TĂNG TỐC KẾT CHUỖI CON VỚI ĐỘ ĐO XOẮN THỜI GIAN ĐỘNG DỰA VÀO SỰ HỖ TRỢ CỦA GPU

Lâm Trường An[1], Dương Tuấn Anh[2,3,*]

[1] Công ty LUXOFT
[2] Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM
[3]Trường Đại Học Bách Khoa, ĐHQG-HCM

lamtruongan0106@gmail.com,anh.dt@huflit.edu.vn

TÓM TẮT— Kết chuỗi con trên chuỗi thời gian là một trong những bài toán khai phá dữ liệu chuỗi thời gian quan trọng. Trong nghiên cứu này, chúng tôi sử dụng một giải thuật kết chuỗi con hữu hiệu, có tên EP-M, mà dựa vào độ đo xoắn thời gian động (dynamic time warping - DTW) và phân đoạn chuỗi thời gian. Do tính hiệu quả cao của độ đo DTW so với các độ đo khoảng cách khác, đã có nhiều nỗ lực trong cộng đồng nghiên cứu nhằm khắc phục tình trạng chi phí tính toán cao của độ đo DTW. Các kỹ thuật phần mềm để tăng tốc việc tính toán độ đo DTW trong khai phá chuỗi thời gian gồm có các kỹ thuật từ bỏ sớm, cắt tỉa dựa vào cận dưới và lập chỉ mục. Tuy nhiên, bên cạnh những giải pháp dựa vào phần mềm như vậy chúng ta nên quay sang những kỹ thuật dựa vào phần cứng để cải thiện hơn nữa tính hữu hiệu của phương pháp hiện có. Trong nghiên cứu này, chúng tôi thiết kế và đánh giá một giải pháp dựa vào phần cứng, đó là đơn vị xử lý đồ họa (Graphics Processing Unit -GPU) để tăng tốc giải thuật kết chuỗi con EP-M bằng cách song song hóa. Kết quả thực nghiệm trên một số bộ dữ liệu mẫu cho thấy tính hữu hiệu cao của phương pháp dựa vào GPU đề xuất cho công tác kết chuỗi con trên chuỗi thời gian.

Từ khóa— chuỗi thời gian, kết chuỗi con, độ đo xoắn thời gian động, phân đoạn, đơn vị xử lý đồ họa GPU.

I. GIỚI THIỆU

Kết chuỗi con trên *chuỗi thời gian* (time series) là tìm kiếm những cặp chuỗi con tương tự nhau trên hai chuỗi thời gian lớn. Hai chuỗi thời gian có thể kết với nhau tại bất cứ vị trí nào và có chiều dài bất kỳ. Kết chuỗi con là một thao tác có tính đối xứng và đem lại những chuỗi con trùng khớp nhau theo kiểu ánh xạ nhiều-nhiều. Kết chuỗi con là sự tổng quát hóa cả hai tác vụ *so trùng toàn chuỗi* (whole sequence matching) và *so trùng chuỗi con* (subsequence matching). Kết chuỗi con đem lại những thông tin hữu ích về những đoạn biến thiên khá giống nhau của hai chuỗi thời gian. Thí dụ, trong lãnh vực thị trường tiền tệ, rất có ích khi thấy được mối tương quan giữa hai tỉ giá ngoại tệ để có thể đưa ra được những quyết định đầu tư. Trong trường hợp này, chúng ta có thể thực hiện kết chuỗi con trên hai đường biểu diễn của hai tỉ giá ngoại tệ để thấy được những mẫu biến thiên hoặc xu hướng biến thiên giống nhau của hai ngoại tệ. Một thí dụ khác, là trường hợp quan sát hai chuỗi thời gian biểu diễn cao độ nốt nhạc của hai bản nhạc, nếu giữa hai chuỗi thời gian này, khi kết chuỗi con cho ra nhiều cặp chuỗi con tương đồng nhau thì có thể thấy có hiện tượng đạo nhạc giữa hai bản nhạc này. Tác vụ kết chuỗi con hữu ích cho nhiều thao tác khai phá chuỗi khác như khám phá motif và phát hiện bất thường trên chuỗi thời gian (Lin và các cộng sự, 2010 [1]) hoặc trên các trình tự sinh học.

Đã có một số công trình nghiên cứu về kết chuỗi con trên chuỗi thời gian, có thể tóm lược như sau. Lin và McCool năm 2010 [2] đề xuất một phương pháp kết chuỗi con mà sử dụng một kỹ thuật phân đoạn không đồng đều (non-uniform segmentation) và tìm kiếm những phân đoạn kết với nhau dựa vào một hàm tương tự (similarity function) trên một tập đặc trưng. Phương pháp này khó hiện thực và có chi phí tính toán cao nên nó không thích hợp để làm việc với những chuỗi thời gian có kích thước lớn. Mueen và các cộng sự, năm 2014 [3] đề xuất một phương pháp khác để kết chuỗi con mà dựa vào sự tương quan giữa các chuỗi con của hai chuỗi thời gian. Cách tiếp cận này hướng đến việc cực đại hóa hệ số tương quan Pearson để tìm thấy chuỗi con tương quan nhất trong hai chuỗi thời gian. Do định nghĩa đặc biệt này về kết chuỗi con, giải thuật Jocor mà Mueen và các cộng sự đề nghị cũng có độ phức tạp tính toán cao. Vĩnh và Anh, năm 2016 [4], đề xuất một phương pháp hữu hiệu để kết chuỗi con trên chuỗi thời gian dựa vào với độ đo *xoắn thời gian động* (dynamic time warping –DTW) và phân đoạn. Phương pháp này, có tên là EP-M, gồm có hai bước chính: (i) phân đoạn chuỗi thời gian sử dụng kỹ thuật điểm cực trị quan trọng (Fink và Gandhi, 2007 [5]) và (ii) so trùng chuỗi con, là một quá trình lặp sử dụng cửa sổ trượt và độ đo khoảng cách DTW để tìm ra tất cả những chuỗi con tương tự nhau trên hai chuỗi thời gian. Một nhược điểm của giải thuật EP-M là vì sử dụng độ đo khoảng cách DTW chỉ với kỹ thuật từ bỏ sớm (early abandoning) nên không thể tăng tốc được nhiều thao tác tính toán độ đo khoảng cách DTW. Thời gian của giải thuật EP-M trên bộ dữ liệu chuỗi thời gian điện tâm đồ Koski-ECG gồm 30.000 điểm dữ liệu sẽ tốn khoảng 1 giờ 46 phút. Do vậy, giải thuật EP-M cần phải được cải tiến hơn nữa về độ hữu hiệu tính toán.

