TẠP CHÍ KHOA HỌC
TRƯỜNG ĐẠI HỌC SƯ PHẠM TP HỒ CHÍ MINH
Tập 19, Số 3 (2022): 435-448
Website: http://journal.hcmue.edu.vn
ISSN: 2734-9918

HO CHI MINH CITY UNIVERSITY OF EDUCATION
JOURNAL OF SCIENCE
Vol. 19, No. 3 (2022): 435-448
https://doi.org/10.54607/hcmue.js.19.3.3280(2022)

Bài báo nghiên cứu

PHÁT HIỆN MOTIF BẰNG THUẬT TOÁN SCRIMP++ CẢI TIẾN

***Nguyễn Thành Sơn*, Trần Thị Dung**

Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh, Việt Nam

Tác giả liên hệ: Nguyễn Thành Sơn – Email: sonnt@hcmute.edu.vn

Ngày nhận bài: 27-9-2021; ngày nhận bài sửa: 14-3-2022; ngày duyệt đăng: 18-3-2022

TÓM TẮT

Motif trên chuỗi thời gian là cặp chuỗi con giống nhau nhất trong một chuỗi thời gian hay các cặp chuỗi giống nhau nhất trong một cơ sở dữ liệu chuỗi thời gian. Khám phá motif trên chuỗi thời gian là bài toán quan trọng trong khai phá dữ liệu chuỗi thời gian. Gần đây, một số thuật toán mới đã được giới thiệu cho bài toán khám phá motif dựa vào vector chứa khoảng cách giữa một chuỗi con với lân cận gần nhất của nó. Các thuật toán này sử dụng kĩ thuật kết hợp việc chuẩn hóa chuỗi thời gian vào trong công thức tính độ đo khoảng cách Euclid khi tính toán ma trận khoảng cách. Phương pháp tiêu biểu cho cách tiếp cận này là thuật toán Scrimp++. Bài báo này giới thiệu một phiên bản cải tiến của thuật toán Scrimp++ cho bài toán khám phá motif nhằm cải thiện thời gian thực thi của thuật toán. Kết quả thực nghiệm cho thấy thuật toán đề xuất thực hiện tốt hơn thuật toán gốc về mặt thời gian nhưng vẫn đảm bảo về độ chính xác.

Từ khóa: ma trận khoảng cách; khám phá motif; chuỗi thời gian; thuật toán Scrimp++; motif trên chuỗi thời gian

1. Giới thiệu

Một chuỗi thời gian là một dãy các số thực được ghi nhận tại những khoảng thời gian bằng nhau. Dữ liệu chuỗi thời gian được sử dụng trong rất nhiều lĩnh vực khác nhau. Ngày nay, dữ liệu chuỗi thời gian ngày càng chiếm một tỉ trọng lớn trong dữ liệu được cung cấp trên thế giới.

Motif trên chuỗi thời gian là cặp chuỗi con giống nhau nhất trong một chuỗi thời gian dài hay các cặp chuỗi giống nhau nhất trong một cơ sở dữ liệu chuỗi thời gian (Mueen et al., 2009). Phát hiện motif trên chuỗi thời gian là một bài toán quan trọng trong khai phá dữ liệu chuỗi thời gian và nhận được nhiều sự quan tâm nghiên cứu trong cộng đồng nghiên cứu về lĩnh vực này.

Từ khi motif trên chuỗi thời gian được Lin và các cộng sự hình thức hóa vào năm 2002 (Lin et al., 2002), rất nhiều phương pháp phát hiện motif đã được giới thiệu (Chiu et al., 2003), (Ferreira et al., 2006), (Yankov et al., 2007), (Castro et al., 2010), (Lin et al.,

Cite this article as: Nguyen Thanh Son, & Tran Thi Dung (2022). Discovering time series motif using the improved Scrimp++ algorithm. *Ho Chi Minh City University of Education Journal of Science, 19*(3), 435-448.

2010), (Mueen et al., 2009), (Mohammad et al., 2014), (Truong et al., 2015), (Nguyen et al., 2016). Cách tiếp cận chung thường được các phương pháp phát hiện motif sử dụng là: (1) dùng một cửa sổ trượt để trích các chuỗi con, (2) chuẩn hóa các chuỗi con và (3) sử dụng thuật toán dựa vào một độ đo tương tự để phát hiện motif. Các thuật toán sử dụng cách tiếp cận này thường phải thực hiện hai lần quét qua chuỗi thời gian: lần quét thứ 1 để chuẩn hóa và lần quét thứ 2 để tính toán khoảng cách.

Gần đây, một số thuật toán đã được đề xuất cho bài toán phát hiện motif trên chuỗi thời gian. Các thuật toán này sử dụng một vector khoảng cách gọi là matrix profile. Vector này chứa khoảng cách của mỗi chuỗi con với chuỗi con lân cận gần nhất với nó trong chuỗi thời gian. Vector khoảng cách được tính toán dựa trên sự kết hợp giữa bước chuẩn hóa và bước tính toán khoảng cách Euclid (Yeh et al., 2016; Zhu et al., 2016; Zhu et al., 2018). Như vậy, các thuật toán sử dụng cách tiếp cận này chỉ cần thực hiện một lần quét qua chuỗi thời gian. Thuật toán tiêu biểu cho cách tiếp cận này là thuật toán Scrimp++.

Bài báo này giới thiệu một phiên bản cải tiến của thuật toán Scrimp++ cho bài toán khám phá motif nhằm cải thiện thời gian thực thi của thuật toán. Kết quả thực nghiệm trên các tập dữ liệu khác nhau cho thấy thuật toán đề xuất thực hiện tốt hơn thuật toán gốc về mặt thời gian nhưng vẫn đảm bảo về độ chính xác.

Phần còn lại của bài báo gồm: phần 2 trình bày về các kiến thức nền tảng và các công trình liên quan; phần 3 trình bày cách tiếp cận của chúng tôi cho bài toán phát hiện motif; phần 4 mô tả đánh giá thực nghiệm trên các tập dữ liệu khác nhau. Cuối cùng, kết luận được trình bày trong phần 5.

2. Nội dung

2.1. Kiến thức nền tảng và các công trình liên quan

2.1.1. Các định nghĩa

Định nghĩa 1. Một chuỗi thời gian T là một dãy các số thực T_i: $T = T_1, T_2, \ldots, T_n$ với n là chiều dài chuỗi thời gian.

