu 2* *MIN_LENGH*. Việc quy định chiều dài tối thiểu cho các chuỗi con là cần thiết bởi vì nếu các chuỗi con quá ngắn sẽ không thể hiện được tính khác biệt về hình dáng đối với các chuỗi con lân cận dẫn đến việc bỏ sót các chuỗi con bất thường.

Sau khi xác định các điểm cực trị quan trọng của chuỗi thời gian $EP = (ep_1, ep_2, \ldots, ep_l)$ được xác định. Các chuỗi con được rút ra từ chuỗi thời gian bằng một cửa sổ trượt qua các điểm ep_i, với $1 \leq i \leq l-2$. Tại mỗi điểm một chuỗi con có chiều dài từ ep_i đến ep_{i+2} sẽ được rút trích ra từ chuỗi ban đầu. Do khoảng cách giữa các điểm cực trị khác nhau nên phương pháp này có khả năng phân đoạn chuỗi thời gian thành những chuỗi con có chiều dài khác nhau. Sử dụng phương pháp phân đoạn dựa trên các điểm cực trị quan trọng trong bài toán tìm kiếm chuỗi con bất thường chúng tôi giả định rằng một chuỗi con bất thường, nếu tồn tại, sẽ bắt đầu tại một điểm cực trị quan trọng.

4.3. Mô hình của giải thuật

Giải thuật tìm kiếm chuỗi con bất thường trong dữ liệu chuỗi thời gian bao gồm 2 bước như hình 4.3 dưới đây.



Hình 4.3. Kiến trúc mô hình

Bước đầu tiên chuỗi thời gian sẽ được phận đoạn thành các chuỗi con. Chúng tôi hiện thực cả hai giải thuật: giải thuật phân đoạn bằng cách dùng đa thức bậc hai của M.Leng và các cộng sự [22] và giải thuật sử dụng các điểm cực trị quan trọng. Các chuỗi con sẽ được tính khoảng cách từng đôi một với nhau bằng công thức (3.2.1). Các khoảng cách này tạo thành một ma trận có *m* hàng và *m* cột, với *m* là số chuỗi con được phân đoạn. Dựa vào ma trận khoảng cách này, hệ số bất thường của từng chuỗi con được tính theo định nghĩa 5 trong mục 2.3. Việc chọn giá trị *k* để tính khoảng cách lân cận thứ *k* sẽ được người dùng quyết định. 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.

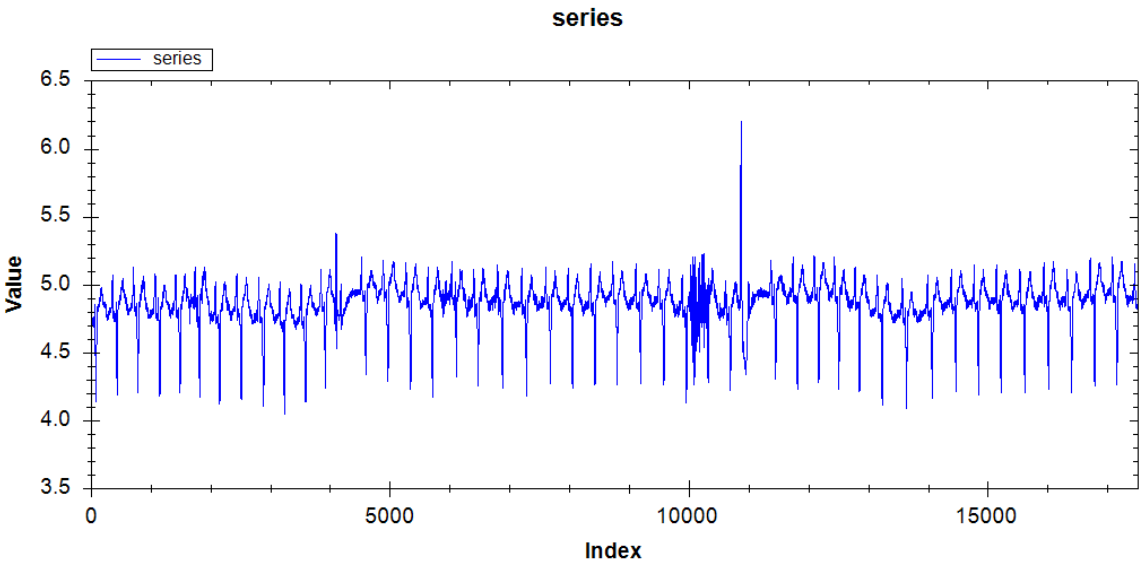
Chương 5
THỰC NGHIỆM

Trong chương này, chúng tôi sẽ trình bày các kết quả mà chúng tôi đã thực hiện để kiểm tra tính hiệu quả của các giải thuật tìm kiếm chuỗi con bất thường mà chúng tôi đã hiện thực bằng cách so sánh với kết quả của giải thuật HOT SAX. Chúng tôi cũng so sánh thời gian chạy của giải thuật khi sử dụng phương pháp tính khoảng cách dựa trên phép biến hình vị tự và thời gian chạy khi sử dụng phương độ đo xoắn thời gian động.

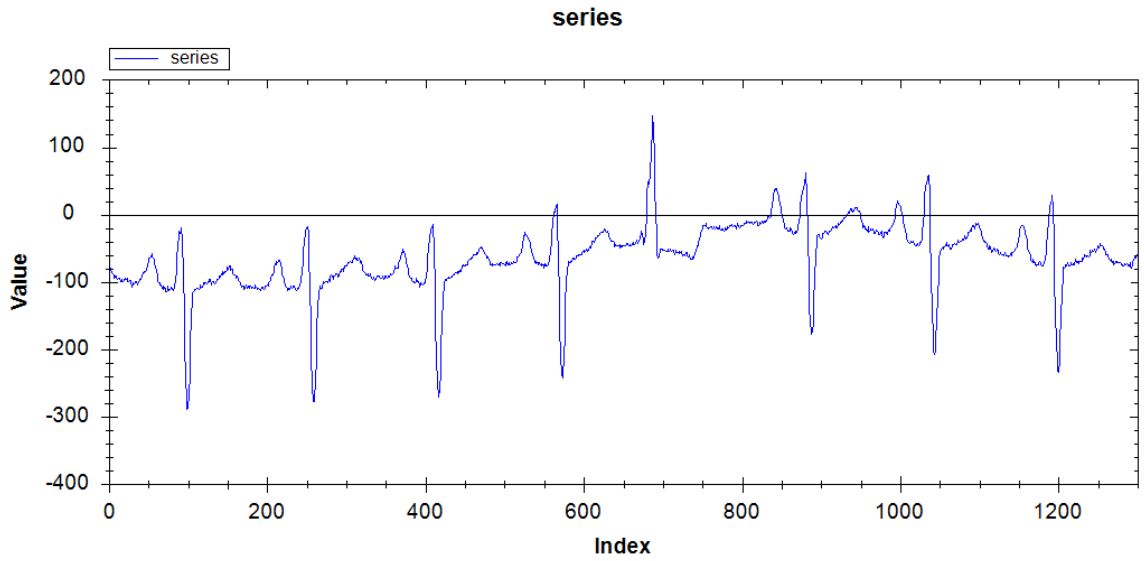
Tất cả các giải thuật đều được hiện thực bằng ngôn ngữ lập trình C# và được chạy thực nghiệm trên máy tính dùng hệ điều hành Microsoft Windows10 32 bit, bộ xử lý Intel® Core™ 2 Duo 2.0GHz, Ram 3072MB.

5.1. Giới thiệu các chuỗi thời gian mẫu

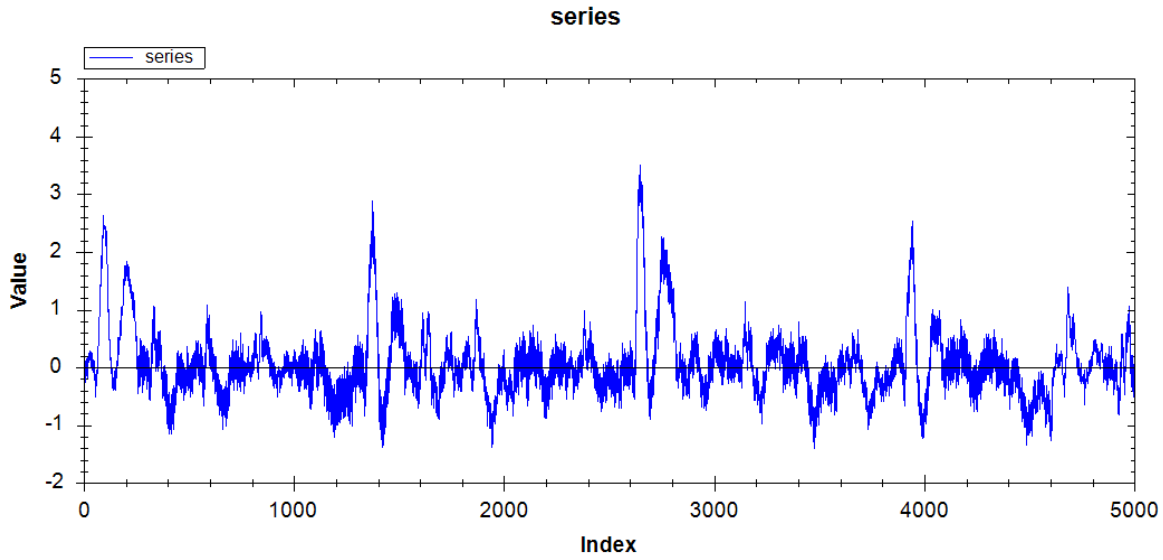
Chúng thôi tiến hành chạy thực nghiệm các giải thuật tìm chuỗi con bất thường trên 8 chuỗi thời gian thuộc các lĩnh vực khác nhau là ECG 108, ECG 308, ERP, Memory, Power Demand In Italy, Dutch Power Demand, Stock20 và TEK16. Các chuỗi thời gian đều được lấy từ trang web của giáo sư E. Keogh [10]. Hình dạng và chiều dài của các chuỗi thời gian được thể hiện trong các hình 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8.



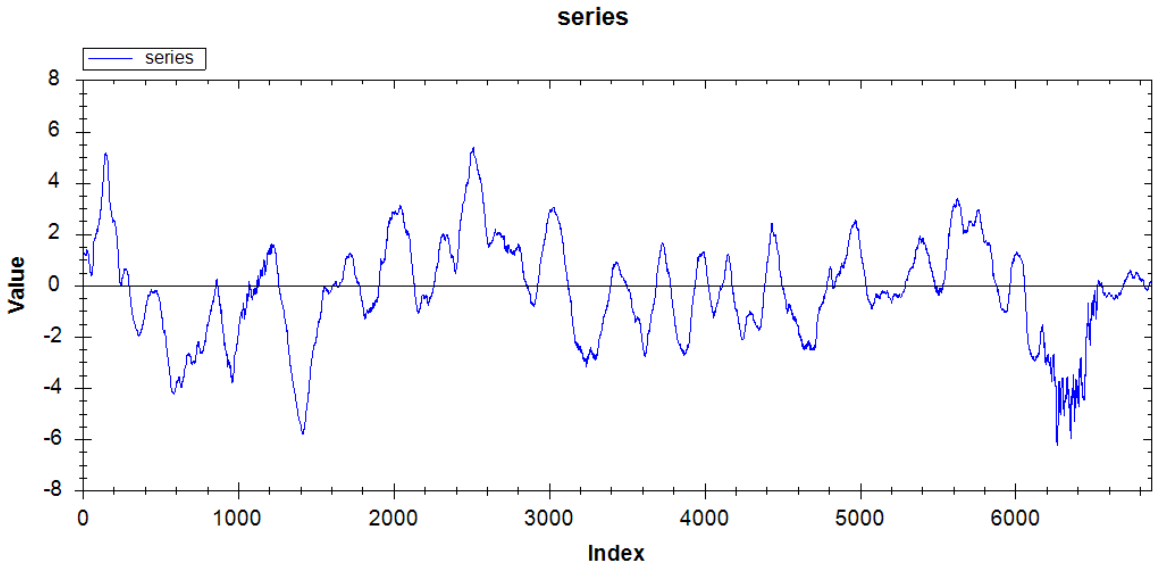
Hình 5.1. Chuỗi thời gian ECG 108, chiều dài 17500 điểm