Đã có nhiều kỹ thuật phần mềm để tăng tốc việc tính toán độ đo DTW trong bài toán tìm kiếm tương tự trên chuỗi thời gian, như từ bỏ sớm, cắt tỉa dựa vào cận dưới (lower-bound based pruning) và lập chỉ mục (indexing). Tuy nhiên chỉ dựa vào các giải pháp phần mềm thì vẫn chưa thể khắc phục được chi phí tính toán cao của độ đo

* Coressponding Author

khoảng cách DTW, nhất là trong tình huống làm việc với các bộ dữ liệu chuỗi thời gian kích thước lớn. Do đó, chúng ta nên kết hợp các kỹ thuật phần mềm tốt nhất với các giải pháp dựa vào phần cứng để cải tiến hơn nữa tính hữu hiệu của các giải thuật hiện hành để đối phó với các tình huống dữ liệu lớn trong thời đại hiện nay.

Gần đây, với sự phát triển của Đơn vị xử lý đồ họa (Graphical Processing Units - GPU) công dụng tổng quát và môi trường lập trình GPU, nhiều nghiên cứu đã triển khai để tìm cách thực hiện các công tác khai phá dữ liệu trên chuỗi thời gian dựa vào GPU ([6], [7], [8], [9]). Sart và các cộng sự, năm 2010 [6] lần đầu tiên đề xuất một phương pháp dựa vào GPU tăng tốc tìm kiếm tương tự trên chuỗi thời gian dạng luồng với độ đo DTW. Chang và các cộng sự, năm 2012 [7] thiết kế một cách hiện thực dựa vào GPU cho giải thuật phát hiện shapelet để phân lớp chuỗi thời gian. Zhu và các cộng sự, năm 2016 [8] đề xuất một giải thuật dựa vào GPU, được gọi là GPU-STOMP, mà dựa vào thông tin ma trận khoảng cách (distance matrix profile) để tăng tốc quá trình phát hiện motif trên chuỗi thời gian có kích thước lớn. Gần đây nhất, Zhu và các cộng sự, năm 2021 [9] đề xuất một khung thức dựa vào GPU để tăng tốc công tác phát hiện motif và bất thường trên chuỗi thời gian. Trong công trình này, Zhu và các cộng sự dựa trên giải thuật chân phương (brute-force) để phát hiện chuỗi con bất thường nhất và chia giải thuật này thành năm phần để xem xét áp dụng song song hóa với GPU. Năm phần đó là: song song hóa dựa trên chuỗi con, tính độ đo Euclid, tính độ đo Euclid chuẩn hóa, tính độ đo DTW và phát hiện chuỗi con bất thường dùng kỹ thuật ngừng sớm (Early Stop). Tất cả các nghiên cứu nêu trên đã đạt được mức độ tăng tốc giải thuật từ hàng chục đến hàng trăm lần nhanh hơn.

Được gợi cảm hứng từ bốn công trình nêu trên sử dụng GPU để tăng tốc công tác khai phá dữ liệu chuỗi thời gian, trong nghiên cứu này, chúng tôi hướng đến thiết kế và đánh giá một giải pháp dựa vào phần cứng, cụ thể là GPU, để cải thiện độ hữu hiệu của giải thuật EP-M trong bài toán kết chuỗi con trên chuỗi thời gian. Ở đây, việc cải tiến độ hữu hiệu của giải thuật EP-M tập trung vào việc tăng tốc tìm kiếm tương tự với độ đo DTW trong giải thuật bằng cách song song hóa khâu tính toán độ đo khoảng cách DTW giữa các cặp chuỗi con dựa vào GPU.

Kết quả thực nghiệm trên năm bộ dữ liệu mẫu cho thấy trong bài toán kết chuỗi con bằng giải thuật EP-M, việc tính toán các độ đo khoảng cách DTW trên GPU kết hợp với dải Sakoe-Chiba và kỹ thuật cận dưới LB_Keogh [10] có thể thực thi nhanh gấp 13 lần so với giải thuật EP-M chạy trên CPU mà có sử dụng bộ kỹ thuật UCR-DTW (Rakthanmanon et al., 2012 [11]) để tăng tốc việc tính toán độ đo khoảng cách DTW.

Phần còn lại của bài báo này được cấu trúc như sau. Phần II giới thiệu các khái niệm căn bản và các công trình liên quan đến bài toán kết chuỗi con. Phần III mô tả phương pháp đề xuất song song hóa dựa vào GPU để tăng tốc quá trình kết chuỗi con trên chuỗi thời gian. Phần IV trình bày kết quả thực nghiệm để đánh giá phương pháp đề xuất. Và phần V trình bày các kết luận và hướng phát triển của nghiên cứu này.

II. CÁC KHÁI NIỆM VÀ CÔNG TRÌNH LIÊN QUAN

A. CÁC ĐỊNH NGHĨA

Một chuỗi thời gian $T = (t_1, t_2,..., t_n)$ là một tập có thứ tự gồm n trị số thực được đo đạc tại những điểm thời gian cách đều nhau. Một chuỗi con S với chiều dài k của T được ký hiệu như là $S = (t_i, t_{i+1}, ..., t_{i+k-1})$, với $1 \le i \le n - k+1$.

Cho chuỗi thời gian $X = (x_1, x_2,..., x_n)$ với chiều dài n và chuỗi thời gian $Y = (y_1, y_2,..., y_m)$ với chiều dài m, những chuỗi con tương tự của X và Y là những chuỗi con S_x (của X) và S_y (của Y) thỏa mãn điều kiện rằng khoảng cách giữa S_x và S_y nhỏ hơn một ngưỡng khoảng cách *th* cho trước. Trong nghiên cứu này, chúng ta sẽ tìm tất cả những cặp chuỗi con tương tự S_x và S_y từ hai chuỗi thời gian.

B. ĐỘ ĐO XOẮN THỜI GIAN ĐỘNG

Độ đo xoắn thời gian động (DTW) là độ đo khoảng cách mà ta có thể ánh xạ một điểm từ chuỗi thời gian này đến nhiều hơn một điểm trên chuỗi thời gian kia (xem Hình 1 (phía dưới)). Do đó, độ đo DTW cho phép tính khoảng cách giữa hai chuỗi thời gian có chiều dài khác nhau.