Định nghĩa 2. Một chuỗi con $T_{i,m}$ trong chuỗi thời gian T là một tập m giá trị liên tục trong T bắt đầu từ vị trí i. Nghĩa là, $T_{i,m} = T_i, T_{i+1}, \ldots, T_{i+m-1}$ với $1 \le i \le n-m+1$.

Định nghĩa 3. Vector khoảng cách D_i là vector chứa khoảng cách giữa chuỗi con $T_{i,m}$ với mỗi chuỗi con trong chuỗi thời gian T. Nghĩa là, $D_i = [d_{i,1}, d_{i,2}, \ldots, d_{i,n-m+1}]$, với $d_{i,j}$ ($1 \le j \le n-m+1$) là khoảng cách giữa $T_{i,m}$ và $T_{j,m}$.

Tập các vector khoảng cách D_i ($1 \le i \le n-m+1$) của tất cả các chuỗi con trong T sẽ hình thành một ma trận khoảng cách. Ma trận này là ma trận đối xứng qua đường chéo và các giá trị khoảng cách trên đường chéo của ma trận là 0.

Định nghĩa 4. Lân cận gần nhất của một chuỗi con

Cho $T_{i,m}$ và $T_{j,m}$ là hai chuỗi con trong chuỗi thời gian T. $T_{j,m}$ được gọi là lân cận gần nhất của $T_{i,m}$ nếu $\text{dist}(T_{i,m}, T_{j,m}) \le \text{dist}(T_{i,m}, T_{a,m})$, $|i-j| \ge w$, $|i-a| \ge w$ với $w > 0$.

Chú ý là trong định nghĩa 3 có áp đặt một ràng buộc đối với vị trí của các chuỗi con khi so trùng. Đó là các chuỗi con được so sánh phải cách nhau ít nhất *w* vị trí. Điều này giúp loại bỏ các so trùng tầm thường và dist() là hàm tính toán khoảng cách giữa hai chuỗi con.

Định nghĩa 5. Motif trên chuỗi thời gian là một cặp chuỗi con lân cận gần nhất có khoảng cách nhỏ nhất trong tất cả các cặp chuỗi lân cận gần nhất khác trong chuỗi thời gian *T*. Nghĩa là, $\{T_{a,m}, T_{b,m}\}$là motif nếu và chỉ nếu $\text{dist}(T_{a,m}, T_{b,m}) \leq \text{dist}(T_{i,m}, T_{j,m}) \ \forall i, j \in \{1, 2, \ldots, n\text{-}m\text{+}1\}$, $\{T_{a,m}, T_{b,m}\}$và $\{T_{i,m}, T_{j,m}\}$ là các cặp chuỗi con lân cận gần nhất trong chuỗi thời gian *T*.

Định nghĩa 6. Khoảng cách Euclid kết hợp với chuẩn hóa zero-mean

Cho 2 chuỗi $T_{i,m}$ và $T_{j,m}$. Khoảng cách Euclid kết hợp với chuẩn hóa zero-mean giữa 2 chuỗi $T_{i,m}$ và $T_{j,m}$ được tính theo công thức sau (Zhu et al., 2016):

$$dist(T_{i,m}, T_{j,m}) = \sqrt{2m(1 - \frac{T_{i,m}T_{j,m} - m\mu_i\mu_j}{m\sigma_i\sigma_j})} \quad (1)$$

Trong đó, $T_{i,m}T_{j,m} = \sum_{k=0}^{m-1} t_{i,k}t_{j,k}$ là tích vô hướng của 2 vector $T_{i,m}$ và $T_{j,m}$

μ_i và μ_j lần lượt là kì vọng của $T_{i,m}$ và $T_{j,m}$

σ_i, σ_j lần lượt là độ lệch chuẩn của $T_{i,m}$ và $T_{j,m}$

Trong công thức (1) ta có thể tính trước kì vọng và độ lệch chuẩn của tất cả các chuỗi con trong chuỗi thời gian với độ phức tạp là O(*n*) bằng cách áp dụng kĩ thuật được giới thiệu trong Rakthanmanon và cộng sự (2013). Ngoài ra, trong Zhu và cộng sự (2016) đã chứng minh $T_{i+1}T_{j+1}$ có thể được tính với thời gian là O(1) một khi đã biết trước T_iT_j

$$T_{i+1}T_{j+1} = T_iT_j - t_it_j + t_{i+m}t_{j+m} \quad (2)$$

Mối quan hệ giữa $T_{i+1}T_{j+1}$ và T_iT_j chỉ ra là một khi chúng ta có vector khoảng cách D_i của chuỗi con $T_{i,m}$ với các chuỗi con khác trong *T*, ta có thể tính vector khoảng cách D_{i+1} của chuỗi con $T_{i+1,m}$ với các chuỗi con khác trong *T* chỉ trong thời gian O(*n*).

Trong trường hợp đặc biệt khi $i = 1$ hoặc $j = 1$ ta có thể tính trước tích vô hướng của 2 vector $T_{i,m}T_{j,m}$ bằng biến đổi Fourier nhanh (FFT).

Định nghĩa 7. Một vector khoảng cách lân cận gần nhất *D* của chuỗi thời gian *T* có chiều dài *n* là một vector chứa khoảng cách của mỗi chuỗi con có chiều dài *m* với chuỗi con lân cận không tầm thường của chúng trong *T*. Nghĩa là $D = [\min(D_1), \min(D_2), \ldots, \min(D_{n\text{-}m\text{+}1})]$ với D_i ($1 \leq i \leq n\text{-}m\text{+}1$) là vector khoảng cách giữa chuỗi con $T_{i,m}$ với các chuỗi con trong *T*.

Trong vector khoảng cách lân cận gần nhất *D*, phần tử thứ *i* cho ta biết khoảng cách giữa chuỗi con $T_{i,m}$ với chuỗi con lân cận gần nhất của nó trong chuỗi thời gian *T*. Tuy nhiên, vector khoảng cách lân cận gần nhất không cho biết vị trí của chuỗi lân cận gần nhất của chuỗi $T_{i,m}$. Vì vậy cần phải sử dụng thêm một vector chỉ mục chứa vị trí các chuỗi lân

cận gần nhất của từng chuỗi con trong chuỗi thời gian *T*. Hình 1 minh họa mối quan hệ giữa ma trận khoảng cách với vector khoảng cách lân cận gần nhất.