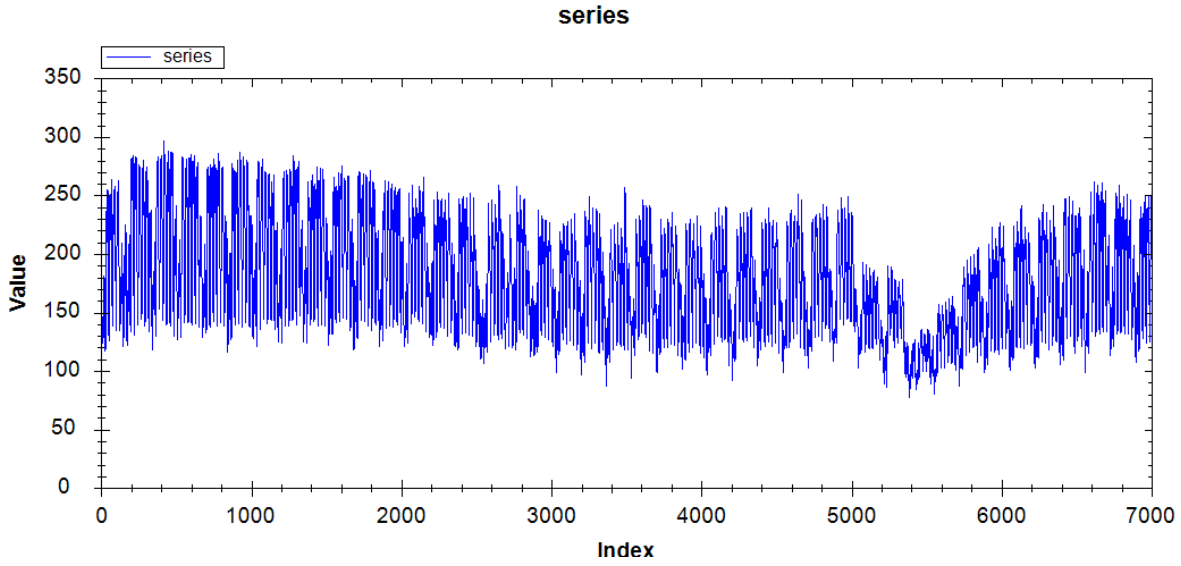
Hình 5.2. Chuỗi thời gian ECG 308, chiều dài 1300 điểm



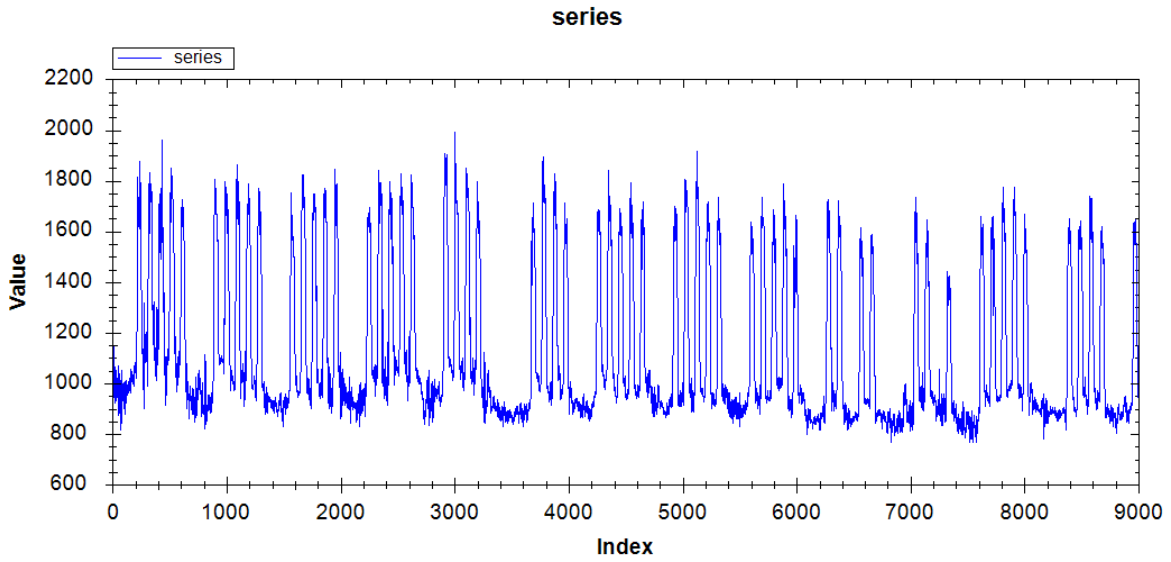
Hình 5.3. Chuỗi thời gian ERP, chiều dài 5000 điểm



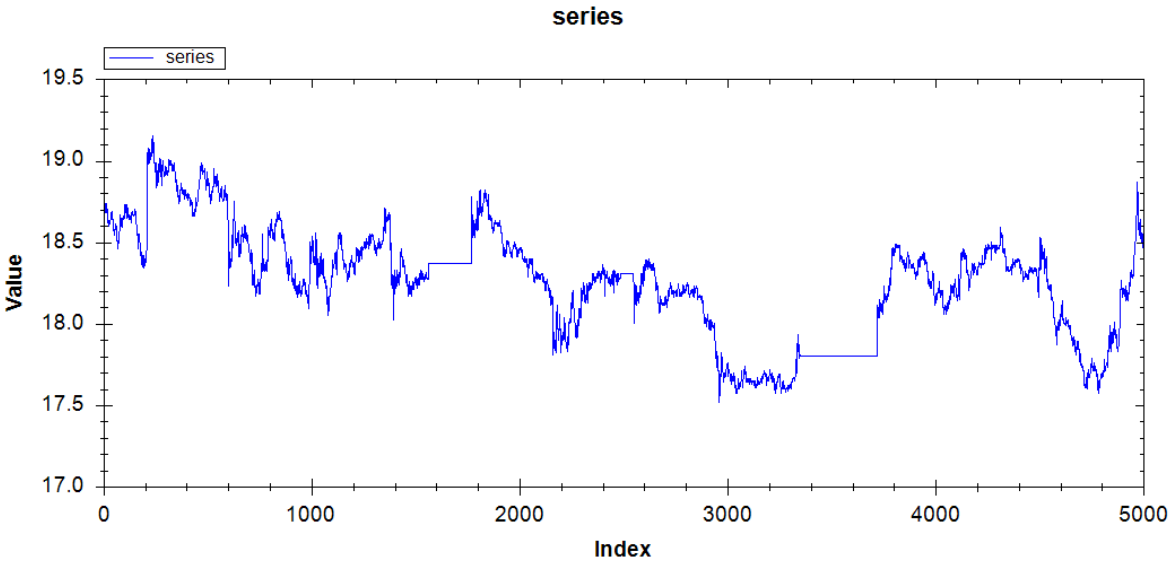
Hình 5.4. Chuỗi thời gian Memory, chiều dài 6875 điểm



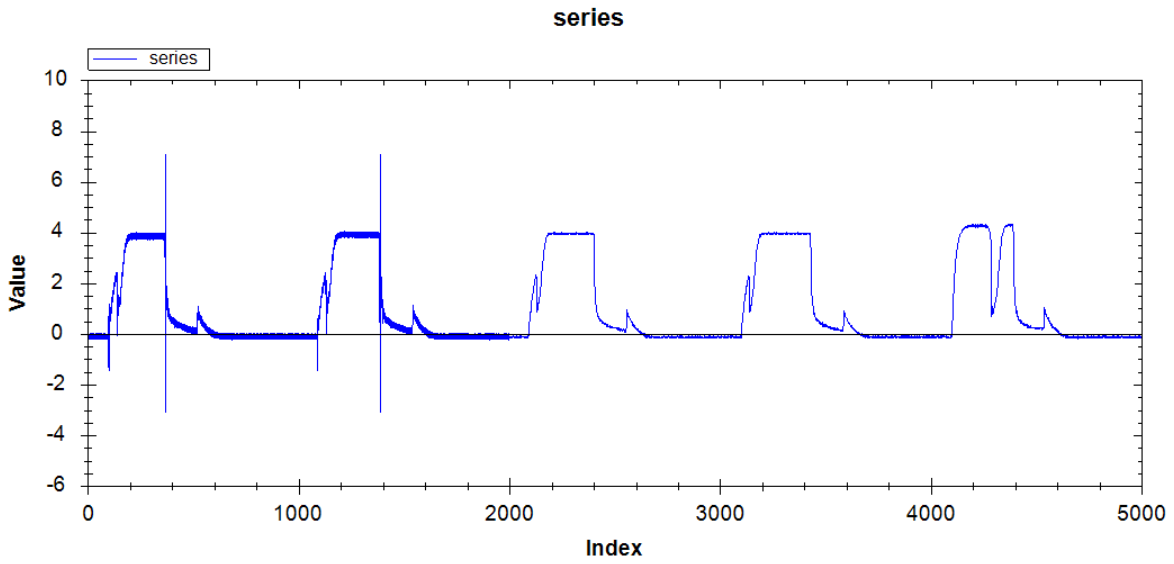
Hình 5.5. Chuỗi thời gian Power Demand In Italy, chiều dài 7000 điểm



Hình 5.6. Chuỗi thời gian Dutch Power Demand, chiều dài 9000 điểm



Hình 5.7. Chuỗi thời gian Stock20, chiều dài 5000 điểm



Hình 5.8. Chuỗi thời gian TEK16, chiều dài 5000 điểm

5.2. Thực nghiệm đánh giá tính hiệu quả của các giải thuật

Để đánh giá khả năng xác định các chuỗi con bất thường của giải thuật đề xuất, chúng tôi tiến hành so sánh kết quả chạy thực nghiệm của giải thuật với kết quả của giải thuật HOT SAX. Chúng tôi chọn giải thuật HOT SAX vì giải thuật HOT SAX duyệt qua từng điểm của chuỗi thời gian để tìm chuỗi con bất thường nên ít có khả năng bỏ sót. Hơn nữa kết quả của giải thuật HOT SAX giống với kết

quả của giải thuật vét cạn nhưng thời gian thực thi nhanh hơn [6], điều này giúp việc chạy thực nghiệm để so sánh nhanh hơn mà vẫn không làm giảm tính chính xác.

Do giải thuật HOT SAX tìm chuỗi con bất thường nhất dựa trên sự so sánh khoảng cách lân cận gần nhất của các chuỗi con, chúng tôi luôn chọn tham số k bằng 1 để tính các hệ số bất thường. Chuỗi con bất thường có độ bất thường lớn nhất sẽ được so sánh với kết quả của HOT SAX. Chúng tôi quan tâm đến độ lệch của điểm bắt đầu giữa hai chuỗi con bất thường khi so sánh giải thuật đề xuất với giải thuật HOT SAX. Độ lệch này được tính theo công thức sau:

$$d = \frac{|p - q|}{l} \quad (5.2.1)$$

Với d là độ lệch, p là điểm bắt đầu của chuỗi con bất thường có hệ số bất thường lớn nhất tìm thấy bởi giải thuật đề xuất, q là điểm bắt đầu của chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX. Giá trị d càng nhỏ thì kết quả của giải thuật đề xuất càng khớp với giải thuật HOT SAX.

Bảng 5.2 và 5.2 dưới đây là bảng giải thích cho các ký hiệu mà chúng tôi sẽ sử dụng trong việc trình bày các kết quả thực nghiệm bên dưới

VL_QR	Giải thuật tìm kiếm chuỗi con bất thường trong dữ liệu chuỗi thời gian bằng phương pháp đánh giá hệ số bất thường sử dụng phương pháp phân đoạn dùng đa thức bậc hai để xấp xỉ các chuỗi con
VL_EP	Giải thuật tìm kiếm chuỗi con bất thường trong dữ liệu chuỗi thời gian bằng phương pháp đánh giá hệ số bất thường sử dụng phương pháp phân đoạn dùng các điểm cực trị quan trọng
HOT SAX	Giải thuật HOT SAX

Bảng 5.1. Bảng ký hiệu các giải thuật

$\varepsilon_1, \varepsilon_2, r$	Các tham số trong phần phân đoạn của giải thuật VL_QR, ý nghĩa của các tham số xem ở mục 3.2.4 và 4.1
R, MIN_LENGH, r	Các tham số trong phần phân đoạn của giải thuật VL_EP, ý nghĩa của các tham số xem ở mục 4.1 và 4.2
a	Ngưỡng bất thường, các chuỗi con có hệ số bất thường lớn hơn giá trị này sẽ được xem là chuỗi con bất thường
n, PAA_LENGTH	Tham số của giải thuật HOT SAX, ý nghĩa của các tham số xem ở mục 3.2.1