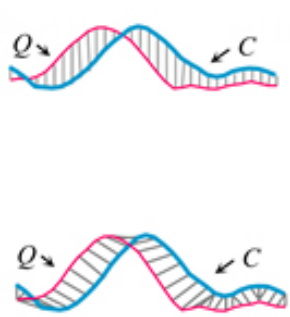
Giả sử chúng ta có hai chuỗi thời gian, chuỗi $Q = (q_1, q_2, ..., q_i, ..., q_n)$ và chuỗi $C = (c_1, c_2, ..., c_j, ..., c_m)$. Để tính độ đo khoảng cách DTW giữa hai chuỗi thời gian này, chúng ta tạo một ma trận $n \times m$, được gọi là *ma trận xoắn* (warping matrix) mà mỗi phần tử $D_{ij} = d(q_i , c_j)$ chứa khoảng cách giữa hai điểm q_i và c_j (tức là $d(q_i , c_j) = (q_i - c_j)^2$). Để tìm một cách ánh xạ tốt nhất giữa hai chuỗi thời gian, chúng ta tìm một lối đi xuyên qua ma trận mà cực tiểu hóa khoảng cách tích lũy toàn bộ giữa hai chuỗi thời gian. Một *lối đi xoắn* (warping path) W là một tập lưu các phần tử kế cận trong ma trận mà diễn tả một ánh xạ giữa Q và C. Lối đi này có thể được tìm thấy bằng cách dùng quy hoạch động để tính hệ thức truy hồi sau đây mà định nghĩa khoảng cách tích lũy $\gamma(i, j)$ bằng khoảng cách $d(i, j)$ tại ô hiện hành (i, j) và trị tối thiểu của các khoảng cách tích lũy tại ba ô kế cận:

$$\gamma(i, j) = d(q_i, c_j) + \min\{\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1)\}$$

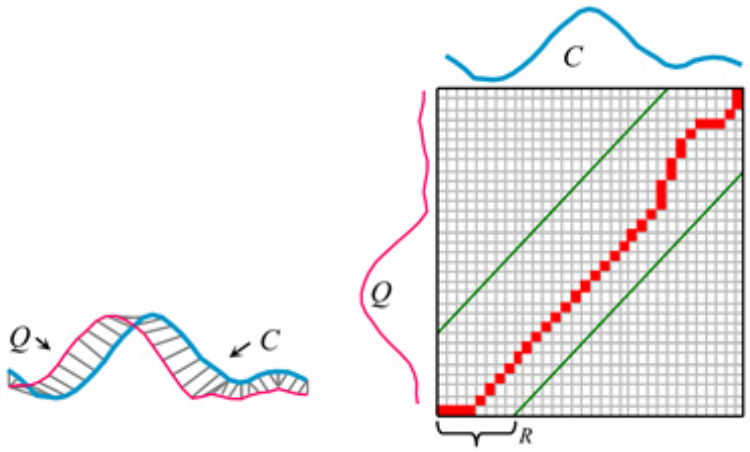
$$\gamma(0,0) = 0;\ \gamma(i,0) = \gamma(0, j) = \infty;\ i = 1, 2,..., n;\ j = 1,2,..., m$$

Như vậy, $\gamma(i, j)$ là tổng của $d(i, j) = (q_i - c_j)^2$ và trị tối thiểu của các khoảng cách tích lũy tại những ô lân cận với ô (i, j). Độ đo khoảng cách của hai chuỗi thời gian Q và C là căn hai của độ đo khoảng cách tích lũy tại ô (m, n).

Để tăng tốc việc tính khoảng cách DTW, chúng ta có thể giới hạn lối đi xoắn bằng cách hạn chế khoảng cách mà nó rời xa đường chéo của ma trận. Tập con của ma trận mà lối đi xoắn được phép đi qua được gọi là *cửa sổ xoắn* (warping window). Cách giới hạn lối đi xoắn thông dụng nhất là *dải Sakoe-Chiba* được đề nghị bởi Sakoe và Chiba, năm 1978 [11]. Dải Sakoe-Chiba band là một mảnh trong ma trận được xác định bởi hai đường thẳng song song với đường chéo chính (xem Hình 2).



Hình 1. *Hai chuỗi thời gian Q và C tương tự nhau về hình dạng nhưng lệch pha: dùng độ đo Euclid (phía trên) và độ đo DTW (phía dưới).*



Hình 2. *Dải Sakoe-Chiba với bề rộng R được dùng để giới hạn lối đi xoắn .*

Ngoài việc giới hạn lối đi xoắn, có cách khác để tăng tốc việc tính độ đo DTW là thay thế phần lớn những tính toán độ đo DTW tốn kém bằng việc tính toán các cận dưới (lower-bound) rất dễ thực hiện. Vài kỹ thuật cận dưới dễ thực hiện như cận dưới LB_Kim (Kim, 2002 [13]), và cận dưới LB_Keogh (Keogh và Ratanamahatana, 2005 [10]). Trong một kỹ thuật cận dưới của độ đo DTW, cận dưới của khoảng cách DTW giữa hai chuỗi thời gian Q và C, được ký hiệu là LB(Q, C), phải thỏa mãn bất đẳng thức: LB(Q, C) ≤ DTW(Q, C).

Nếu LB(Q, C) > best_so_far, thì DTW(Q, C) > best_so_far và chúng ta không cần phải tính DTW(Q, C) và kết luận rằng hai chuỗi thời gian Q và C không thể tương tự nhau.

Ngoài ra, một phương pháp khác để tăng tốc việc tính độ đo DTW là sử dụng kỹ thuật từ bỏ sớm (early abandoning). Chúng ta có thể tính độ đo DTW từ trái sang phải và khi tính gia tăng từ 1 đến k, ta cộng thêm phần độ đo DTW chưa đầy đủ với phần đóng góp của cận dưới từ k+1 đến n. Tổng giá trị DTW(Q1:k, C1:k) + LB(Qk+1:m, Ck+1:n) sẽ là cận dưới của độ đo DTW thực sự (tức là DTW(Q1:m, C1:n)). Nếu tại bất kỳ thời điểm nào, cận dưới này lớn hơn giá trị khoảng cách best-so-far thì ta có thể dừng lại và bỏ qua chuỗi thời gian C này.

C. BỘ KỸ THUẬT UCR-DTW

Rakthanmanon và các cộng sự, năm 2013 [11] giới thiệu một bộ kỹ thuật có tên UCR-DTW, để tăng tốc tìm kiếm tương tự trên chuỗi thời gian kích thước lớn với độ đo DTW. Bộ kỹ thuật UCR-DTW sử dụng một tổ hợp gồm bốn

ý tưởng: chuẩn hóa có từ bỏ sớm, từ bỏ sớm có sắp thứ tự lại, đảo ngược vai trò giữa truy vấn và dữ liệu khi tính cận dưới, và vận dụng liên hoàn (cascading) hai kỹ thuật cận dưới LB_Kim và LB_Keogh.