Các vector khoảng cách	D_1	D_2	…	D_{n-m+1}
D_1	$d_{1,1}$	$d_{1,2}$	…	$d_{1,n-m+1}$
D_2	$D_{2,1}$	$D_{2,2}$	…	$d_{2,n-m+1}$
…	…	…	…	…
D_{n-m+1}	$d_{n-m+1,1}$	$d_{n-m+1,2}$	…	$d_{n-m+1,n-m+1}$
Vector kc lân cận gần nhất	↓	↓	↓	↓
P	Min(D_1)	Min(D_1)	…	Min(D_{n-m+1})

Hình 1. *Ma trận khoảng cách và vector khoảng cách lân cận gần nhất của các chuỗi con trong chuỗi thời gian T*

Định nghĩa 7. Một vector chỉ mục I= [I_1, I_2, …,I_{n-m+1}] của một chuỗi thời gian *T* là một vector với I_i,∀1≤ i ≤ n-m+1, là số nguyên chỉ vị trí của chuỗi lân cận gần nhất với chuỗi con $T_{i,m}$. Chẳng hạn, $I_i = k$ cho biết chuỗi con $T_{i,m}$ có chuỗi con lân cận gần nhất với nó là chuỗi con $T_{k,m}$.

2.1.2. Các công trình liên quan

Nhiều thuật toán đã được giới thiệu để giải quyết bài toán phát hiện motif trên chuỗi thời gian từ khi bài toán này được hình thức hóa vào năm 2002 (Lin et al., 2002). Trong nghiên cứu của mình, Lin và các cộng sự định nghĩa bài toán phát hiện motif trên chuỗi thời gian dựa vào một ngưỡng *R* và một chiều dài motif *m* do người dùng xác định. Do độ phức tạp cao của thuật toán phát hiện motif chính xác, các nhà nghiên cứu đã chuyển sang nghiên cứu các giải pháp phát hiện motif xấp xỉ. Nói chung, các giải pháp này thường có độ phức tạp là O(*n*) hoặc O(*nlogn*) (*n* là số chuỗi trong cơ sở dữ liệu chuỗi thời gian hay chiều dài của chuỗi thời gian mà từ đó các chuỗi con được trích ra) với một số các tham số phải xác định trước (Mueen et al., 2009). Năm 2003, Chiu và các cộng sự đề xuất thuật toán chiếu ngẫu nhiên để phát hiện motif trên chuỗi thời gian theo cách tiếp cận xấp xỉ (B. Chiu et al., 2003). Thuật toán này dựa trên kĩ thuật băm bảo toàn tính lân cận (locality preserving hashing) và sử dụng phương pháp rời rạc hóa SAX để biểu diễn các chuỗi con trong chuỗi thời gian ban đầu. Độ phức tạp của thuật toán này là tuyến tính theo độ dài của từ SAX, số chuỗi con, số lần lặp và số lần đụng độ.

Trong (Yankov et al., 2007), Yankov và các cộng sự đã giới thiệu một thuật toán được cải tiến từ thuật toán chiếu ngẫu nhiên để có thể khám phá motif mà không bị ảnh hưởng bởi sự biến đổi co giãn theo trục hoành. Khái niệm về motif trên dữ liệu chuỗi thời gian đã được định nghĩa lại theo nghĩa "lân cận gần nhất": motif là một cặp chuỗi con giống nhau nhất trong một chuỗi thời gian dài. Cách tiếp cận này có nhược điểm giống như thuật toán chiếu ngẫu nhiên. Ngoài ra tổng chi phí của cách tiếp cận này cao do phải tìm các hệ số biến đổi co giãn theo trục hoành tốt nhất.

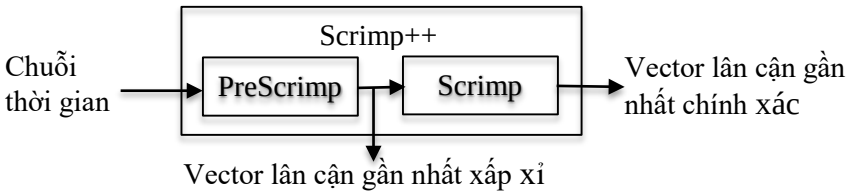
Với định nghĩa motif trong chuỗi thời gian là cặp chuỗi hoặc chuỗi con giống nhau nhất, năm 2009 Mueen và các cộng sự đã giới thiệu một thuật toán phát hiện motif chính xác, gọi là giải thuật MK (Mueen et al., 2009). Cách tiếp cận này sử dụng các điểm tham chiếu được chọn ngẫu nhiên và ý tưởng từ bỏ sớm việc tính toán khoảng cách Euclid khi tổng tích lũy khoảng cách hiện hành lớn hơn khoảng cách của ứng viên motif tốt nhất tại thời điểm đang xét. Quá trình phát hiện motif của thuật toán này dựa vào thông tin heuristic được xác định bởi thứ tự của khoảng cách giữa đối tượng đang xét với các điểm tham chiếu ngẫu nhiên. Mueen và các cộng sự đã cho thấy rằng thuật toán này có thể thực hiện nhanh hơn gấp vài ngàn lần thuật toán brute-force khi tìm trong cơ sở dữ liệu lớn, dù trong trường hợp xấu nhất độ phức tạp của thuật toán là bậc hai.