Bảng 5.2. Bảng ký hiệu các tham số

Dưới đây là kết quả chi tiết cho từng chuỗi dữ liệu mẫu

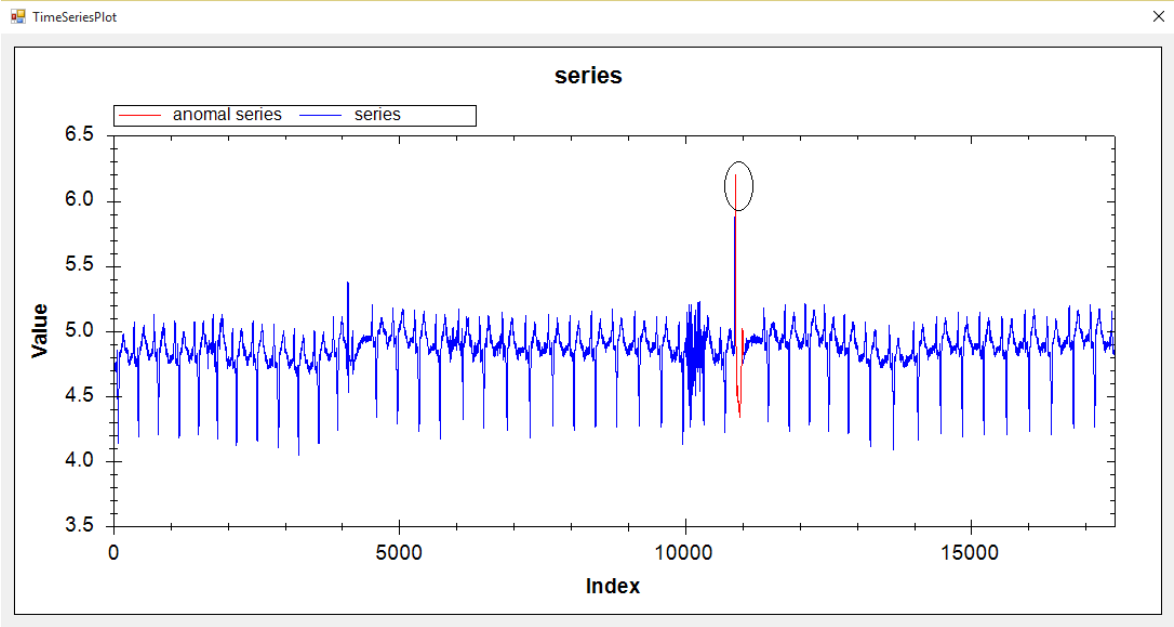
5.2.1. Kết quả thực nghiệm của chuỗi thời gian ECG 108

Kết quả thực nghiệm của chuỗi thời gian ECG 108 được thể hiện trong bảng 5.3

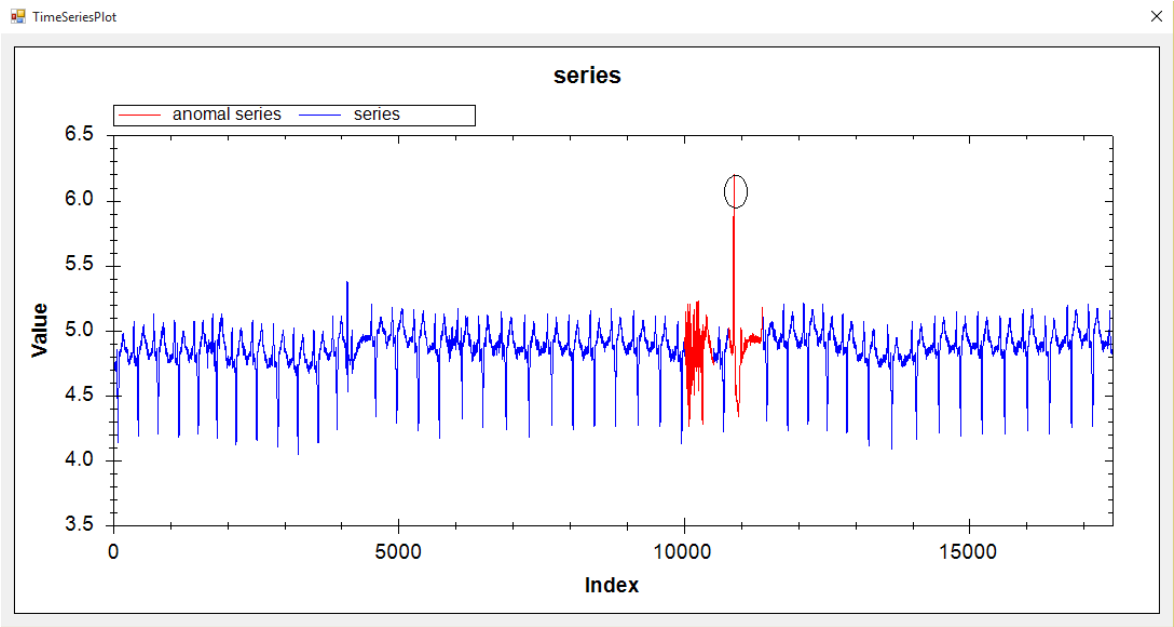
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1 = 5.0, ε_2 = 0.3, *a*= 3.5, *r* = 0.1	10875	*d*=1.5%	10
VL_EP	*R*=1.04,*MIN_LENGH*=50, *a*=4, *r* = 0.1	10792	*d*=12.3%	47
HOT SAX	*n* = 600, *PAA_LENGTH*=60	10866	*d* = 0%	468

Bảng 5.3. Kết quả thực nghiệm trên chuỗi dữ liệu ECG 108.

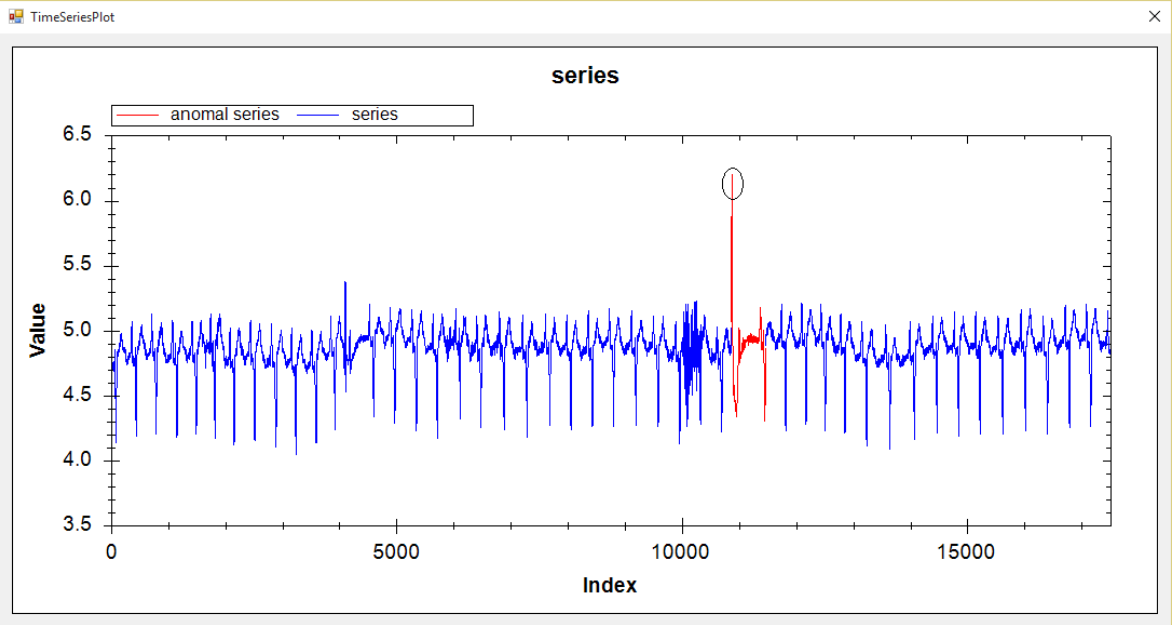
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.9, 5.10 và 5.11. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.9. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu ECG 108



Hình 5.10. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu ECG 108



Hình 5.11. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu ECG 108

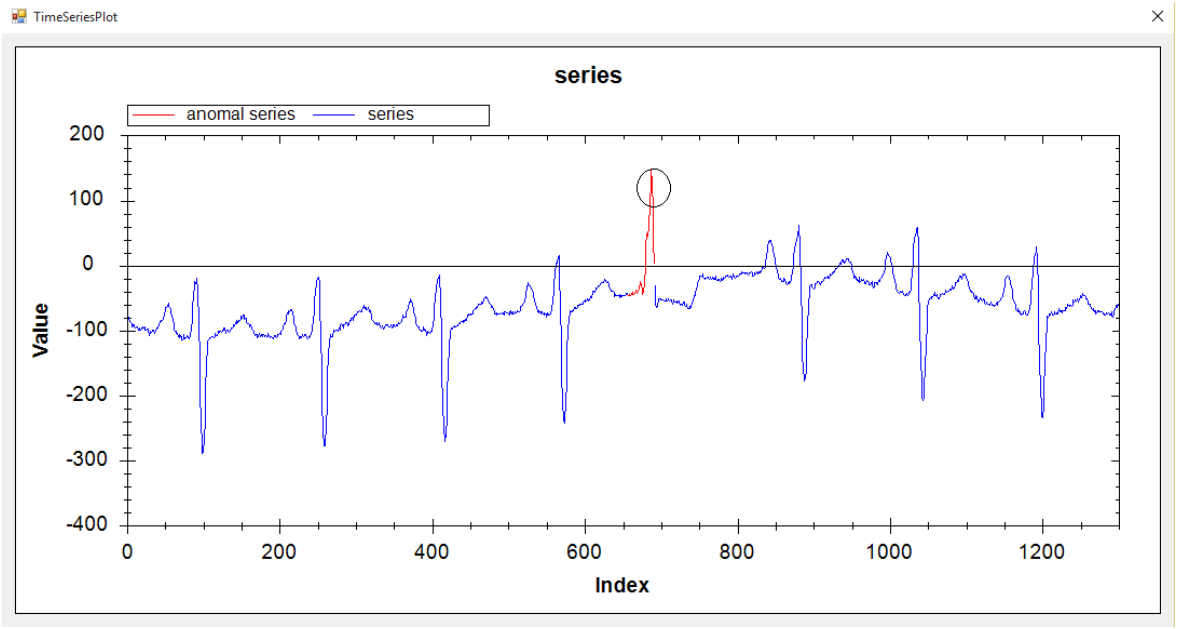
5.2.2. Kết quả thực nghiệm của chuỗi thời gian ECG 308

Kết quả thực nghiệm của chuỗi thời gian ECG 308 được thể hiện trong bảng 5.3

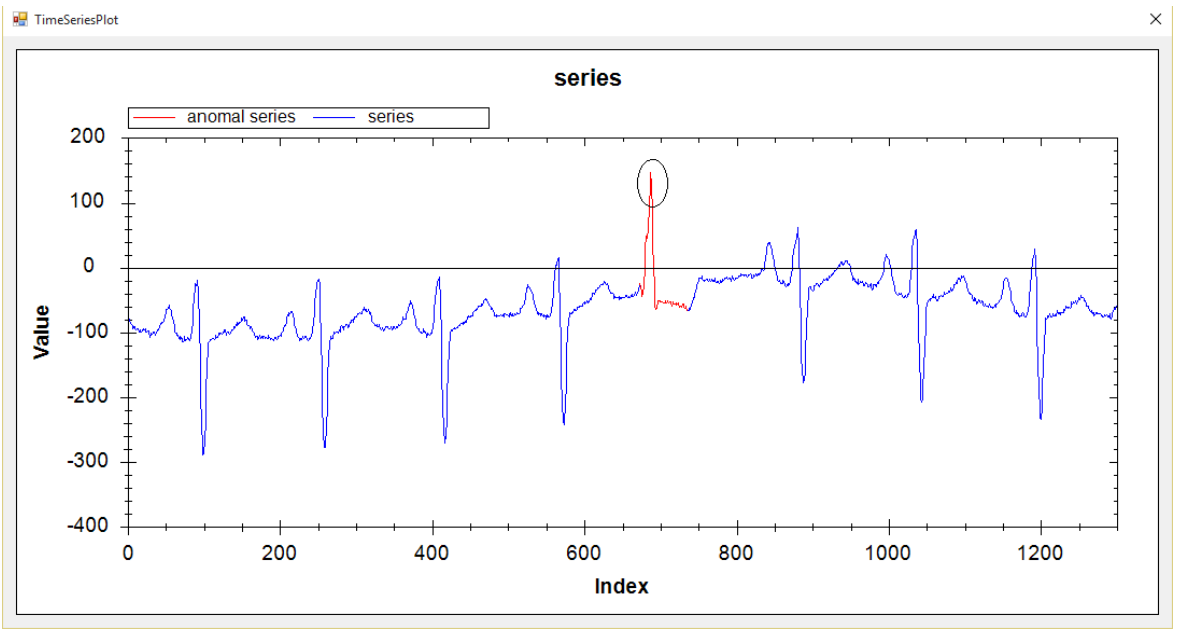
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=51000, ε_2=50, a=2.2, r=0.1	656	d=33%	0.31
VL_EP	R=1.03, *MIN_LENGH*=10, a=2.5, r = 0.25	673	d=5%	2
HOT SAX	n=60, *PAA_LENGTH*=6	676	d=0%	13