D. PHÂN ĐOẠN CHUỖI THỜI GIAN DỰA VÀO CÁC ĐIỂM CỰC TRỊ QUAN TRỌNG

Giải thuật EP-M sử dụng một phương pháp phân đoạn chuỗi thời gian mà dựa vào các *điểm cực trị quan trọng* (major extreme point) được đề xuất bởi Fink and Gandhi, 2007 [5]. Bài báo [5] cho các định nghĩa về điểm cực tiểu, điểm cực tiểu quan trọng, điểm cực đại, điểm cực đại quan trọng và dùng chúng để nén chuỗi thời gian. Trong phương pháp điểm cực trị quan trọng, có một tham số mà người dùng phải xác định, đó là hệ số nén *R*. Hệ số nén càng lớn thì sẽ có ít điểm được rút trích ra từ chuỗi thời gian. Bắt đầu từ điểm đầu tiên của chuỗi thời gian *T*, chúng ta có thể nhận diện ra mọi điểm cực đại quan trọng và mọi điểm cực tiểu quan trọng trong chuỗi thời gian nhờ vào giải thuật All-Extrema, của Fink và Gandhi [5].

Trong giải thuật EP-M, sau khi tìm ra mọi điểm cực trị quan trọng trong một chuỗi thời gian, giải thuật có thể dùng những điểm cực trị quan trọng này để rút trích những phân đoạn (segment) từ chuỗi thời gian.

E. GIẢI THUẬT EP-M KẾT CHUỖI CON TRÊN CHUỖI THỜI GIAN VỚI ĐỘ ĐO DTW

Giải thuật EP-M, được đề xuất trong một nghiên cứu trước đây (Vĩnh và Anh, năm 2016 [4]), để kết chuỗi con trên chuỗi thời gian. Giải thuật này có tên EP-M (viết tắt từ Extreme Points and Matching). Giải thuật EP-M để kết chuỗi con trên chuỗi thời gian gồm hai bước chính sau đây.

Bước 1 (Phân đoạn): Giải thuật tìm tất cả những điểm cực trị quan trọng của hai chuỗi thời gian T_1 và T_2. Kết quả của công việc này là tìm được *m1* điểm cực trị quan trọng trên chuỗi thời gian T_1 tại các vị trí $ep1_1$, $ep1_2$,.., $ep1_{m1}$ và *m2* điểm cực trị quan trọng trên chuỗi thời gian T_2 tại các vị trí $ep2_1$, $ep2_2$,…, $ep2_{m2}$. Giải thuật dùng hai danh sách chứa vị trí những điểm cực trị quan trọng của T_1 và T_2, đó là *EP1* = ($ep1_1$, $ep1_2$,.., $ep1_{m1}$) và *EP2* = ($ep2_1$, $ep2_2$,…, $ep2_{m2}$). Trong bước tiếp theo, khi tạo ra những chuỗi con từ một chuỗi thời gian (T_1 hoặc T_2), giải thuật rút trích từng đoạn chuỗi con nằm giữa hai điểm cực trị quan trọng ep_i và ep_{i+2}.

Bước 2 (Kết chuỗi con): Giữ chuỗi thời gian T_1 cố định và ứng với mỗi chuỗi con *s* rút trích từ T_1, giải thuật tìm tất cả những chuỗi con khớp với nó từ chuỗi T_2 bằng cách dịch chuyển một cửa sổ trượt với chiều dài bằng với chiều dài của *s* dọc theo chuỗi T_2 mỗi lần dịch chuyển một điểm. Giải thuật lưu tất cả những chuỗi con trùng khớp vào tập kết quả S_1.

Giữ chuỗi thời gian T_2 cố định và ứng với mỗi chuỗi con *s* rút trích từ T_2, giải thuật tìm tất cả những chuỗi con khớp với nó từ chuỗi T_1 bằng cách dịch chuyển một cửa sổ trượt với chiều dài bằng với chiều dài của *s* dọc theo chuỗi T_1 mỗi lần dịch chuyển một điểm. Giải thuật lưu tất cả những chuỗi con trùng khớp vào tập kết quả S_2.

Trong bước 1, để giải thuật có thể nhận dạng mọi điểm cực trị quan trọng của chuỗi thời gian, chúng ta phải xác định thông số hệ số nén *R*. Theo Fink và Gandhi [5], các tác giả đề nghị *R* phải lớn hơn 1.

Mã giả mô tả Bước 2 của giải thuật EP-M được cho như trong **Giải thuật 1**.

Giải thuật 1: Subsequence_Join (T_1[1.. n_1], T_2[1.. n_2], *th*)

Input*:* T_1 và T_2 là hai chuỗi thời gian đã được phân đoạn;

th là ngưỡng khoảng cách được cho.

Output*:* Hai tập S_1 và S_2 chứa những cặp chuỗi con trùng khớp tìm thấy

1. **for** mỗi chuỗi con *s* rút trích từ T_1 trong Bước 1 **do**

   tìm tất cả những chuỗi con trong T_2 tương tự với *s* bằng cách gọi trình con Subsequence_Matching (*s*, T_2, *th*),

   endfor

   Lưu tất cả những cặp chuỗi con kết quả trong tập S_1

2. **for** mỗi chuỗi con *s* rút trích từ T_2 trong Bước 1 **do**

   tìm tất cả những chuỗi con trong T_1 tương tự với *s* bằng cách gọi trình con Subsequence_Matching (*s*, T_1, *th*),

   endfor

   Lưu tất cả những cặp chuỗi con kết quả trong tập S_2

Procedure Subsequence_Matching(*s*[1.. *m*], *T*[1.. *n*], *th*)

Input: *T* là một chuỗi thời gian có chiều dài *n*, *s* là một chuỗi con có chiều dài *m* ($m \le n$)

và *th* là một ngưỡng khoảng cách

Output: *S* là tập chứa những chuỗi con khớp với *s* trong *T*.