Trong Nguyen và cộng sự (2016), các tác giả đã giới thiệu thuật toán phát hiện k motif sử dụng cấu trúc chỉ mục đa chiều và chỉ mục đường chân trời (skyline index) kết hợp với phương pháp thu giảm số chiều MP_C. Hai phương pháp này giúp tăng tốc độ tìm kiếm lân cận gần nhất của một chuỗi con. Tuy nhiên, chúng có nhược điểm là cần nhiều tham số đầu vào. (Nguyen et al., 2006)

Gần đây, một số thuật toán đã được đề xuất cho bài toán phát hiện motif trên chuỗi thời gian dựa vào vector khoảng cách lân cận gần nhất (Yeh et al., 2016; Zhu et al., 2016; Zhu et al., 2018). Vì thời gian tính toán các vector khoảng cách chiếm thời gian lâu nhất, các thuật toán theo hướng tiếp cận này thường tập trung vào việc giảm thời gian tính toán các vector khoảng cách. Thuật toán Stamp (Yeh et al., 2016) thực hiện tính khoảng cách giữa chuỗi con $T_{i,m}$ với mỗi chuỗi con trong T theo thứ tự ngẫu nhiên và sử dụng phép biến đổi Fourier nhanh để tính tích vô hướng giữa hai chuỗi con. Độ phức tạp tính toán một vector khoảng cách của thuật toán này là O($nlogn$) với n là chiều dài chuỗi thời gian T và độ phức tạp về thời gian của toàn bộ tiến trình là O(n^2logn) (Zhu et al., 2018). Thuật toán Stomp (Zhu et al., 2016) được cải tiến từ thuật toán Stamp bằng cách thay vì tính các vector khoảng cách một cách độc lập như thuật toán Stamp, thuật toán Stomp tính vector khoảng cách dựa vào sự phụ thuộc của các khoảng cách giữa các chuỗi con liên tiếp nhau. Nghĩa là tích vô hướng của hai chuỗi con trong công thức tính khoảng cách được tính dựa vào tích vô hướng của hai chuỗi con đã được tính trước đó. Chi phí tính toán của thuật toán Stomp chỉ còn O(n^2). Thuật toán Scrimp++ kết hợp các đặc trưng của hai thuật toán Stamp và Stomp như thời gian tính toán không phụ thuộc vào chiều dài chuỗi con, phí tổn bộ nhớ thấp. Thuật toán Scrimp++ thực hiện phát hiện motif qua hai giai đoạn: Đầu tiên xử lí bằng thuật toán PreScrimp để có vector lân cận gần nhất xấp xỉ, sau đó tiếp tục tinh chỉnh vector lân cận gần nhất bằng thuật toán Scrimp cho đến khi nó hội tụ đến kết quả chính xác. Tuy độ phức tạp về thời gian của thuật toán vẫn là O(n^2) nhưng cho kết quả hội tụ nhanh hơn.

2.2. Thuật toán Scrimp++ và đề xuất cải tiến thuật toán

Thuật toán Scrimp++ (Zhu et al., 2018) được xây dựng dựa trên hai thuật toán PreScrimp và Scrimp. Hình 2 minh họa ý tưởng của thuật toán này.



Hình 2. *Ý tưởng thuật toán Scrimp++*

2.2.1. Thuật toán PreScrimp

Thuật toán PreScrimp dựa trên tính chất duy trì lân cận liên tục (Consecutive Neighborhood Preserving (CNP) Property) của các chuỗi con trong chuỗi thời gian (Zhu et al., 2018). Tính chất này có thể mô tả như sau: vector chỉ mục có thể được phân chia thành nhiều đoạn các giá trị liên tục. Trong mỗi đoạn, một tập các chuỗi con liên tiếp sẽ có một tập các chuỗi con liên tiếp khác là lân cận gần nhất với nó. Tính chất này gọi là tính chất CNP. Hình 3 minh họa tính chất CNP.

Vị trí chuỗi con	1	2	3	4	…	59	60	61	…
Vị trí lân cận gần nhất	70	71	111	112	…	189	190	191	…

Hình 3. *Minh họa vector chỉ mục I*

Vì các chuỗi con liên tiếp trong chuỗi thời gian thường có đoạn giao nhau nên nếu chuỗi con thứ i giống với chuỗi con thứ j thì chuỗi con thứ i+1 có xác suất cao sẽ giống với chuỗi con thứ j+1. Các chuỗi mẫu được lấy từ chuỗi thời gian T cách nhau một khoảng cố định s. Dễ thấy là nếu chọn s càng nhỏ thì độ chính xác của thuật toán càng cao, nhưng thời gian thực thi của thuật toán càng lâu. Tác giả đã đề xuất nên chọn $s = m/4$ để đảm bảo tất cả các chuỗi con đều có phần giao với ít nhất một chuỗi mẫu ít nhất là 87,5%.

Với mỗi chuỗi mẫu, thuật toán sẽ tìm chuỗi con lân cận gần nhất với nó dựa vào tính chất CNP. Giả sử, $T_{i,m}$ là một chuỗi mẫu và chuỗi con lân cận gần nhất với nó là $T_{j,m}$. Theo tính chất CNP thì có khả năng cao lân cận gần nhất với chuỗi con $T_{i+k,m}$ là $T_{j+k,m}$ (với k=*-s+1, -s+2, …, -2, -1, 1, 2, …, s-2, s-1*). Khoảng cách giữa các cặp chuỗi con này được tính toán và nếu nó nhỏ hơn khoảng cách tương ứng hiện có trong vector khoảng cách lân cận gần nhất thì vector lân cận gần nhất sẽ được cập nhật. Hình 4 minh họa thuật toán này.

Dữ liệu vào: Chuỗi thời gian T, chiều dài chuỗi con m, khoảng cách lấy mẫu s
Dữ liệu ra: Véc tơ lân cận gần nhất P và véc tơ chỉ mục I

(1) Tính kì vọng và độ lệch chuẩn của tất cả các chuỗi con
(2) Duyệt qua lần lượt các vị trí chuỗi con được lấy mẫu trong T với khoảng cách lấy mẫu là s, thực hiện:
- Chọn ngẫu nhiên 1 chuỗi mẫu $T_{i,m}$
- Tìm chuỗi con lân cận gần nhất $T_{j,m}$ của $T_{i,m}$ sử dụng độ đo (1).
- Cập nhật vào vector lân cận gần nhất P và vector chỉ mục I
- Tính tích vô hướng 2 chuỗi con $T_{i,m}$ và $T_{j,m}$
- Duyệt lần lượt qua các cặp chuỗi ($T_{i+k,m}$, $T_{j+k,m}$), k=1, 2, … cho đến khi gặp chuỗi mẫu

kế hoặc đến cuối chuỗi thời gian, với mỗi cặp chuỗi thực hiện:

+ Tính tích vô hướng của cặp chuỗi ($T_{i+k,m}$, $T_{j+k,m}$) theo công thức (2)
+ Tính khoảng cách của cặp chuỗi ($T_{i+k,m}$, $T_{j+k,m}$) theo công thức (1)
+ Cập nhật lại giá trị trong vector lân cận gần nhất P và vector chỉ mục I nếu khoảng cách vừa tính nhỏ hơn giá trị của phần tử tương ứng $P_{i+k,m}$ hay $P_{j+k,m}$ trong P