Bảng 5.4. Kết quả thực nghiệm trên chuỗi dữ liệu ECG 308

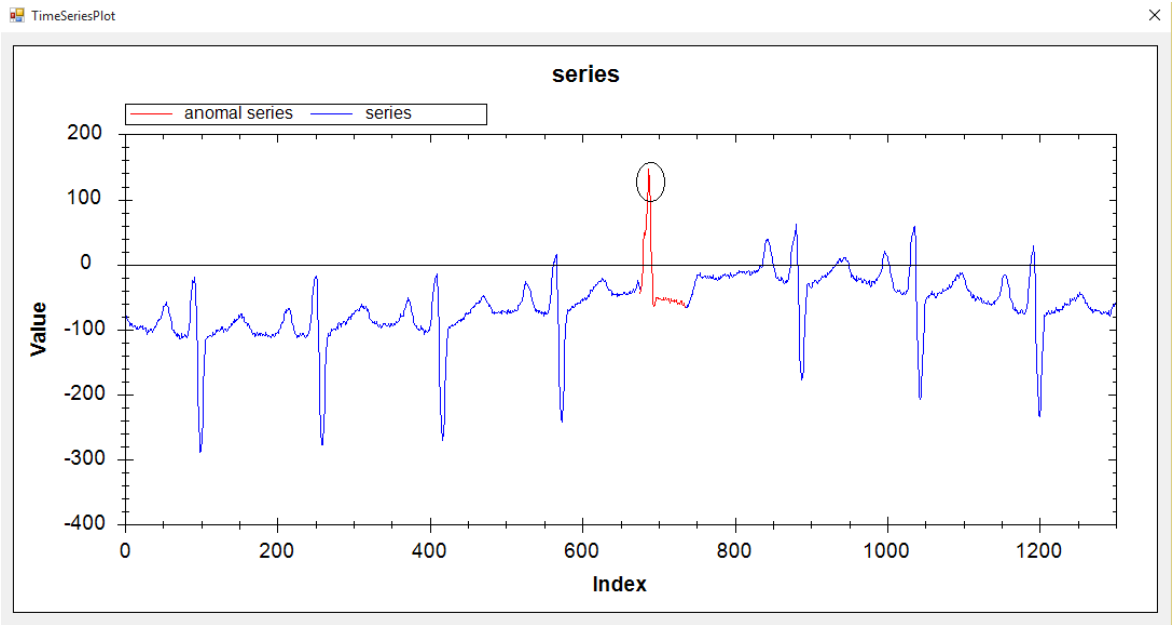
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.12, 5.13 và 5.14. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.12 Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu ECG 308.



Hình 5.13. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu ECG 308.



Hình 5.14. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu ECG 308.

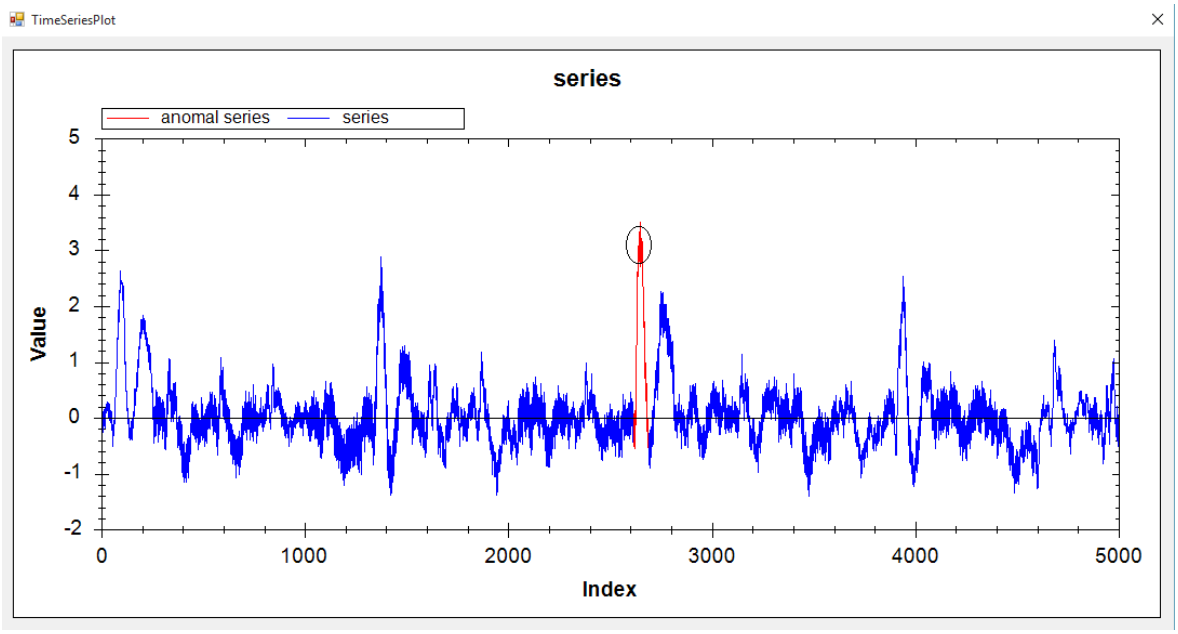
5.2.3. Kết quả thực nghiệm của chuỗi thời gian ERP

Kết quả thực nghiệm của chuỗi thời gian ERP được thể hiện trong bảng 5.4

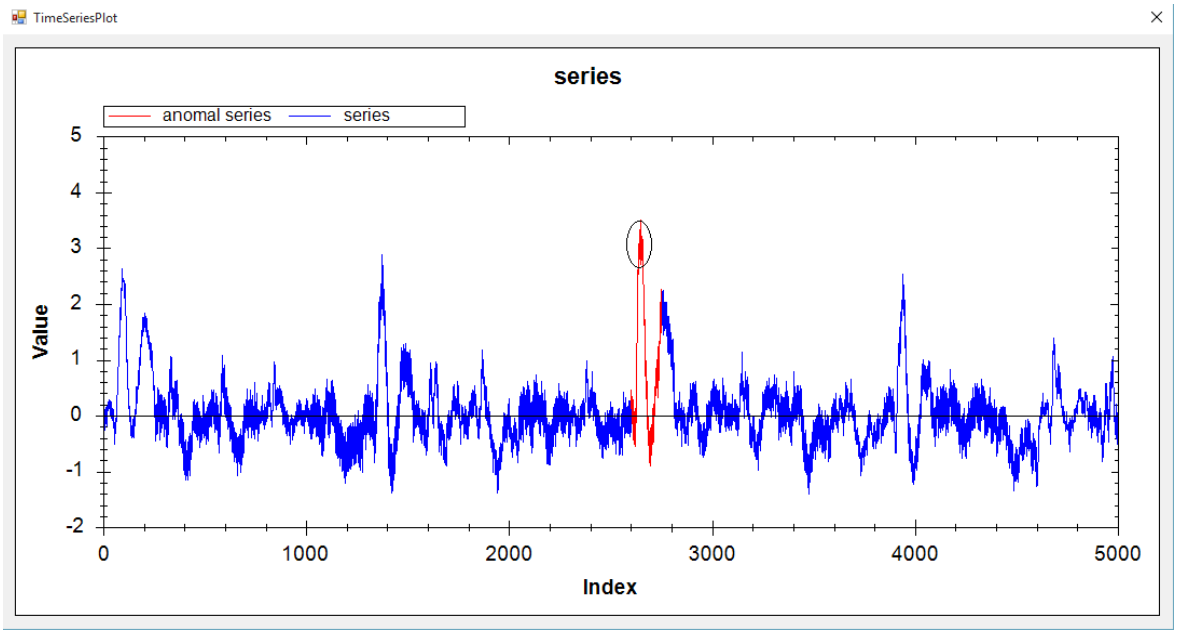
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=3.0, ε_2=1.0, a=3.0, r = 0.15	2617	d=32%	7
VL_EP	R = 1.42, *MIN_LENGH* = 10, a = 3.5, r = 0.1	2602	d=17%	14
HOT SAX	n=100, *PAA_LENGTH*=10	2585	d=0%	60

Bảng 5.5. Kết quả thực nghiệm trên chuỗi dữ liệu ERP

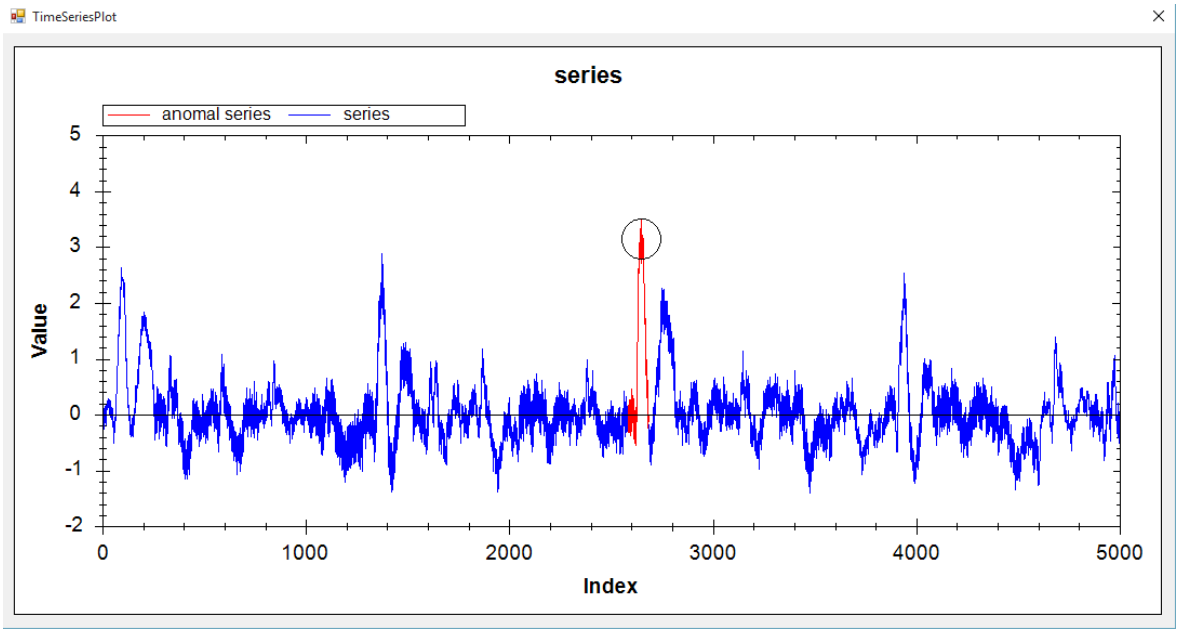
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.15, 5.16 và 5.17. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.15. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu ERP.



Hình 5.16. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu ERP.



Hình 5.17. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu ERP.

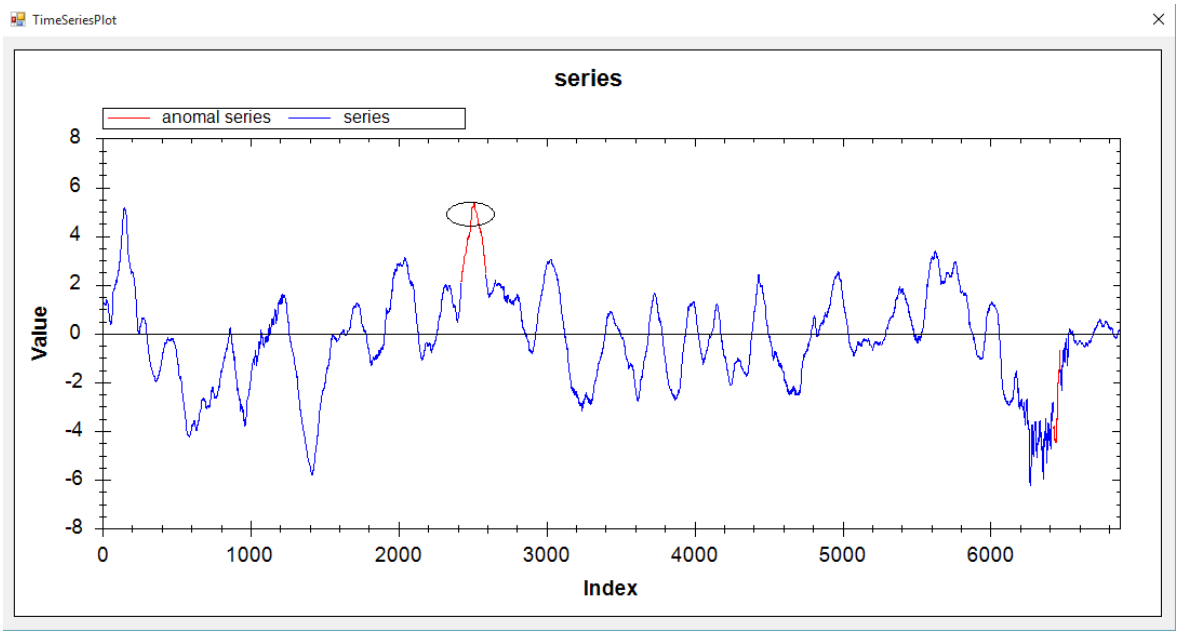
5.2.4. Kết quả thực nghiệm của chuỗi thời gian Memory