1 **for** *i* = 1 **to** *n* – *m* + 1 **do**

2 *segment_of_T* = chuỗi con $T_{i, i+m-1}$

3 *dtw_distance* = DTW_EA(*s*, *segment_of_T*)

4 **if** (*dtw_distance* <= *th*) **then**

5 lưu cặp <*s*, *segment_of_T*> vào tập kết quả *S*

6 **endfor**

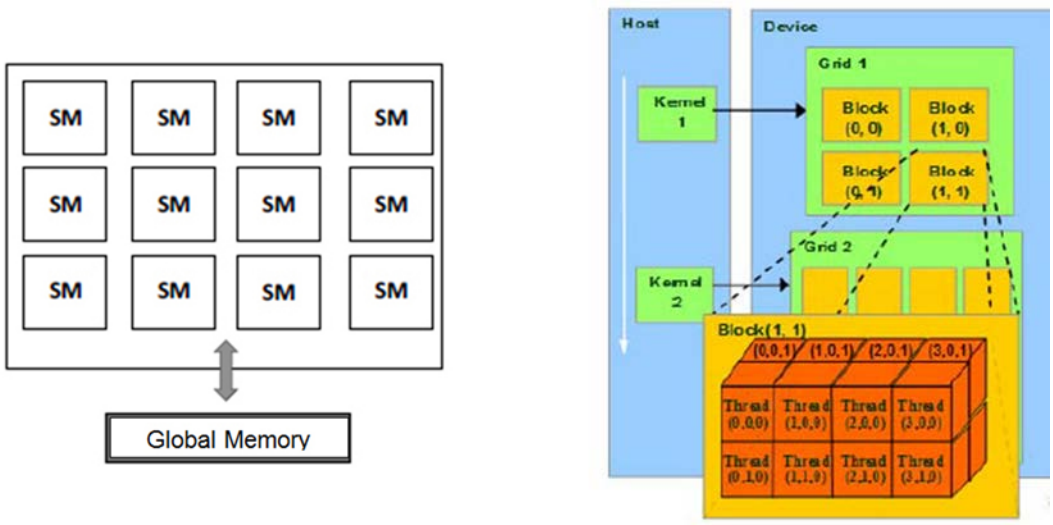
end Procedure

Chú ý rằng trong giải thuật EP-M, khi tính độ đo khoảng cách DTW, giải thuật dùng cách tính khoảng cách DTW trực tiếp có kết hợp với kỹ thuật *từ bỏ sớm* được đề xuất bởi Li và Wang (2009) [14] mà được gọi là DTW_EA trong câu lệnh 3 của thủ tục *Subsequence_Matching*. Để sự so trùng có ý nghĩa giữa hai chuỗi thời gian, cả hai chuỗi phải được chuẩn hóa. Đối với giải thuật EP-M, trước khi áp dụng giải thuật EP-M, chúng tôi dùng phép *chuẩn hóa cực tiểu-cực đại* (min-max normalization) để biến đổi tầm vực (scale) của một chuỗi thời gian về tầm vực của một chuỗi thời gian khác dựa vào các trị nhỏ nhất và lớn nhất của hai chuỗi thời gian.

III. TĂNG TỐC KẾT CHUỖI CON TRÊN CHUỖI THỜI GIAN DỰA VÀO GPU

A. GPU

GPU là phương tiện phần cứng rất phù hợp để giải quyết một số vấn đề bằng cách thực hiện chúng thông qua tính toán song song về mặt dữ liệu ([15], [16]). Đóng vai trò chủ (host) đối với thiết bị GPU, CPU tổ chức và khởi động các *hàm hạt nhân* (kernel function) làm việc trên thiết bị GPU. Một thiết bị GPU bao gồm một số *bộ đa xử lý luồng* (Streaming Multiprocessor - SM), mà mỗi SM gồm các bộ máy xử lý đơn giản được gọi là *lõi CUDA* (CUDA core) trong thuật ngữ của NVIDIA. Mỗi SM có một *bộ nhớ dùng chung* (shared memory) mà có thể được truy cập một cách bình đẳng bởi mọi lõi CUDA trong cùng SM. Tại một chu kỳ bất kỳ, các lõi CUDA trong một SM thực thi cùng một câu lệnh trên nhiều mục dữ liệu khác nhau. Các SM trao đổi thông tin với nhau thông qua *bộ nhớ toàn cục* (global memory) của GPU (Xem Hình 3.a).



Hình 3. *(a) Kiến trúc phần cứng CUDA* *(b) Mô hình lập trình CUDA*

Về phương diện lập trình, mô hình CUDA bao gồm một tập hợp *luồng* (thread) thực thi song song. Một *warp* là một tập hợp luồng có thể thực thi đồng thời trên một SM. Kích thước của một warp là cố định đối với một GPU cụ thể. Lập trình viên phải quyết định về số *khối* (block) và số luồng được thực thi. Nếu số khối và số luồng vượt qua kích thước của warp, chúng sẽ được chia xẻ thời gian với nhau một cách nội bộ trong SM. Tập hợp luồng mà làm thành một khối sẽ thực thi trên một bộ đa xử lý tại một lúc nào đó. Một *khối luồng* (thread block) được định nghĩa như là một lô (batch) luồng mà được đảm bảo có thể thực thi đồng thời và hợp tác với nhau thông qua các tài nguyên dùng chung. Nhiều khối có thể được gán vào một bộ đa xử lý và sự thực thi của chúng có thể chia xẻ thời gian với nhau. Một thực thi đơn tạo ra một số khối. Tập hợp tất cả các khối của một thực thi đơn được gọi là

một *lưới* (grid) (xem Hình 3.b). Tất cả các luồng của một khối mà thực thi trên cùng một bộ đa xử lý đơn sẽ dùng chung tài nguyên của bộ đa xử lý đó một cách bình đẳng với nhau. Mỗi luồng và mỗi khối được cho một định danh (ID) để có thể truy đạt đến trong quá trình thực thi của luồng. Mỗi luồng thực thi một tập lệnh được gọi là *hạt nhân* (kernel). Các chi tiết sâu hơn về GPU, độc giả quan tâm có thể tham khảo tài liệu [15].

B. TĂNG TỐC KẾT CHUỖI CON TRÊN CHUỖI THỜI GIAN DỰA VÀO GPU

Việc sử dụng cách tiếp cận GPU là hướng đến tính toán song song các độ đo khoảng cách DTW để giảm thiểu chi phí tính toán của kết chuỗi con trên chuỗi thời gian. Ở Bước 2 của giải thuật EP-M, sau khi chuỗi thời gian T_1 được phân đoạn thành những chuỗi con, với mỗi chuỗi con s trong T_1 chúng ta tìm kiếm tất cả những chuỗi con trùng khớp với nó bằng cách dịch chuyển một cửa sổ trượt với chiều dài bằng chiều dài của s dọc theo chuỗi T_2 mỗi lần dịch một điểm (Hình 4). Công việc tìm kiếm những cặp chuỗi con tương tự với độ đo DTW này bao gồm những tác vụ cơ bản là *tính độ đo DTW của mỗi cặp chuỗi con* và những tác vụ cơ bản này có thể được song song hóa nhờ vào GPU. Tiếp theo, công việc tìm kiếm chuỗi con tương tự lại được thực hiện trên chuỗi T_1 với chuỗi T_2 đóng vai trò là chuỗi cố định ở vòng lặp ngoài và công việc tìm kiếm những cặp chuỗi con tương tự với độ đo DTW cũng được song song hóa nhờ vào GPU. Như vậy, **Giải thuật 1** (xem mục II.E) tương ứng với bước 2 trong giải thuật EP-M được cải biên thành phiên bản song song hóa **Giải thuật 2** với tên gọi *Subsequence_Join_GPU*. Giải thuật 2 gọi thủ tục *Build_Headlist*() để lập danh sách những chuỗi con trên chuỗi thời gian đang xét sẽ phải tính độ đo DTW trực tiếp đến chuỗi con truy vấn *s*.