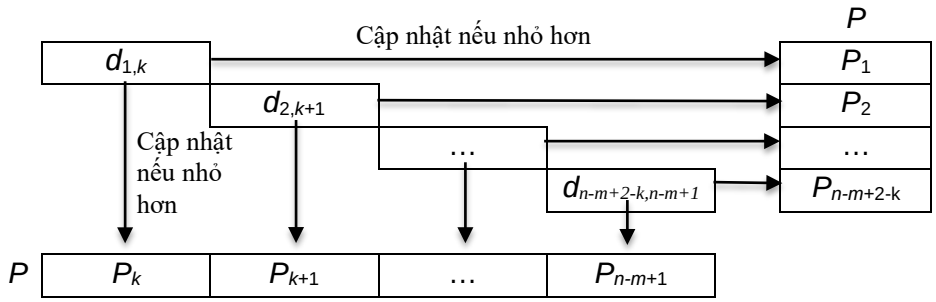
- Duyệt lần lượt qua các cặp chuỗi ($T_{i-k,m}$, $T_{j-k,m}$), k=1, 2, … cho đến khi gặp chuỗi mẫu phía trước hoặc đến đầu chuỗi thời gian, với mỗi cặp chuỗi thực hiện:
  + Tính tích vô hướng của cặp chuỗi ($T_{i-k,m}$, $T_{j-k,m}$) theo công thức (2)
  + Tính khoảng cách của cặp chuỗi ($T_{i-k,m}$, $T_{j-k,m}$) theo công thức (1)
  + Cập nhật lại giá trị trong vector lân cận gần nhất P và vector chỉ mục I nếu khoảng cách vừa tính nhỏ hơn giá trị của phần tử tương ứng $P_{i-k,m}$ hay $P_{j-k,m}$ trong P

(3) Trả về P và I

Hình 4. *Minh họa thuật toán PreScrimp*

2.2.2. Thuật toán Scrimp

Thuật toán Scrimp (Zhu et al., 2018) được các tác giả cải tiến từ thuật toán Stomp dựa trên nhận xét: thuật toán Stomp tính toán ma trận khoảng cách (Hình 1) theo thứ tự từng dòng và cập nhật vào vector khoảng cách. Tính toán theo thứ tự này sẽ ngăn cản việc phát hiện sớm motif nằm ở vị trí cuối của chuỗi thời gian T. Để khắc phục điều này, các tác giả thực hiện tính toán ma trận khoảng cách theo đường chéo theo thứ tự ngẫu nhiên vì công thức (2) có thể giúp thực hiện điều này. Các khoảng cách $d_{1,k}$, $d_{2,k}$, …, $d_{n-m+2-k,n-m+1}$ được tính toán lần lượt từng cái một. Vector lân cận gần nhất P và vector chỉ mục I sẽ được cập nhật Nếu $d_{i,i+k-1}$ nhỏ hơn P_i hay P_{i+k-1}. Hình 5 minh họa điều này và thuật toán Scrimp được minh họa trong Hình 6.



Hình 5. *Minh họa cách tính khoảng cách theo đường chéo và cập nhật vào P*

Dữ liệu vào: Chuỗi thời gian T, chiều dài chuỗi con m
Dữ liệu ra: Véc tơ lân cận gần nhất P và Véc tơ chỉ mục I

(1) Tính kì vọng và độ lệch chuẩn của tất cả các chuỗi con
(2) Tạo một dãy thứ tự tính toán ngẫu nhiên Q cho các phần tử trên đường chéo
(3) Với mỗi k trong Q
  - Với mỗi i từ 1 tới n-m+2-k
    + Nếu i = 1: tính tích vô hướng của $T_{1,m}$ và $T_{k,m}$
      Ngược lại, tính tích vô hướng của $T_{i,m}$ và $T_{i+k-1,m}$ theo công thức (2)

+ Tính khoảng cách dist($T_{i,m}$, $T_{i+k-1,m}$) theo công thức (1)
+ Nếu khoảng cách vừa tính nhỏ hơn giá trị tương ứng P_i hoặc P_{i+k-1}
thì cập nhật vào véc tơ khoảng cách lân cận gần nhất P và véc tơ chỉ mục I.

(4) Trả về P và I

Hình 6. *Minh họa thuật toán Scrimp*

2.2.3. Cải tiến thuật toán Scrimp++

Để tính được đầy đủ vector lân cận gần nhất, thuật toán Scrimp++ phải duyệt qua tất cả các chuỗi mẫu. Với mỗi chuỗi mẫu thuật toán phải tính khoảng cách giữa chuỗi mẫu và các chuỗi con khác để tìm lân cần gần nhất của nó. Chuỗi thời gian càng dài thì số lượng cặp chuỗi con cần so sánh càng nhiều để tìm ra cặp giống nhau nhất. Liệu có thể giảm số lần so sánh các cặp chuỗi con mà vẫn có thể tìm ra cặp chuỗi con giống nhau nhất?

Để giải quyết vấn đề nêu trên và nhằm mục đích tăng tốc độ thực thi của thuật toán, chúng tôi áp dụng ý tưởng dưới đây vào thuật toán Scrimp++: (1) Dùng bài toán nghịch lí ngày sinh để xác định số lượng chuỗi con cần xem xét và (2) xác định thứ tự chuỗi mẫu cần xem xét. Ý tưởng này tương tự như ý tưởng được đề xuất trong (Pariwatthanasak et al., 2019). Tuy nhiên, có một sự khác biệt giữa nghiên cứu của chúng tôi và nghiên cứu trong bài báo của Pariwatthanasak và các cộng sự. Đó là Pariwatthanasak và các cộng sự áp dụng bài toán nghịch lí ngày sinh và xác định thứ tự chuỗi mẫu cần xem xét vào thuật toán MASS để tính vector khoảng cách lân cận gần nhất, trong khi chúng tôi áp dụng hai ý tưởng trên vào thuật toán Scrimp++ để xác định cặp chuỗi con lân cận gần nhất có khoảng cách nhỏ nhất (motif). Ý tưởng cải tiến như sau:

- Xác định trước số lượng chuỗi con k cần xem xét (số lần lặp trong thuật toán Scrimp++) để tìm được cặp chuỗi con giống nhau nhất dựa vào bài toán nghịch lí ngày sinh. Với giả sử rằng mỗi ngày trong năm (trừ ngày 29 tháng hai) có xác suất ngang nhau, bài toán này được phát biểu như sau: Trong một nhóm ngẫu nhiên 23 người, xác suất để 2 người trong nhóm có cùng ngày sinh nhật là khoảng 50%.