Kết quả thực nghiệm của chuỗi thời gian Memory được thể hiện trong bảng 5.5

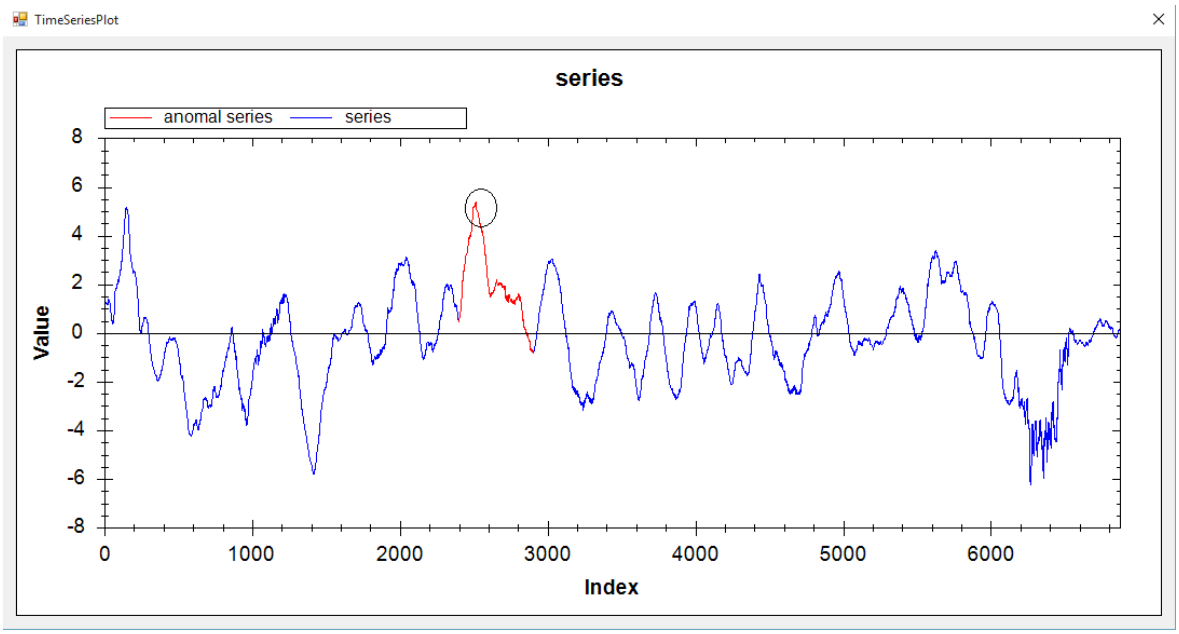
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=8.0, ε_2=0.1, a=2.2, r = 0.1	2425	d=2%	4
VL_EP	R=1.1, *MIN_LENGH*=40, a=1.6, r = 0.1	2399	d=11%	6
HOT SAX	n=200, *PAA_LENGTH*=20	2421	d=0%	79

Bảng 5.6. Kết quả thực nghiệm trên chuỗi dữ liệu ERP

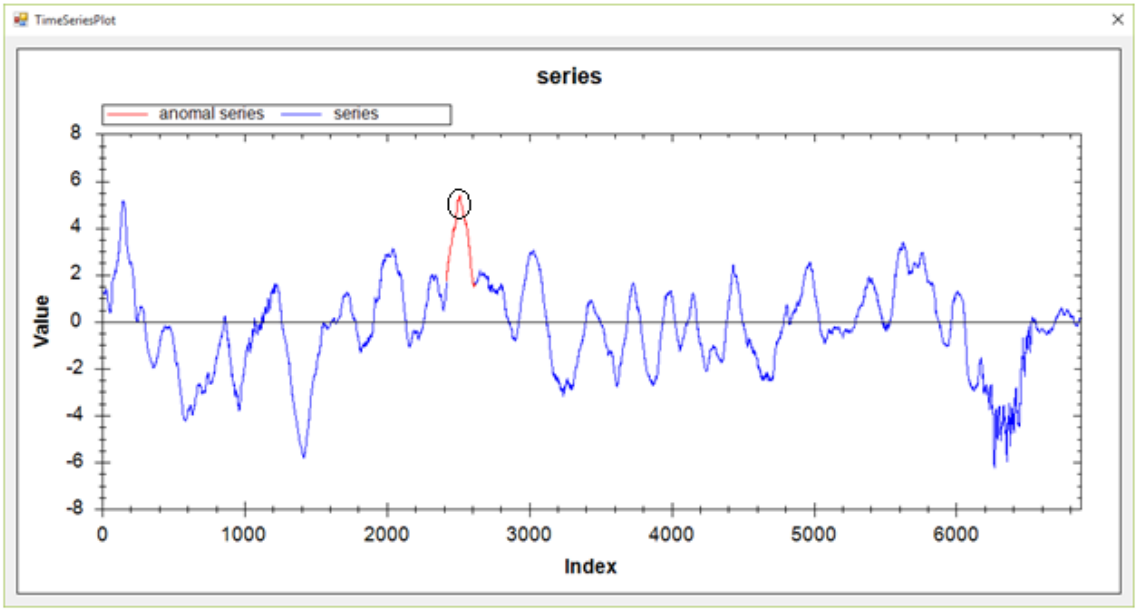
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.18, 5.19 và 5.20. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.18. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu Memory.



Hình 5.19. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu Memory.



Hình 5.20. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu Memory.

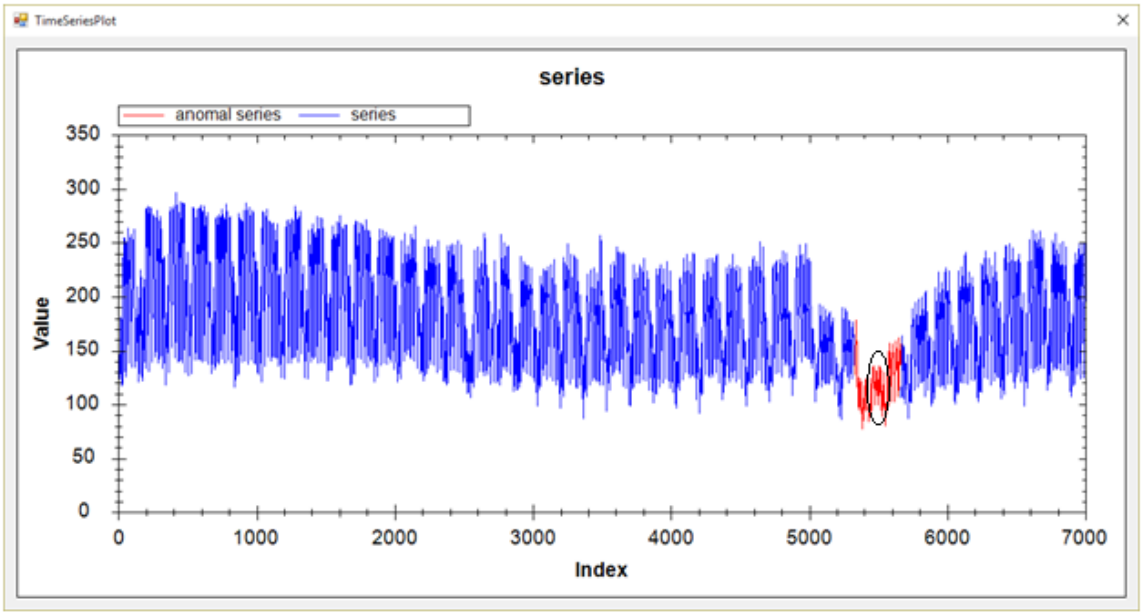
5.2.5. Kết quả thực nghiệm của chuỗi thời gian Power Demand In Italy

Kết quả thực nghiệm của chuỗi thời gian Power Demand In Italy được thể hiện trong bảng 5.6

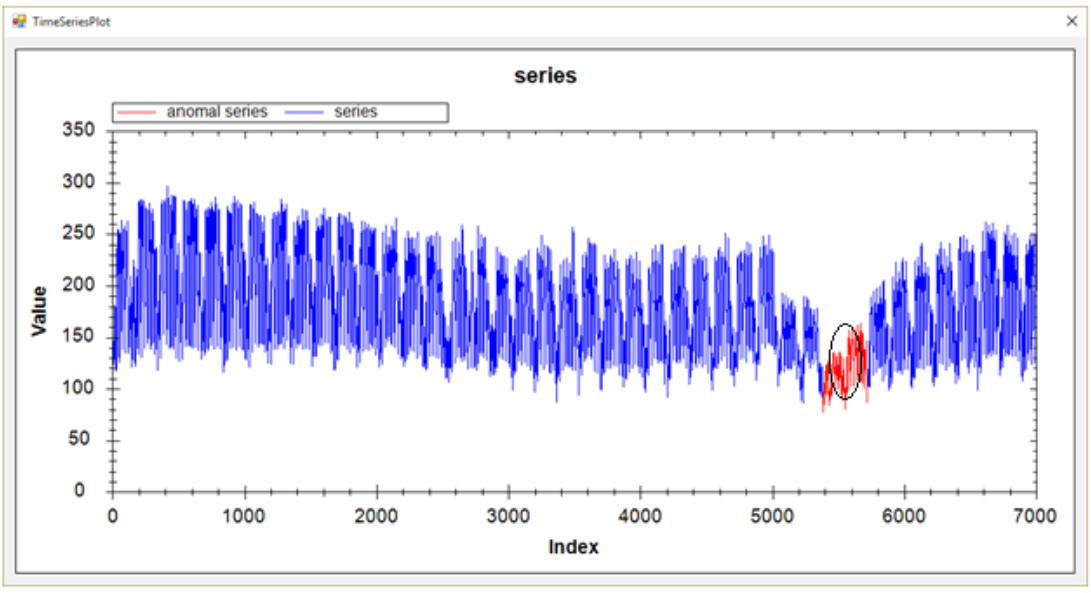
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1= 100000,ε_2= 100, a=2.5, r = 0.1	5331	d=4%	9
VL_EP	R = 1.8, *MIN_LENGH* = 20, a = 3.0, r = 0.1	5383	d=13.33%	11
HOT SAX	n=300, *PAA_LENGTH*=30	5343	d = 0%	102

Bảng 5.7. Kết quả thực nghiệm trên chuỗi dữ liệu Power Demand In Italy

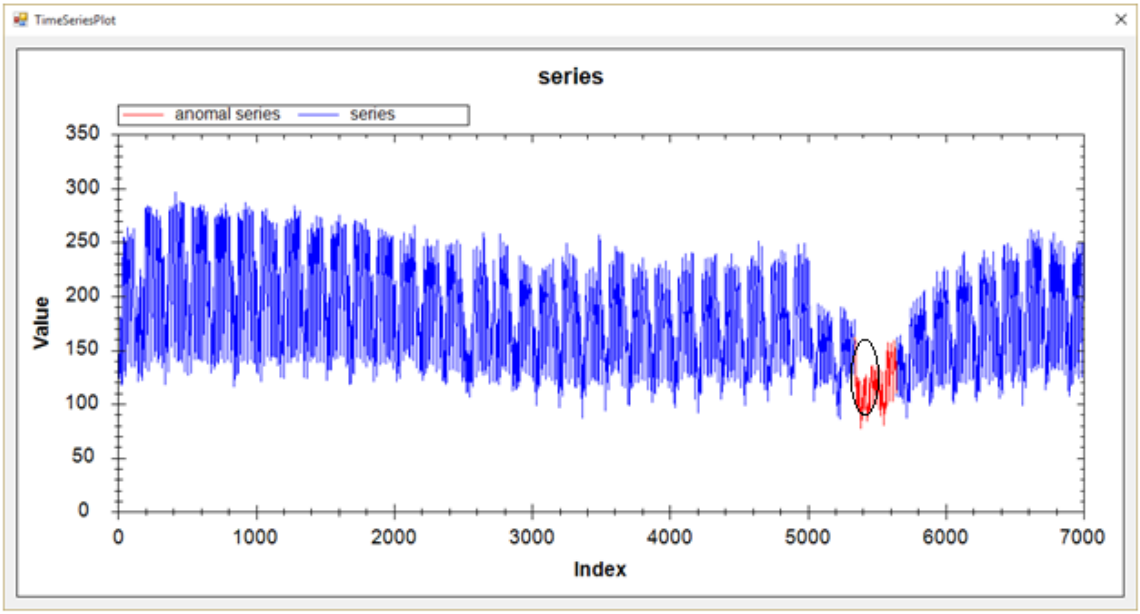
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.21, 5.22 và 5.23. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.21. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu Power Demand In Italy.



Hình 5.22. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu Power Demand In Italy.