Giải thuật 2: Subsequence_Join_GPU (T_1[1.. n_1], T_2[1.. n_2], *th*)

Input*:* T_1 và T_2 là hai chuỗi thời gian đã được phân đoạn;

th là ngưỡng khoảng cách được cho.

Output*:* Hai tập S_1 và S_2 chứa những cặp chuỗi con trùng khớp tìm thấy

1. **for** mỗi chuỗi con *s* rút trích từ T_1 trong Bước 1 **do**

   Build_Headlist(*s*, T_2, *th, headlist2*)

   Xem xét tất cả chuỗi con trong *headlist2* nếu có độ đo khoảng cách DTW đến *s* nhỏ hơn *th* sẽ được đưa vào tập kết quả T_1 // quá trình này sẽ được song song hóa trên GPU

   endfor

2. **for** mỗi chuỗi con *s* rút trích từ T_2 trong Bước 1 **do**

   Build_Headlist(*s*, T_1, *th, headlist1*)

   Xem xét tất cả chuỗi con trong *headlist1* nếu có độ đo khoảng cách DTW đến s nhỏ hơn *th* sẽ được đưa vào tập kết quả T_2 // quá trình này sẽ được song song hóa trên GPU

   endfor

Procedure Build_Headlist(*s*[1.. *m*], *T*[1.. *n*], *th*, *headlist*)

Input: *T* là một chuỗi thời gian có chiều dài *n*, *s* là một chuỗi con có chiều dài *m* ($m \le n$)

và *th* là một ngưỡng khoảng cách

Output: *headlist* là danh sách chứa những chuỗi con trong *T* sẽ phải tính độ đo DTW trực tiếp đến chuỗi con *s*

1 **for** *i* = 1 **to** *n* – *m* + 1 **do**

2 *segment_of_T* = chuỗi con $T_{i,\ i+m-1}$

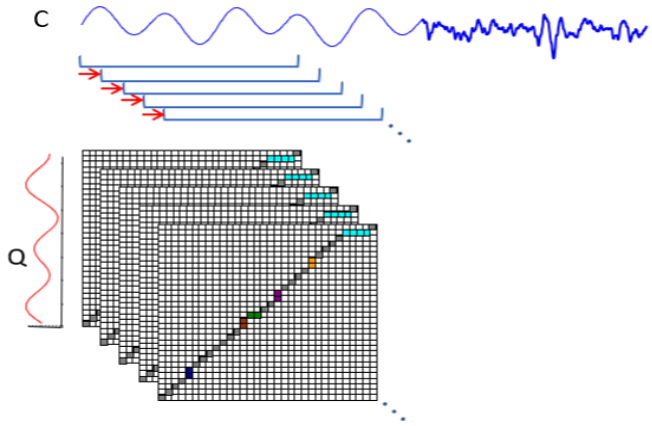
3 **if** (LB_Keogh(*segment_of_T*, *s*) <= *th*) **then**

4 đưa vị trí của *segment_of_T* vào danh sách *headlist*

5 **endfor**

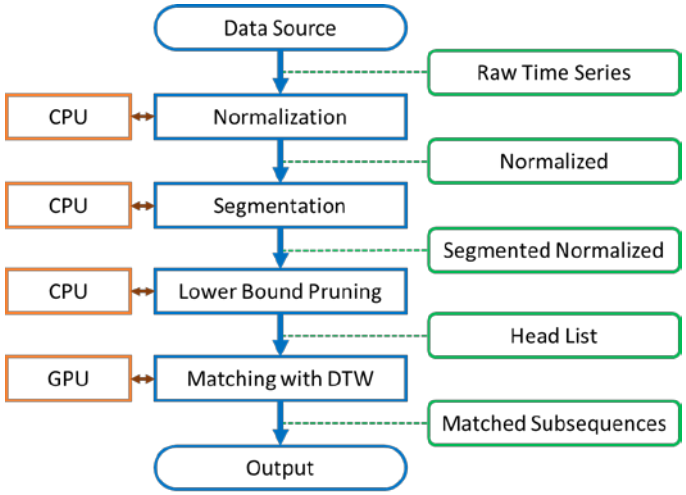
end Procedure

[1]
[2]



Hình 4. *So trùng chuỗi con trong giải thuật EP-M dựa vào GPU*

Kiến trúc *Một Câu Lệnh Nhiều Dữ Liệu* (Single Instruction Multiple Data - SIMD) cho phép chúng ta triển khai mỗi lượt tính độ đo khoảng cách DTW được thực thi song song trên nhiều phân đoạn khác nhau (dưới cửa sổ trượt) của một chuỗi thời gian. Cách hiện thực hóa tính toán độ đo khoảng cách DTW một cách song song trên GPU được thực hiện theo hai phiên bản: DTW không sử dụng kỹ thuật cận dưới LB_Keogh và DTW có sử dụng kỹ thuật cận dưới LB_Keogh. Hình 5 minh họa tiến trình của cách hiện thực phiên bản của giải thuật EP-M dựa vào GPU trong đó độ đo DTW có sử dụng kỹ thuật cận dưới LB_Keogh.

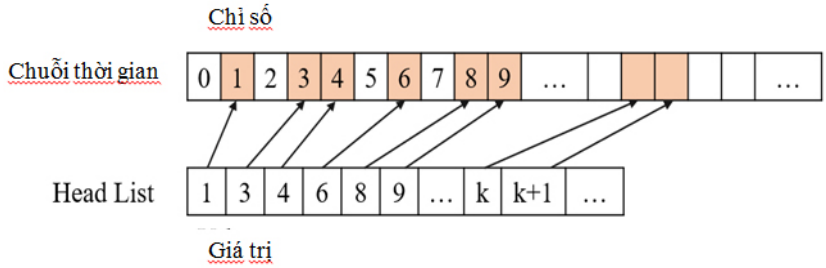


Hình 5. *Hiện thực phiên bản của giải thuật EP-M dựa vào GPU tính toán độ đo DTW có sử dụng kỹ thuật cận dưới.*

Đối với phiên bản của giải thuật EP-M tính toán độ đo DTW có sử dụng cận dưới LB_Keogh, trong khi so trùng chuỗi con, nếu cận dưới của chuỗi con truy vấn với một chuỗi con nào đó lớn hơn ngưỡng khoảng cách *th*, chúng ta có thể bỏ qua việc tính toán khoảng cách DTW của cặp chuỗi con này. Sau khi phân đoạn chuỗi thời gian T_1, với mỗi chuỗi con s của T_1, chúng ta phải so s với tất cả các chuỗi con rút trích từ chuỗi T_2 dựa vào một cửa sổ trượt, nếu bất kỳ chuỗi con s_2 nào trong các chuỗi con này thỏa điều kiện cận dưới ($LB(s, s_2) \leq th$), thì chúng ta sẽ lưu vị trí của s_2 vào danh sách có tên *headlist*. Công việc này được thể hiện ở thủ tục *Build_Headlist*().