Bài toán tìm cặp chuỗi con giống nhau nhất trong một chuỗi thời gian tương tự như bài toán nghịch lí ngày sinh. Vì vậy, ta có thể áp dụng bài toán nghịch lí ngày sinh để giảm số lần so sánh các cặp chuỗi con trong thuật toán Scrimp++.

Xác xuất để 2 người trong một nhóm có cùng ngày sinh nhật có thể xem giống như xác suất để 2 chuỗi con giống nhau nhất trong một chuỗi thời gian. Trong đó, n-m+1 gian có độ dài n ($m << n$) được xem tương tự như 365 ngày trong năm và n-m+1 chuỗi con được xem tương tự như số người trong nhóm.

Với ý tưởng của bài toán nghịch lí ngày sinh ta có thể tính được số lượng chuỗi con k cần thiết phải xem xét để tìm được cặp chuỗi con giống nhau nhất dựa trên công thức tính xác suất để có 2 chuỗi con giống nhau trong k chuỗi con như sau:

$$P(k,m,n) = 1 - e^{\frac{-k^2}{2(n-m+1)}} \quad (3)$$

Hình 7 minh họa thuật toán xác định số lượng chuỗi con k cần thiết phải xem xét để

tìm được cặp chuỗi con giống nhau nhất.

Input: Chiều dài chuỗi thời gian n, chiều dài chuỗi con m và ngưỡng xác suất p
Output: Số chuỗi con k

$k = 1$
xacsuat = 0
Trong khi xacsuat < p

$$xacsuat = 1 - e^{\frac{-k^2}{2*(n-m+1)}}$$

$k = k + 1$
Trả về k

Hình 7. *Minh họa thuật toán xác định số lần lặp k*

- Thứ tự chuỗi mẫu cần xem xét: Chuỗi mẫu đầu tiên $T_{i,m}$ được lấy ngẫu nhiên như thuật toán Scrimp++. Giả sử chuỗi con lân cận gần nhất với $T_{i,m}$ là $T_{j,m}$. Các chuỗi mẫu tiếp theo được chọn theo nguyên tắc: Nếu $T_{j,m}$ đã được chọn làm chuỗi mẫu trước đó thì chuỗi mẫu tiếp theo vẫn được chọn ngẫu nhiên, ngược lại chọn $T_{j,m}$ làm chuỗi mẫu tiếp theo.

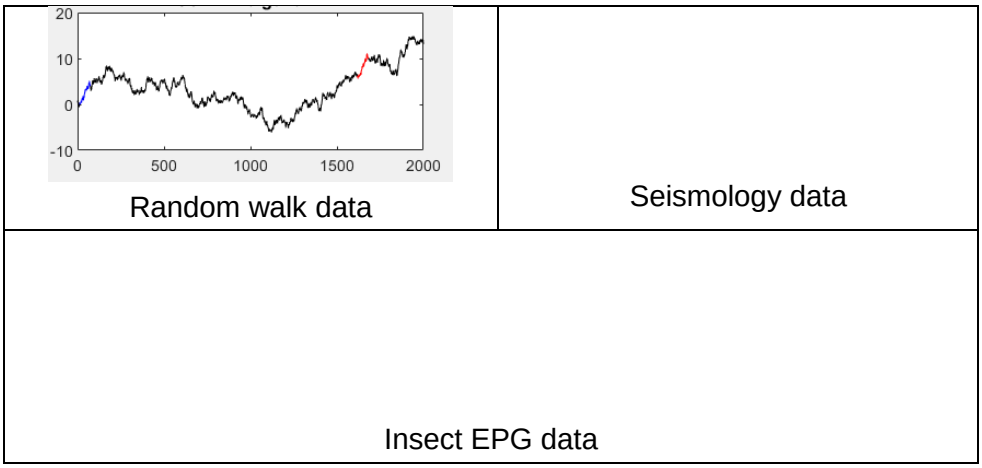
2.3. Đánh giá bằng thực nghiệm

Trong nghiên cứu này, chúng tôi so sánh thuật toán Scrimp++ cải tiến và thuật toán Scrimp và Scrimp++. Các thuật toán được so sánh bằng thực nghiệm với các trường hợp chiều dài motif khác nhau và chiều dài tập dữ liệu khác nhau. Sự so sánh dựa trên 2 tiêu chí thời gian thực thi và độ chính xác. Độ chính xác được đánh giá dựa vào vị trí motif mà thuật toán xác định được.

2.3.1. Môi trường và dữ liệu thực nghiệm

Các thực nghiệm được thực hiện trên máy tính Dell Inspiron 15, Inter® core™ i5-4200U CPU @1.60GHz, 6GB RAM, hệ điều hành Windows 10.

Thực nghiệm được thực hiện trên 1 tập dữ liệu nhân tạo được phát sinh ngẫu nhiên (Random walk data) và hai tập dữ liệu thực: dữ liệu sóng địa chấn (Seismology data), dữ liệu về hành vi của côn trùng (Insect EPG data). Các tập dữ liệu này được lấy trong khoảng 5/2019, từ nguồn https://sites.google.com/site/scrimpplusplus/. Hình 8 minh họa hình ảnh các tập dữ liệu này



Hình 8. *Hình ảnh các tập dữ liệu dùng trong thực nghiệm*

2.3.2. Kết quả thực nghiệm

Bảng 1 là kết quả thực nghiệm ba thuật toán trên tập dữ liệu Random walk với độ dài là 10.000 điểm và chiều dài motif thay đổi từ 64 đến 1024. Bảng 2 là kết quả thực nghiệm của thuật toán trên tập dữ liệu Random walk với độ dài thay đổi từ 8000 đến 30.000 và chiều dài motif là 64.