Hình 5.23. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu Power Demand In Italy

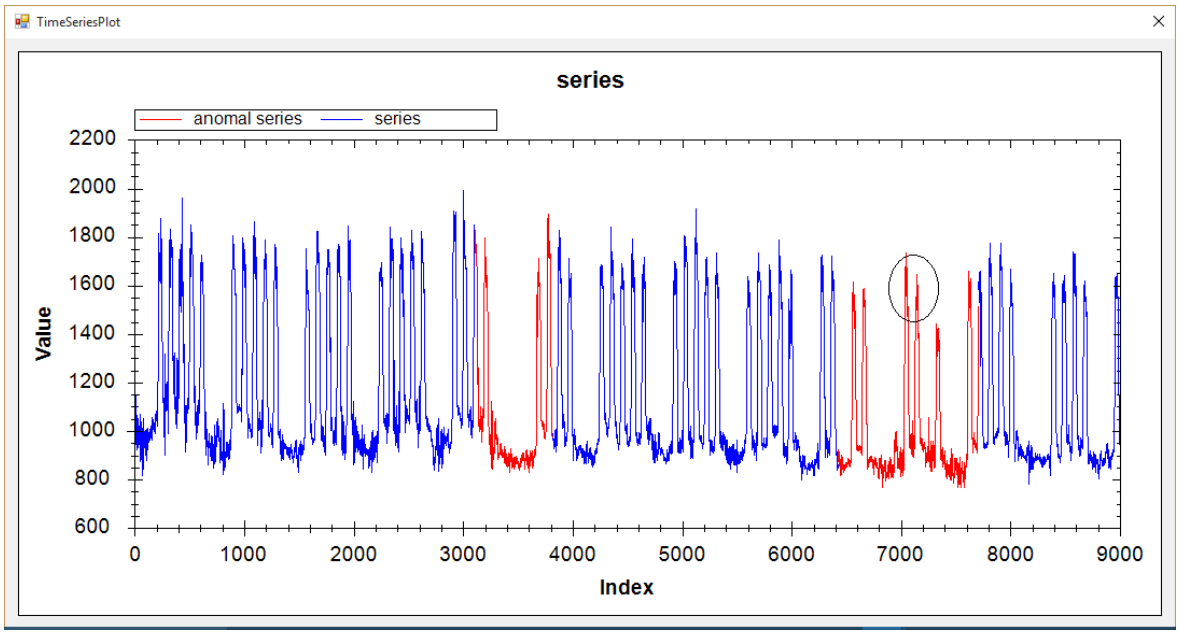
5.2.6. Kết quả thực nghiệm của chuỗi thời gian Dutch Power Demand

Kết quả thực nghiệm của chuỗi thời gian Dutch Power Demand được thể hiện trong bảng 5.7

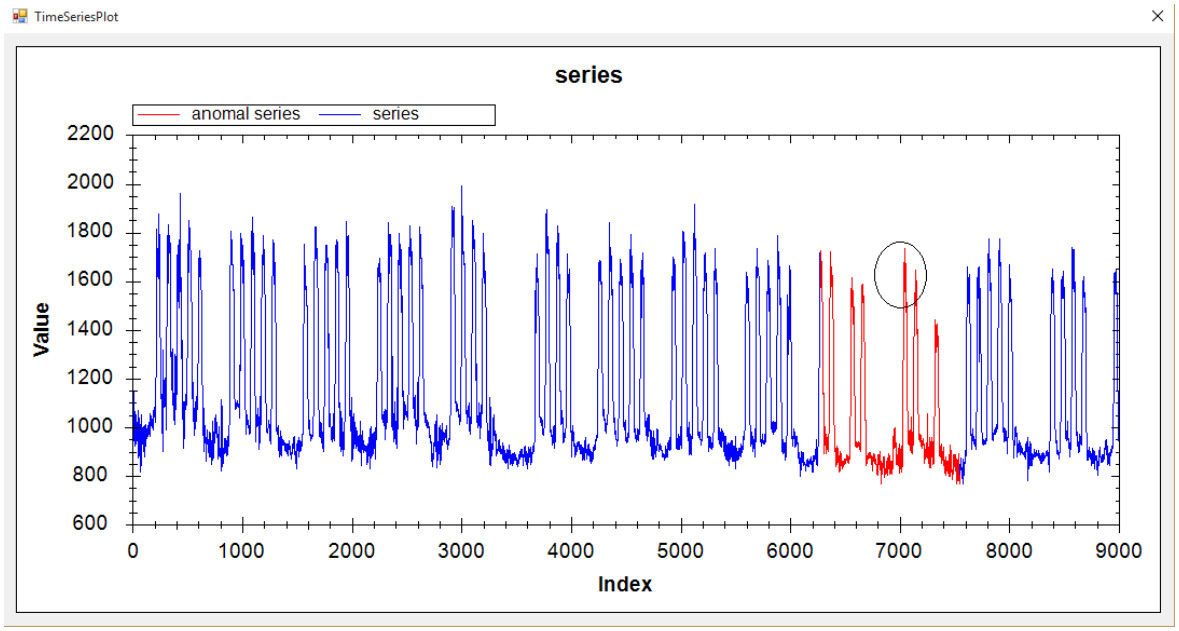
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=31000000, ε_2=500, a=1.6, r=0.2	6424	d=1.3%	1
VL_EP	R=1.7, *MIN_LENGH*=180, a=1.33, r=0.25	6277	d=11%	9
HOT SAX	n=1200,*PAA_LENGTH*=120	6466	d=0%	302

Bảng 5.8. Kết quả thực nghiệm trên chuỗi dữ liệu Dutch Power Demand

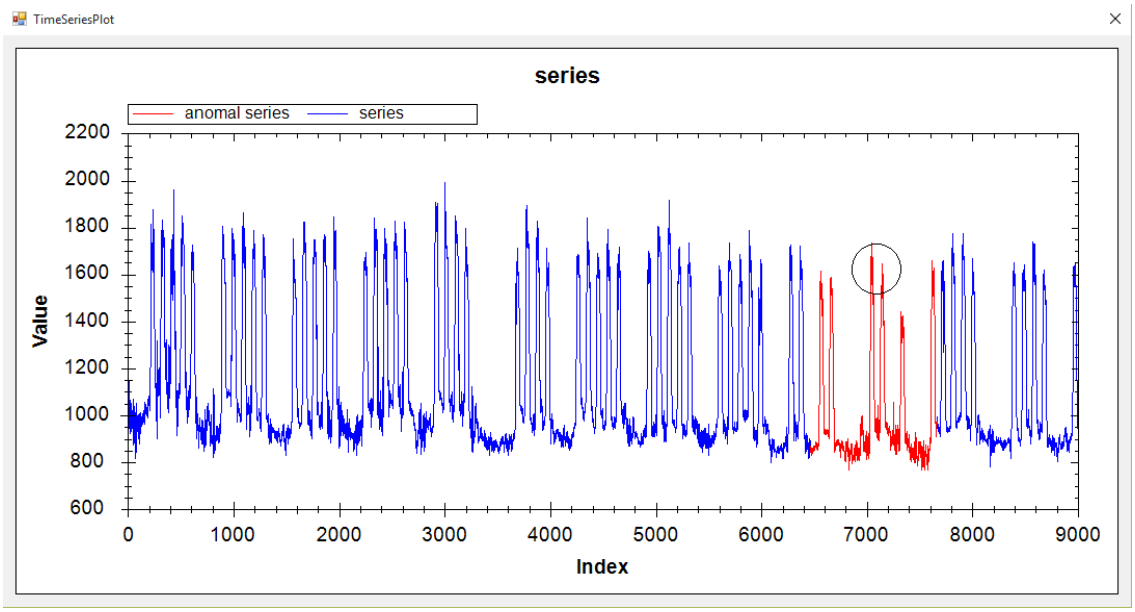
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.24, 5.25 và 5.26. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.24. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu Dutch Power Demand.



Hình 5.25. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu Dutch Power Demand



Hình 5.26. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu Dutch Power Demand

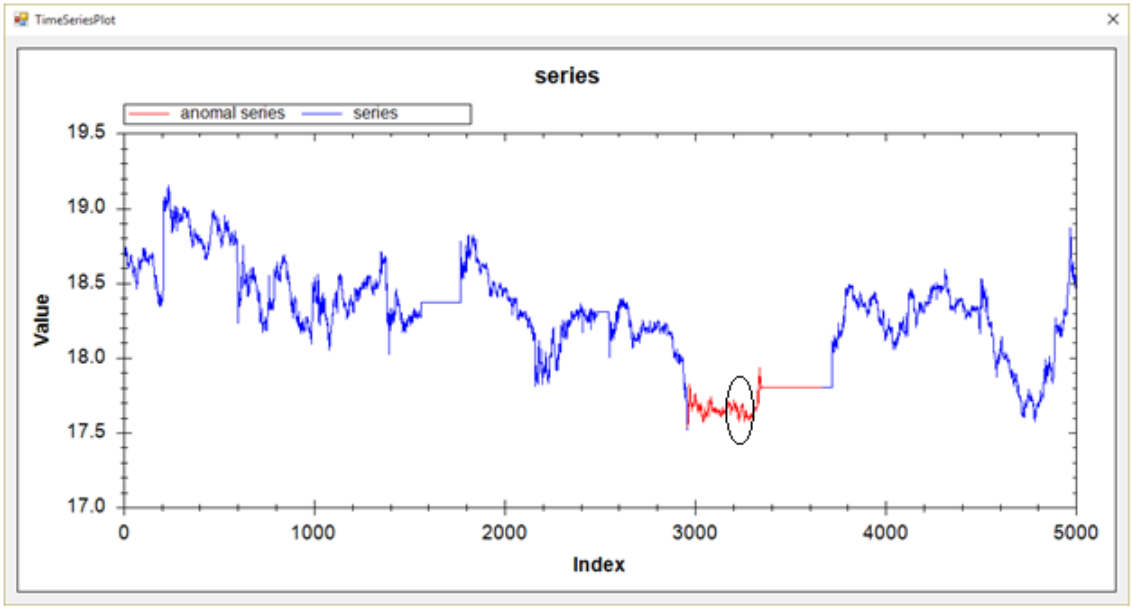
5.2.7. Kết quả thực nghiệm của chuỗi thời gian Stock20

Kết quả thực nghiệm của chuỗi thời gian Stock20 được thể hiện trong bảng 5.8

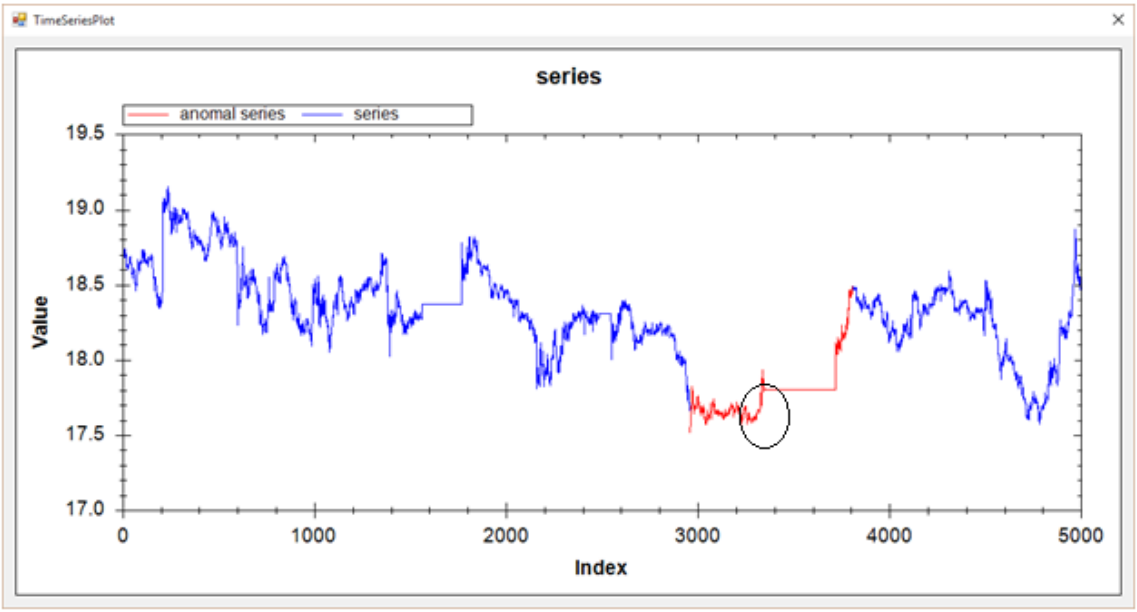
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=2.0, ε_2=0.1, a=3.5, r = 0.1	2961	d=1.7%	1
VL_EP	R=1.01, *MIN_LENGH*=200, a=1.5, r = 0.1	2960	d=1.6%	1
HOT SAX	n=700, *PAA_LENGTH*=70	2949	d=0%	94