Hình 6 minh họa vai trò của danh sách *headlist* dùng để lưu vị trí của tất cả những chuỗi con mà chúng ta phải tính độ đo khoảng cách DTW giữa nó với chuỗi con truy vấn.

Rõ ràng là với chuỗi con truy vấn q có chiều dài m và chuỗi thời gian T có chiều dài n, thì tổng số phần tử trong *headList* thường nhỏ hơn $n - m + 1$ khá nhiều khi ta áp dụng phiên bản tính độ đo DTW với kỹ thuật cận dưới. Trong trường hợp chúng ta áp dụng phiên bản tính độ đo DTW không dùng kỹ thuật cận dưới, thì tổng số phần tử trong *headList* bằng đúng $n - m + 1$. Chú ý rằng chiều dài của *headlist* sẽ ảnh hưởng đến tính hữu hiệu của cách hiện thực dựa vào GPU của giải thuật EP-M. Điều này hàm ý rằng việc có sử dụng kỹ thuật cận dưới hay không trong giải pháp phần mềm của giải thuật EP-M sẽ ảnh hưởng đến hiệu năng của việc vận dụng giải pháp phần cứng dựa vào GPU cho giải thuật này.



Hình 6. *Headlist như là một danh sách chứa vị trí của các chuỗi con*

Công việc so trùng chuỗi con trong giải thuật EP-M (tức là trong các vòng lặp trong của Bước 2 trong giải thuật EP-M) có thể được thực thi song song trên GPU theo cách thức như sau. Mỗi luồng (thread) truy đạt đến một phân đoạn kế tiếp nhau với chiều dài cửa sổ trượt (tương ứng với một phần tử trong *headlist*) để kiểm tra xem nó có tương tự với chuỗi con truy vấn hay không. Nói một cách khác, mỗi luồng của GPU phải tính một độ đo khoảng cách DTW thực sự giữa một cặp chuỗi con. Điều này có nghĩa là mỗi luồng của GPU phải làm việc trên một ma trận xoắn được định nghĩa bởi một cặp chuỗi con như minh họa ở Hình 2. Do đó, số luồng được sử dụng trong công việc này bằng với số phần tử trong *headlist*, mà length(*headlist*) ≤ $n - m + 1$.

Khác với cách tính toán tuần tự trên CPU, thiết bị GPU có thể làm việc với bộ nhớ dùng chung của GPU. Do đó sự hiện thực hóa dựa vào GPU của công tác tính toán độ đo DTW bao gồm ba bước chính: (i) CPU chép dữ liệu sang bộ nhớ của GPU, (ii) CPU gọi hạt nhân (kernel) GPU để khởi động việc tính toán trên GPU và (iii) CPU chép dữ liệu kết qủa từ GPU về CPU.

Trong bước thứ nhất của ba bước nêu trên, CPU chép nguyên chuỗi thời gian sang bộ nhớ GPU. Nếu chuỗi thời gian này dài hơn bộ nhớ của GPU, CPU tách chuỗi này thành nhiều lô và xử lý mỗi lần một lô. Trong bước thứ hai, CPU gọi hạt nhân tại GPU. Mỗi luồng hạt nhân làm việc với một chuỗi con cụ thể tại cửa sổ trượt để tính toán độ đo khoảng cách DTW thực sự giữa chuỗi con này với chuỗi con truy vấn (xem Hình 4). Cuối cùng tại bước thứ ba, khi tất cả các luồng kết thúc, CPU chép tất cả các khoảng cách DTW được tính mà được lưu tại một mảng trong bộ nhớ của GPU về bộ nhớ của CPU.

Khi bắt đầu một lần tính khoảng cách DTW, các luồng hạt nhân chép một chuỗi con truy vấn cụ thể từ bộ nhớ toàn cục sang bộ nhớ dùng chung (shared memory) của GPU. Cấu trúc dữ liệu nội tại để tính toán độ đo DTW không nhất thiết phải là một ma trận $m \times m$ mà là một mảng gồm hai vector cột có kích thước m. Điều này là do khi áp dụng quy hoạch động để tính độ đo DTW giữa hai chuỗi con có cùng độ dài, mỗi lượt lặp của vòng lặp ngoài của giải thuật chỉ liên quan đến hai cột kế cận nhau trong ma trận xoắn. Như vậy, nhờ vào sự lệ thuộc dữ liệu giữa hai cột lân cận, chúng ta chỉ cần dùng một mảng gồm hai vector cột là đủ để hỗ trợ cho việc tính toán độ đo khoảng cách DTW trực tiếp. Hai vector cột này thích hợp để được lưu tại bộ nhớ dùng chung khi chiều dài chuỗi con m thường nhỏ. Với tất cả những dữ liệu đã có tại chỗ như vậy, việc tính toán độ đo DTW sẽ được thực hiện. Mỗi luồng sẽ lưu khoảng cách DTW được tính vào một mảng toàn cục mà được đánh chỉ số theo ID của luồng.

IV. THỰC NGHIỆM

Chúng tôi thực nghiệm các phương pháp đối sánh trên năm bộ dữ liệu mẫu để so sánh hiệu năng của các phương pháp về phương diện độ hữu hiệu thời gian. Các phương pháp đối sánh được lập trình bằng ngôn ngữ C# và CUDA với môi trường Microsoft Visual Studio 2017. Các thực nghiệm đối với các phiên bản song song được tiến hành trên nền tảng GPU (NVIDIA TESLA P100 PCIE 16GB). CPU được dùng có cấu hình Intel® Xeon® Silver 4114, 40 cores, 2.2GHz, RAM 200 GB. Một vài thông số của GPU TESLA P100 được cho như sau. Số bộ đa xử lý Streaming Multiprocessors (SMs) là 56; số lõi CUDA là 3584; kích thước bộ nhớ là 16 GB; và băng thông bộ nhớ là 720 GB/sec.