Kết quả thực nghiệm trên tập dữ liệu này cho thấy thời gian thực thi của thuật toán Scrimp++ cải tiến nhanh hơn khoảng 5 lần (tính trung bình) so với 2 thuật toán còn lại. Trong đa số trường hợp độ chính xác của thuật toán là tương đương nhau. Chỉ riêng trường hợp độ dài tập dữ liệu là 10.000 và độ dài motif là 256 thì vị trí chuỗi con đầu tiên của motif do thuật toán Scrimp++ cải tiến tìm được bị lệch so với vị trí chuỗi con đầu tiên của motif do 2 thuật toán còn lại tìm được. Kết quả thực nghiệm cũng cho thấy thời gian thực thi của các thuật toán không ảnh hưởng nhiều bởi chiều dài motif mà chỉ bị ảnh hưởng bởi chiều dài của chuỗi dữ liệu.

Bảng 1. *Kết quả thực nghiệm trên tập Random walk với chiều dài là 10.000, chiều dài motif thay đổi*

Chiều dài motif	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
64	(1957, 9839)	(1957, 9839)	(1957, 9839)	3,80	4,35	0,85
128	(5919,7133)	(5919,7133)	(5919,7133)	3,69	4,13	0,76
256	(8585,9271)	(8585,9271)	(1075,9271)	3,68	3,96	0,45
512	(824,8334)	(824,8334)	(824,8334)	3,65	3,78	0,40
1024	(311,7821)	(311,7821)	(311,7821)	3,25	3,17	0,32

Bảng 2. *Kết quả thực nghiệm trên tập Random walk với chiều dài thay đổi, chiều dài motif là 64*

Chiều dài tập dữ liệu	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
8000	(1265,7193)	(1265,7193)	(1265,7193)	2,68	3,83	0,90
10000	(1957, 9839)	(1957, 9839)	(1957, 9839)	3,80	4,35	0,85
15000	(8779,14593)	(8779,14593)	(8779,14593)	9,51	10,91	2,21
20000	(2055,15013)	(2055,15013)	(2055,15013)	16,70	18,19	2,91
25000	(2055,15013)	(2055,15013)	(2055,15013)	26,35	29,05	4,33
30000	(2055,15013)	(2055,15013)	(2055,15013)	37,41	44,62	6,20

Bảng 3 là kết quả thực nghiệm trên tập dữ liệu Seismology với chiều dài cố định là 10000 và chiều dài motif thay đổi. Bảng 4 là kết quả thực nghiệm trên tập dữ liệu Seismology với chiều dài thay đổi và chiều dài motif cố định là 64.

Bảng 3. *Kết quả thực nghiệm trên tập Seismology với chiều dài là 10000, chiều dài motif thay đổi*

Chiều dài motif	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
64	(4601,7865)	(4601,7865)	(4601,7865)	4,74	5,81	0,97
128	(9614,9840)	(9614,9840)	(9614,9840)	4,60	5,07	0,71
256	(7021,9472)	(7021,9472)	(7021,9472)	4,20	4,17	0,49
512	(2056, 9288)	(2056, 9288)	(2056, 9288)	4,36	4,03	0,48
1024	(382,1641)	(382,1641)	(382,1641)	3,99	3,27	0,36

Bảng 4. *Kết quả thực nghiệm trên tập Seismology với chiều dài thay đổi, chiều dài motif là 64*

Chiều dài tập dữ liệu	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
8000	(4061,7865)	(4061,7865)	(4061,7865)	3,53	3,68	0,72
10000	(4601,7865)	(4601,7865)	(4601,7865)	4,74	5,81	0,97
15000	(8218, 13218)	(8218, 13218)	(8218, 13218)	10,82	11,29	1,89
20000	(10875,15695)	(10875,15695)	(10875,15695)	19,75	22,02	3,74
25000	(10875,15695)	(10875,15695)	(10875,15695)	27,81	32,17	4,56
30000	(10875,15695)	(10875,15695)	(10875,15695)	40,98	42,99	6,75

Kết quả thực nghiệm trên tập dữ liệu này cho thấy thời gian thực thi của thuật toán

Scrimp++ cải tiến nhanh hơn khoảng 6 lần (tính trung bình) so với 2 thuật toán còn lại và độ chính xác của 3 thuật toán là tương đương nhau. Kết quả thực nghiệm cũng cho thấy thời gian thực thi của các thuật toán không ảnh hưởng nhiều bởi chiều dài motif mà chỉ bị ảnh hưởng bởi chiều dài của chuỗi dữ liệu.

Bảng 5 là kết quả thực nghiệm trên tập dữ liệu Insect EPG với chiều dài cố định là 10000 và chiều dài motif thay đổi. Bảng 6 là kết quả thực nghiệm trên tập dữ liệu Insect EPG với chiều dài thay đổi và chiều dài motif cố định là 64.

Kết quả thực nghiệm trên tập dữ liệu này cho thấy thời gian thực thi của thuật toán Scrimp++ cải tiến nhanh hơn khoảng 6 lần (tính trung bình) so với 2 thuật toán còn lại và độ chính xác của 3 thuật toán là tương đương nhau. Kết quả thực nghiệm cũng cho thấy thời gian thực thi của các thuật toán không ảnh hưởng nhiều bởi chiều dài motif mà chỉ bị ảnh hưởng bởi chiều dài của chuỗi dữ liệu.

Bảng 5. *Kết quả thực nghiệm trên tập Seismology với chiều dài là 10.000, chiều dài motif thay đổi*

Chiều dài motif	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
64	(6189,7109)	(6189,7109)	(6189,7109)	4,74	4,76	0,86
128	(2459,9839)	(2459,9839)	(2459,9839)	4,58	4,52	0,61
256	(2531,6217)	(2531,6217)	(2531,6217)	3,91	4,00	0,53
512	(2507,2757)	(2507,2757)	(2507,2757)	3,96	4,00	0,42
1024	(2587,6023)	(2587,6023)	(2587,6023)	3,41	3,37	0,35

Bảng 6. *Kết quả thực nghiệm trên tập Seismology với chiều dài thay đổi, chiều dài motif là 64*