Bảng 5.9. Kết quả thực nghiệm trên chuỗi dữ liệu Stock20

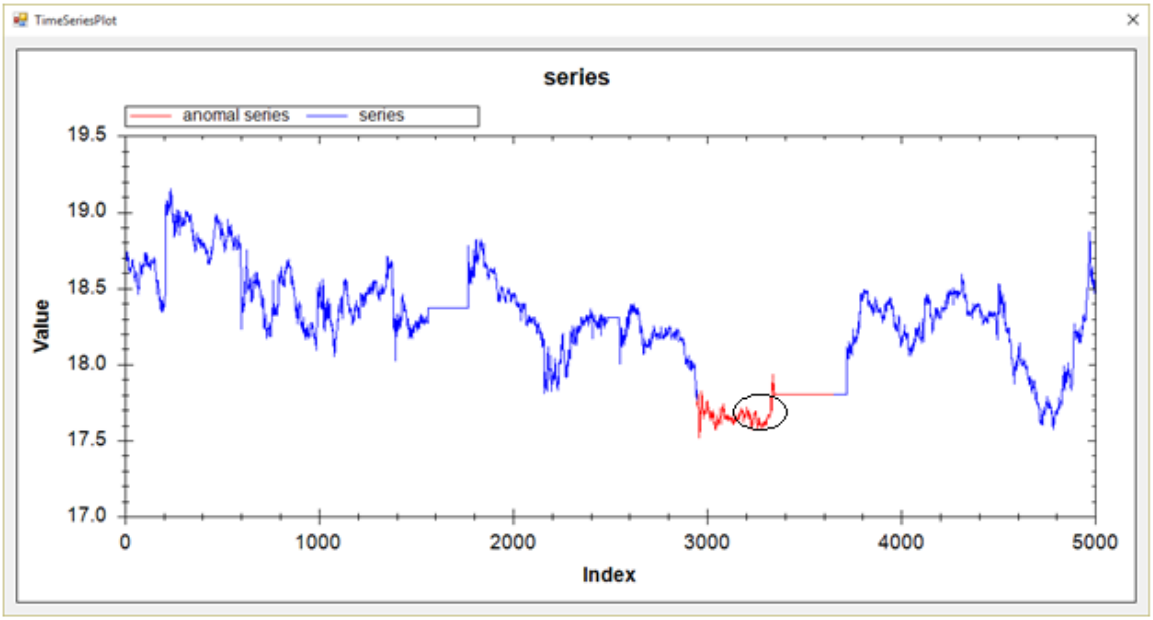
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.27, 5.28 và 5.29. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.27. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu Stock20



Hình 5.28. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu Stock20



Hình 5.29. . Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu Stock20

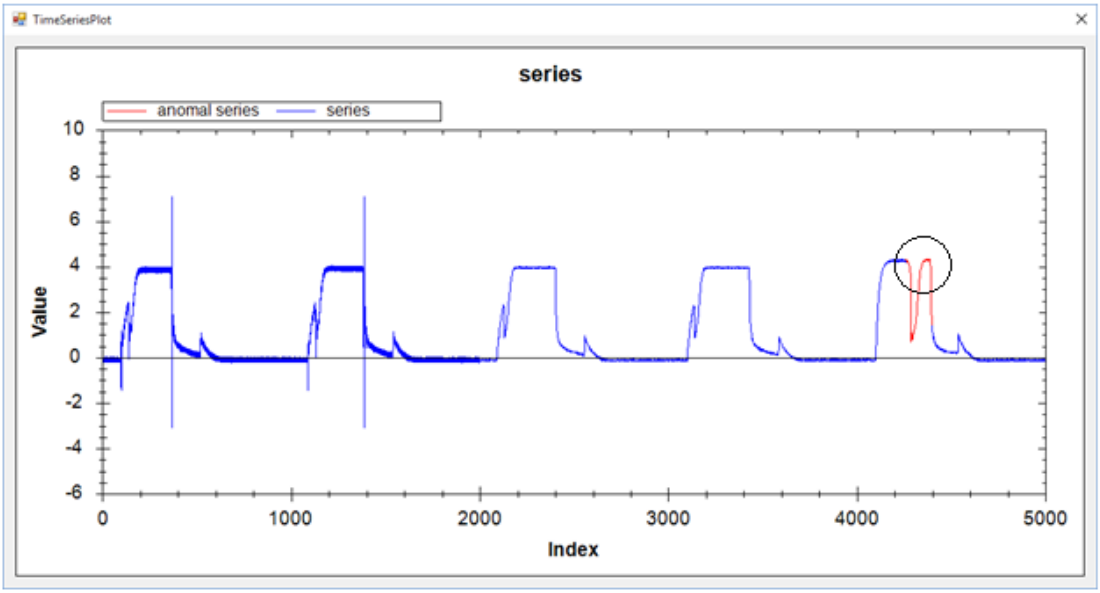
5.2.8. Kết quả thực nghiệm của chuỗi thời gian TEK16

Kết quả thực nghiệm của chuỗi thời gian TEK16 được thể hiện trong bảng 5.9

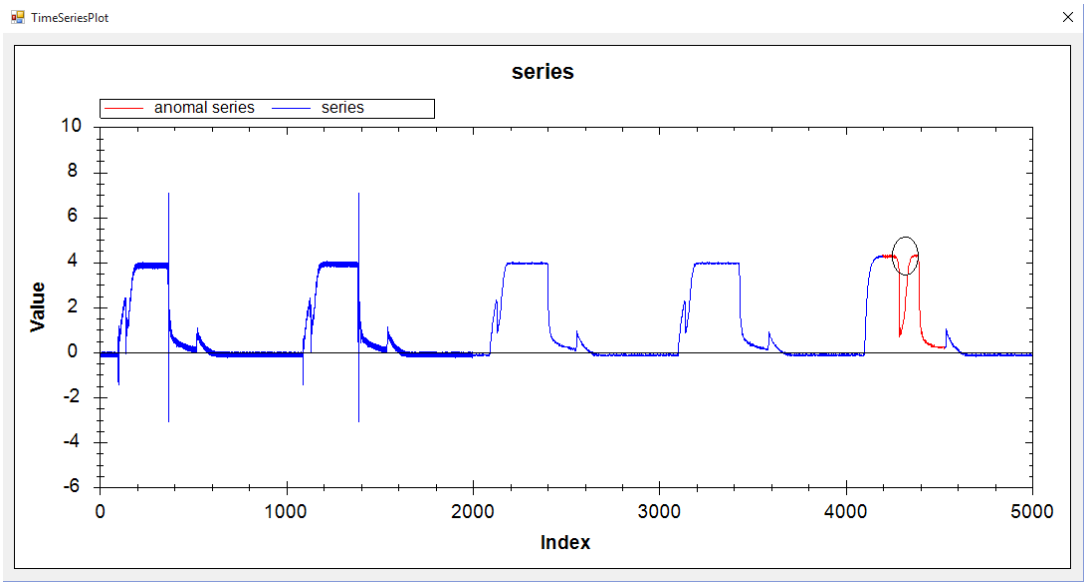
Giải thuật	Tham số	Điểm bắt đầu của chuỗi con bất thường nhất	Độ lệch	Thời gian chạy (giây)
VL_QR	ε_1=14.5, ε_2=0.1, a=2.0, r = 0.1	4262	d=24%	1
VL_EP	R=1.02, *MIN_LENGH*=50,a=1.5, r= 0.1	4197	d=27%	1
HOT SAX	n=128, *PAA_LENGTH*=8	4231	d=0%	52

Bảng 5.10. Kết quả thực nghiệm trên chuỗi dữ liệu TEK16

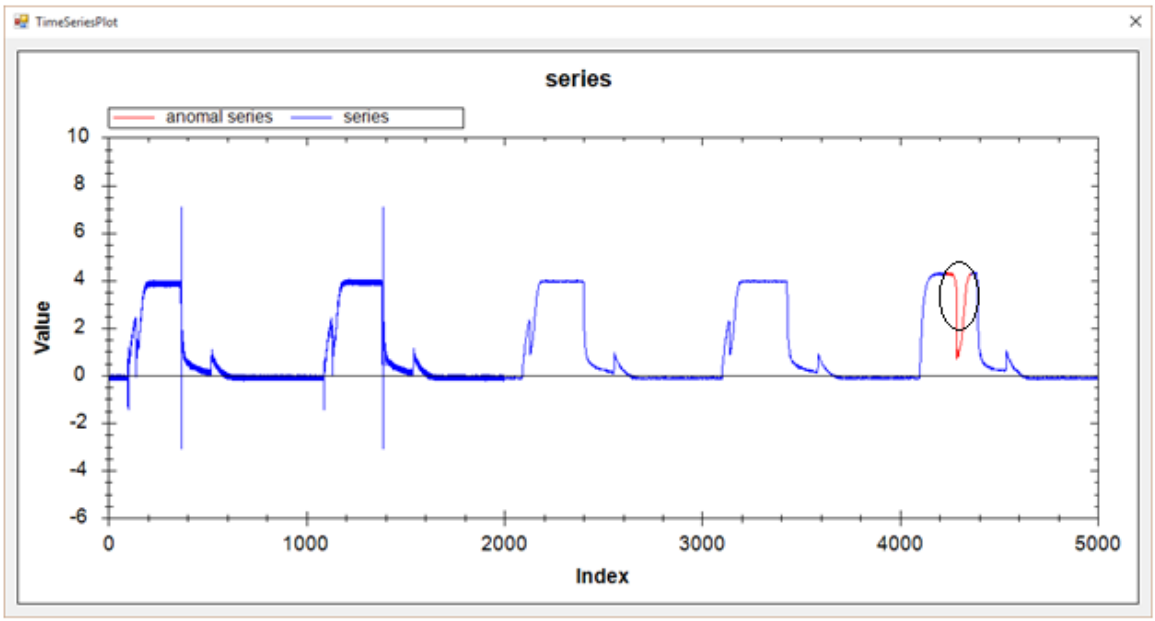
Các chuỗi con bất thường được tìm thấy bởi 3 giải thuật được thể hiện lần lượt trên các hình 5.30, 5.31 và 5.32. Chuỗi con có dấu tròn là chuỗi con bất thường nhất.



Hình 5.30. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_QR trên bộ dữ liệu TEK16.



Hình 5.31. Các chuỗi con bất thường tìm thấy bởi giải thuật VL_EP trên bộ dữ liệu TEK16.



Hình 5.32. Các chuỗi con bất thường tìm thấy bởi giải thuật HOT SAX trên bộ dữ liệu TEK16

5.2.9. Nhận xét

a) Nhận xét về độ chính xác

Các kết quả thực nghiệm trên cho thấy giải thuật VL_QR và VL_EP tìm ra các chuỗi con bất thường khá khớp với giải thuật HOT SAX.

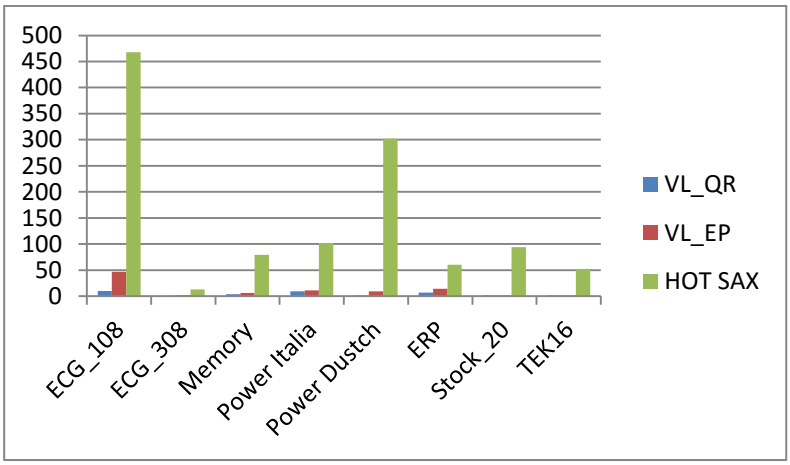
Độ lệch của giải thuật VL_QR chạy thực nghiệm trên các bộ dữ liệu ECG 108, Memory, Power Demand In Italy, Dutch Power Demand, Stock20, chỉ khoảng từ 1.3% đến 4%. Đối với bộ dữ liệu ECG 308, ERP và TEK16 độ lệch của giải thuật VL_QR lần lượt là 33%, 32% và 24%. Tuy nhiên so sánh các hình 5.12 với 5.14, 5.15 với 5.17, 5.30 với 5.32 ta thấy các độ lệch này vẫn ở mức chấp nhận được và và chuỗi con bất thường tìm được bởi giải thuật VL_QR khá khớp với kết quả của giải thuật HOT SAX.

Độ lệch của giải thuật VL_EP chạy thực nghiệm trên các chuỗi dữ liệu ECG 108, ECG 308, ERP, Memory, Power Demand In Italy, Dutch Power Demand, Stock20, chỉ khoảng từ 5% đến 17%. Đối với bộ dữ liệu TEK16 độ lệch của giải

thuật VL_EP là 27%. Tuy nhiên so sánh hình 5.31. và 5.32, ta thấy chuỗi con bất thường tìm được vẫn khớp với kết quả của giải thuật HOT SAX.

b) Nhận xét về thời gian thực thi và việc ước lượng tham số

Hai giải thuật VL_QR và VL_EP có thời gian thực thi nhanh hơn nhiều lần so với giải thuật HOT SAX trên tất cả các bộ dữ liệu. Hinh 5.33 là biểu đồ so sánh thời gian chạy của 3 giải thuật. Ta thấy thời gian thực thi của giải thuật HOT SAX lớn hơn từ 8 đến 94 lần so với giải thuật VL_QR và lớn hơn từ 6 đến 94 lần so với giải thuật VL_EP. Thời gian thực thi của giải thuật VL_QR nhỏ hơn thời gian thực thi của giải thuật VL_EP.



Hình 5.33. Thời gian thực thi của VL_QR, VL_EP và HOT SAX

Bảng 5.11 mô tả các giá trị tham số ờ phần phân đoạn của giải thuật VL_QR và giải thuật VL_EP. Dựa vào bảng này, ta thấy việc ước lượng các tham số phần phân đoạn cho giải thuật VL_EP dễ hơn so với giải thuật VL_QR do có mức độ thay đổi ít hơn. Tham số ε_1 của giải thuật VL_QR có giá trị thay đổi từ 2 đến 31000000, tỉ lệ biến đổi là 1550000000%. Tham số ε_2 của giải thuật VL_QR có giá trị thay đổi từ 0.1 đến 500, tỉ lệ biến đổi là 500000%. Trong khi đó, tham số R của giải thuật VL_EP có giá trị thay đổi từ 1.01 đến 1.8, tỉ lệ thay đổi 178%. Tham số *MIN_LENGTH* của giải thuật VLWEP thay đổi từ 10 đến 200, tỉ lệ biến đổi 2000%.

Tham số *r* đều nhận giá trị nhỏ vào khoảng 0.1 đến 0.25 cho tất cả các bộ dữ liệu. Điều này chứng tỏ công thức (4.1.3) có thể thay thế cho công thức (3.2.1) để giảm số lần tính khoảng cách của phương pháp chiều dài biến đổi của M. Leng và các cộng sự [22] mà không làm giảm độ chính xác của giải thuật.

Chuỗi dữ liệu	Tham số của VLWQ		Tham số của VLWEP	
	ε_1	ε_2	*R*	*MIN_LENGH*
ECG 108	5.0	0.3	1.04	50
ECG 308	51000	50	1.03	10
ERP	3	1	1.42	10
Memory	8	0.1	1.1	40
Power Demand In Italy	100000	100	1.8	20
Dutch Power Demand	31000000	500	1.7	180
Stock20	2	0.1	1.01	200
TEK16	14.5	0.1	1.02	50

Bảng 5.11. Bảng so sách các tham số của hai giải thuật VL_QR và VL_EP

5.3. Thực nghiệm đánh giá sự cải thiện tốc độ thực thi của giải thuật khi áp dụng phương pháp tính khoảng cách Euclid kết hợp với phép vị tự.

Trong mục 4.2, chúng tôi đã chứng minh độ phức tạp tính toán của phương pháp tính khoảng cách bằng công thức Euclid và pháp biến hình vị tự có độ phức tạp tính toán tuyến tính trong khi phương pháp tính khoảng cách bằng độ đo xoắn thời gian động có độ phức tạp tính toán bình phương. Do đó nếu áp dụng phương pháp tính khoảng cách bằng công thức Euclid và pháp biến hình vị tự thì thời gian chạy của giải thuật tìm chuỗi con bất thường dựa trên hệ số bất thường do M.Length và các cộng sự [22] sẽ giảm. Trong mục này chúng tôi tiến hành chạy giải thuật VL_QR với hai phương pháp tính khoảng cách nói trên trên 8 chuỗi dữ liệu mẫu để

khẳng định kết luận này bằng thực nghiệm. Do thời gian chạy của giải thuật VL_QR khi áp dụng phương pháp tính khoảng cách xoắn thời gian động rất chậm nên chúng tôi chỉ trích một phần trong các chuỗi thời gian mẫu để thực nghiệm. Giải thuật xoắn thời gian động được hiện thực *đa luồng* (multithreading) theo cách của tác giả Văn Thế Huy [32]. Kết quả thực nghiệm được thể hiện trên bảng 5.12.

Chuỗi dữ liệu	Thời gian chạy của VLWQ với công thức tính khoảng cách Euclid và phép vị tự (giây)	Thời gian chạy của VLWQ với phương pháp tính khoảng cách xoắn thời gian động (giây)
ECG 108 (1000 điểm)	1	14
ECG 308 (1300 điểm)	1	11
ERP (1000 điểm)	1	5
Memory (1000 điểm)	1	4
Power Demand In Italy (3000 điểm)	6	200
Dutch Power Demand (3000 điểm)	1	24
Stock20 (3000 điểm)	2	194
TEK16 (2000 điểm)	1	81

Bảng 5.12. Bảng kết quả thực nghiệm so sánh tốc độ thực thi khi áp dụng phươn pháp tính khoảng cách Euclid và phép biến hình vị tự.

Dựa vào bảng 5.12 ta thấy tốc độ thực thi đã tăng đáng kể khi áp dụng phương pháp tính khoảng cách bằng công thức Euclid và phép biến hình vị tự thay

cho phương pháp tính khoảng cách bằng độ đo xoắn thời gian động. Thời gian thực thi tăng lên từ 4 (bộ dữ liệu Memory) đến 97 lần (bộ dữ liệu Stock20).

Chương 6
KẾT LUẬN

Chương này trình bày những đóng góp, những hạn chế và hướng phát triển của luận văn

6.1. Đóng góp của luận văn

Luận văn đã giải quyết bài toán tìm các chuỗi con bất thường có độ dài khác nhau mà không cần biết trước chiều dài của các chuỗi con bất thường bằng cách phân đoạn chuỗi thời gian và tính hệ số bất thường cho các chuỗi con. Các chuỗi con có hệ số bất thường lớn hơn một ngưỡng được người dùng quy định chính là các chuỗi con bất thường. Mô hình tìm chuỗi con bất thường này đã được M. Leng và các cộng sự đưa ra trong [22]. Tuy nhiên, phương pháp của M. Leng và các cộng sự gặp hai hạn chế: độ phức tạp tính toán cao và khóa ước lượng các tham số trong quá trình phân đoạn. Luận văn đã khắc phục hai hạn chế này bằng ba đóng góp sau:

- Thứ nhất đề ra phương pháp tính khoảng cách bằng công thức Euclid và phép biến hình vị tự thay cho phương pháp tính khoảng cách xoắn thời gian động. Điều này giúp giảm độ phức tạp tính toán của cộng thức tính khoảng cách từ đó tăng tốc độ thực thi cho giải thuật, làm cho giải thuật có khả năng áp dụng được cho các chuỗi thời gian có kích thước lớn.
- Thứ hai đưa thêm tham số r và thay thế công thức (3.2.1) bằng công thức (4.1.3). Điều này giúp làm giảm số lần tính khoảng cách khi xây dựng ma trận khoảng cách cho các chuỗi con và tăng tốc độ thực thi cho giải thuật.
- Thứ ba đề xuất thêm phương pháp phân đoạn bằng các điểm cực trị quan trọng. Việc ước lượng các tham số cho phương pháp phân đoạn này đơn giản hơn việc ước lượng các tham số cho phương pháp phân đoạn bằng cách dùng đa thức bậc hai để xấp xỉ các chuỗi con như đề xuất của M. Leng và các cộng sự [22]. Điều này giúp cho giải thuật dễ sử dụng hơn.

6.2. Hạn chế của luận văn

Việc ước lượng các tham số cho phương pháp phân đoạn dựa trên các điểm cực trị quan trọng tuy đơn giản hơn so với việc ước lượng tham số cho phương pháp phận đoạn bằng cách dùng đa thức bậc 2 xấp xỉ các chuỗi con nhưng vẫn là một công việc tốn nhiều công sức. Công việc này chủ yếu dựa vào kinh nghiệm và hiểu biết của người sử dụng về bộ dữ liệu cần xử lý. Việc phân đoạn chuỗi thời gian bằng các điểm cực trị quan trọng dựa trên giả định rằng các chuỗi con bất thường có điểm bắt đầu tại những điểm cực trị này. Do đó việc xác định các điểm cực trị quan trọng không phù hợp sẽ dẫn đến việc xác định sai hay bỏ sót các chuỗi con bất thường.

Việc đánh giá xem một chuỗi con có phải là chuỗi con bất thường phải dựa vào tham số *a*. Chọn một giá trị phù hợp cho tham số này phụ thuộc vào kiến thức của người sử dụng về bộ dữ liệu cần xử lý. Giá trị hợp lý của *a* thay đổi theo từng bộ dữ liệu và có thể thay đổi ngay trong một bộ dữ liệu khi có các điểm dữ liệu mới xuất hiện. Điều này làm cho mô hình khó sử dụng được trong thực tế.

6.3. Hướng phát triển của luận văn.

Luận văn có thể phát triển dựa trên hướng giải quyết các hạn chế nêu ở mục trên. Phát triển một thủ tục có thể ước lượng tự động các tham số hoặc tìm ra một số chỉ dẫn cho việc ước lượng các tham số dễ dàng hơn và có thể áp dụng cho nhiều bộ dữ liệu. Phát triển một phương pháp tự động giúp hỗ trợ cho người sử dụng xác định tham số *a* cho các bộ dữ liệu khác nhau.

TÀI LIỆU THAM KHẢO

[1] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira. *A Method Based on RBF-DDA Neural Networks for Improving Novelty Detection in Time Series*. Neural Networks Proceedings of the International Joint Conference, Volume 3, pp. 2123-2128, 2004.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] E. Keogh, S. Lonardi, B. Y. Chiu. *Finding Surprising Patterns in a Time Series Database in Linear Time and Space*. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and Data mining, pp. 550-556, 2002.

[10] E. Keogh. *www.cs.ucr.edu/~eamonn/discords/*, 2015.

[11] F. Itakura. *Minimum prediction residual principle applied to speech recognition*. IEEE Transactions on Acoustics, Speech and Signal Processing, Volume 23, pp. 67 – 72, 1975.

[12] G. Li, O. Braysy, L. Jiang, Z. Wu, Y.Wang. *Finding time series discord based on bit representation clustering.* Journal Knowledge-Based System, Volume 54, December, pp. 243-254, 2013.

[13] H. Sakoe, S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing, Volume 26, pp. 43–49, 1978.

[14] Hồ Chí Thanh. *Phát hiện chuỗi con bất thường trên dữ liệu chuỗi thời gian*. Luận văn thạc sĩ, Học Viện Công Nghệ Bưu Chính Viễn Thông, 2015

[15] Huỳnh Nguyễn Tín. *Nhận diện motif trên dữ liệu chuỗi thời gian dựa vào điểm cực trị quan trọng*. Luận văn thạc sĩ, Đại Học Bách Khoa TP Hồ Chí Minh, 2012.

[16] 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.

[17] J. Ma, S. Perkins. *Times series Novelty Detection Using One-class Support Vector Machines*. Neural Networks Proceedings of the International Joint Conference, Volume 3, pp. 1741-1745, 2003.

[18] K.B. Pratt, E. Fink. *Search for Patterns in Compressed Time Series*. International Journal of Image and Graphics, Volume 2, No. 1, pp. 89-106, 2002.

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

[20] L. Wei, E. Keogh, X. Xi. *Efficiently Finding Unsual Shapes in Large Image Database*. Data Mining and Knowledge Discovery, Volume 17, Issuse 3, pp. 343-376, 2008.

[21] M. C. Chuah, F. Fu. *ECG Anomaly Detection via Time Series Analysis*. Fontiers of High Performance Computing and Networking ISPA, Volume 4743, pp. 123-135, 2007.

[22] 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.

[23] M.C. Dani, F.X Follois, M. Nadif, C. Freixo. *Adaptive Threshold for Anomaly Detection Using Time Series Segmentation*. Neural Information Processing, Volume 9491of the series Lecture Notes in Computer Science, pp. 82-89, 2015

[24] N.H. Kha, D.T. Anh. *From Cluster-Based Outlier Detection to Tine Series Discord Discovery*. Trends and Aplications in Knowledge Discovery and Data Mining, Volume 9441 of the series Lecture Notes in Computer Science, pp. 16-28, 2015.

[25] 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, 2015.

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

[27] S. Salvador, P. Chan, J. Brodie. *Learning States and Rules for Time Series Anomaly Detection*. Applied Intelligence, Volume 23, Issuse 3, pp. 241-255, 2005.

[28] 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.

[29] 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.

[30] 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.

[31] V. Chandola, A. Banerjee, V. Kumar. *Anomaly Detection : A Survey*. ACM Computing Surveys, Volume 41, Issuse 3, Article No. 15, 2009.

[32] Văn Thế Huy. *Gom cụm dữ liệu chuỗi thời gian với độ đo xoắn thời gian động dựa vào một kỹ thuật xấp xỉ*. Luận văn thạc sĩ, Đại Học Bách Khoa TP Hồ Chí Minh, 2015.