Chiều dài tập dữ liệu	Vị trí motif			Thời gian thực thi (giây)		
	Scrimp	Scrimp++	Scrimp++ cải tiến	Scrimp	Scrimp++	Scrimp++ cải tiến
8000	(8218,13247)	(8218,13247)	(8218,13247)	2,39	3,48	0,70
10000	(6189,7109)	(6189,7109)	(6189,7109)	4,74	4,76	0,86
15000	(10875,15695)	(10875,15695)	(10875,15695)	8,65	9,17	1,79
20000	(10875,15695)	(10875,15695)	(10875,15695)	15,04	15,51	2,63
25000	(10875,15695)	(10875,15695)	(10875,15695)	23,31	25,48	4,08
30000	(10875,15695)	(10875,15695)	(10875,15695)	32,92	35,46	5,38

3. Kết luận

Trong bài báo này, chúng tôi giới thiệu một phiên bản cải tiến của thuật toán Scrimp++ bằng cách áp dụng bằng cách áp dụng ý tưởng bài toán nghịch lí ngày sinh và thứ tự chọn chuỗi mẫu vào thuật toán Scrimp++ và đánh giá bằng thực nghiệm phương

pháp đề xuất trên trên ba tập dữ liệu (1 tập dữ liệu được tạo ngẫu nhiên và 2 tập dữ liệu thực). Kết quả thực nghiệm cho thấy thời gian thực thi của thuật toán cải tiến nhanh hơn khoảng 5 đến 6 lần (tính trung bình) so với thời gian thực thi của 2 thuật toán còn lại nhưng vẫn giữ được độ chính xác tương đương trong hầu hết các trường hợp thực nghiệm.

Trong tương lai, chúng tôi sẽ tiếp tục cải tiến thuật toán đề xuất sao cho độ chính xác của thuật toán tương đương với 2 thuật toán còn lại trong mọi trường hợp nhưng vẫn đảm bảo về thời gian thực thi của thuật toán tốt so với 2 thuật toán còn lại.

- ***Tuyên bố về quyền lợi:*** *Các tác giả xác nhận hoàn toàn không có xung đột về quyền lợi.*
- ***Lời cảm ơn:*** *Bài báo được thực hiện từ kinh phí đề tài nghiên cứu khoa học năm 2021 của Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh.*

TÀI LIỆU THAM KHẢO

Cao, D. T., & Duong, T. A. (2015). An efficient method for motif and anomaly detection in time series based on clustering. *International Journal of Business Intelligence and Data Mining, 10*, 356-377.

Castro, N., & Azevedo, P. (2010). Multiresolution Motif Discovery in Time Series. *the SIAM International Conference on Data Mining (SDM 2010)*, (pp. 665-676). Columbus, Ohio, USA.

Chiu, B., Keogh, E.,& Lonardi, S. (2003). Probabilistic discovery of time series motifs. *the 9th International Conference on Knowledge Discovery and Data mining (KDD'03)*, (pp. 493-498).

Ferreira, P., Azevedo, P., Silva, C., & Brito, R. (2006). Mining approximate motifs in time series. *the 9th Int. Conf. on Discovery Science*, (pp. 89-101).

Lin, J., Keogh, E., Lonardi, S., & Patel, P. (2002). Finding motifs in time series. *Proc. of 2nd Workshop on Temporal Data Mining (KDD'02).*

Lin, Y., McCool, M. D., & Ghorbani, A. A. (2010). Motif and anomaly discovery of time series based on subseries join. *IAENG International Conference on Data Mining and Applications, ICDMA.*

Mohammad, Y., & Nishida, T. (2014). Exact Discovery of Length-Range Motifs. *series Lecture Notes in Computer Science, 8398*, 23-32.

Mueen, A., Keogh, E., Zhu, Q., & Cash, S. (2009). Exact Discovery of Time Series Motifs. *Proc. of SIAM Int. on Data Mining*, 473-484.

Nguyen. T. S., & Duong, T. A. (2016). Discovery of time series k-motifs based on multidimensional index. *Knowledge and Information Systems, 46*(1), 59-86.

Pariwatthanasak, K., & Ratanamahatana, C. A. (2019). Time Series Motif Discovery Using Approximated Matrix Profile. In S. S. Yang XS. (Ed.), *Third International Congress on Information and Communication Technology, 797*. Springer, Singapore.

Rakthanmanon, T., Campana, B. J. L., Mueen, A., Batista, G., Westover, M. B., Zhu, Q., J., Zakaria, & Keogh, E. J. (2013). Addressing Big Data Time Series: Mining trillions of time series subsequences under dynamic time warping. *TKDD 7.3 (2013)*, 10.

Yankov, D., Keogh, E., Medina, J., Chiu, B., & Zordan, V. (2007). Detecting Motifs Under Uniform Scaling. *the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 844-853).

Yeh, C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., & Silva, D. F. (2016). Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, (pp. 1317-1322).

Zhu Y., Yeh, C.-C. M., Zimmerman, Z., Kamgar, K., & Keogh, E. (2018). Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive. *ICDM*.

Zhu, Y., Zimmerman Z., Senobari, N. S., Yeh, C.C. M., Funning, G., Mueen, A., Brisk, P., & Keogh, E. (2016). Matrix profile II: Exloiting a Novel Algorithm and CPUs yo break he one Hunded Milion Barries for Time Series motifs and joins. *ICDM*.

DISCOVERING TIME SERIES MOTIF USING THE IMPROVED SCRIMP++ ALGORITHM

Nguyen Thanh Son*, Tran Thi Dung

HCM City University of Technology and Education, Vietnam

**Corresponding author: Nguyen Thanh Son – Email: sontt@hcmue.edu.vn*

Received: September 27, 2021; Revised: March 14, 2022; Accepted: March 18, 2022

ABSTRACT

A time series motif is a nearest neighbor subsequence pair of a long time series or a nearest neighbor sequence pair in a time series database. Time series motif discovery is a vital task in time series data mining. Recently, some algorithms were proposed for discovering time series motifs. These algorithms use a vector called matrix profile which contains distances between each subsequence and its nearest neighbor in time series. The vector is computed based on the combination between time series normalization and Euclidean distance measure. The typical method for the approach is the Scrimp++ algorithm. This paper introduces an improved version of this algorithm to improve the speed of the algorithm. The experimental results shows that our proposed algorithm outperforms the original algorithm in terms of running time with same accuracy.

Keywords: matrix profile; motif discovery; time series; Scrimp++ algorithmtime series motif