A. CÁC BỘ DỮ LIỆU MẪU VÀ CÁC GIÁ TRỊ THAM SỐ

Các thực nghiệm được tiến hành trên năm bộ dữ liệu mẫu tải về từ thư viện dữ liệu chuỗi thời gian của trường Đại Học California Riverside (The UCR Time Series Data Mining Archive [17]). Tên gọi và chiều dài của từng bộ dữ liệu mẫu được liệt kê như sau. Power (35450 điểm dữ liệu), Koski-ECG (144002 điểm dữ liệu), Chromosome (999541 điểm dữ liệu), Stock (2119415 điểm dữ liệu), EEG (10957312 điểm dữ liệu). Những bộ dữ liệu này từ những lãnh vực ứng dụng khác nhau: y khoa, tài chính và công nghiệp.

Vì kết chuỗi con phải làm việc trên hai chuỗi thời gian có một mức độ tương tự với nhau, cũng giống như ở nghiên cứu [4], trong nghiên cứu này, chúng tôi cần một phương pháp để tạo ra một bộ dữ liệu tổng hợp T_2 từ bộ dữ liệu chuỗi thời gian T_1. Bộ dữ liệu tổng hợp được tạo ra bằng cách áp dụng qui tắc sau đây:

$$x_i = x_{i-1} \pm |x_{i-1} - \varepsilon| \qquad \text{với} \quad \varepsilon = \frac{\sum_{i=1}^{6} x_i}{6}$$

Trong công thức trên, + hay – được chọn một cách ngẫu nhiên. Bộ dữ liệu tổng hợp được tạo ra sau khi bộ dữ liệu tương ứng đã được chuẩn hóa.

Trong nghiên cứu này, chúng tôi áp dụng heuristic sau đây để xác định giá trị cho hệ số nén *R*, một thông số cho phương pháp điểm cực trị quan trọng. Cho *SD* là độ lệch chuẩn của chuỗi thời gian. Chúng tôi chọn *R* bằng 1.5 * *SD*. Như vậy *R* thay đổi theo từng bộ dữ liệu. Bảng 1 thống kê chiều dài của phân đoạn đầu tiên được rút trích từ mỗi chuỗi thời gian mẫu.

Bảng 1. *Chiều dài của phân đoạn đầu tiên được rút trích từ mỗi chuỗi thời gian mẫu*

Bộ dữ liệu	Power	ECG	Chromosome	Stock	EEG
Chiều dài chuỗi con	201	101	51	11	201

Đối với thông số ngưỡng khoảng cách *th* trong giải thuật EP-M, chúng tôi chọn trị cho thông số này tùy thuộc vào đặc điểm của từng bộ dữ liệu mẫu và chọn thông số này thông qua thực nghiệm.

B. KẾT QUẢ THỰC NGHIỆM

Trong các thực nghiệm, chúng tôi so sánh thời gian thực thi của năm phiên bản hiện thực giải thuật EP-M: một phiên bản tuần tự và bốn phiên bản song song dựa vào GPU với các tên gọi như sau:

- UCR_DTW (phiên bản tuần tự của EP-M có sử dụng bộ kỹ thuật UCR-DTW khi tính độ đo DTW)
- pDTW (phiên bản song song dựa vào GPU của EP-M sử dụng cách tính độ đo DTW trực tiếp)
- pDTW_SC (phiên bản song song dựa vào GPU của EP-M có kết hợp dải Sakoe-Chiba khi tính độ đo DTW)
- pDTW_LB (phiên bản song song dựa vào GPU của EP-M có kết hợp kỹ thuật cận dưới LB_Keogh khi tính độ đo DTW)
- pDTW_LBSC (phiên bản song song dựa vào GPU của EP-M có kết hợp dải Sakoe-Chiba và kỹ thuật cận dưới LB_Keogh khi tính độ đo DTW).

Chú ý rằng UCR_DTW là phiên bản tuần tự của EP-M mà sử dụng các giải pháp phần mềm tốt nhất khi tính độ đo DTW và pDTW_LBSC là phiên bản song song của EP-M mà kết hợp kỹ thuật tăng tốc dựa vào GPU với hai kỹ thuật phần mềm rất hữu hiệu khi tính toán độ đo DTW: dải Sakoe-Chiba và kỹ thuật cận dưới LB_Keogh. Có hai phiên bản song song của EP-M mà có sử dụng kỹ thuật cận dưới LB_Keogh là pDTW_LB và pDTW_LBSC.

Trong các thực nghiệm, chúng tôi tập trung vào việc đo đạc thời gian thực thi của năm phiên bản của giải thuật EP-M trên năm bộ dữ liệu mẫu.

Thời gian thực thi thay đổi theo từng bộ dữ liệu, chiều dài của từng bộ dữ liệu và từng phiên bản của giải thuật EP-M. Bảng 2 cho thấy thời gian thực thi là một hàm của chiều dài của bộ dữ liệu. Từ kết quả thực nghiệm trong Bảng 2 chúng ta có thể thấy pDTW_LB_SC là phương pháp hữu hiệu nhất trong bốn phương pháp dựa vào GPU để tăng tốc quá trình kết chuỗi con trên chuỗi thời gian. Hiệu năng cao của phiên bản pDTW_LBSC cho thấy sự kết hợp giữa kỹ thuật tăng tốc dựa vào phần cứng với các giải pháp phần mềm được ưa chuộng có thể đem lại độ hữu hiệu về thời gian cao hơn là chỉ sử dụng kỹ thuật tăng tốc dựa vào phần cứng.

Bảng 2. *Thời gian thực thi (tính bằng phút) của bốn phiên bản song song hóa của EP-M dựa vào GPU*

Bộ dữ liệu	Chiều dài	Phương pháp			
		pDTW	pDTW_SC	pDTW_LB	pDTW_LBSC
Power	10,000	0.332	0.317	0.326	0.312
	25,000	5.142	4.770	3.645	3.412
	35,450	11.906	11.066	8.405	7.849
Ecg	10,000	0.380	0.363	0.232	0.223
	25,000	5.644	5.242	2.065	1.943
	50,000	29.142	27.157	10.033	9.709
Chromosome	10,000	0.186	0.180	0.132	0.130
	25,000	2.477	2.312	0.436	0.423
	50,000	12.737	11.908	1.088	1.074
Stock	10,000	0.224	0.216	0.143	0.140
	25,000	3.480	3.241	0.952	0.918
	50,000	18.480	17.246	1.788	1.737
Eeg	10,000	0.133	0.131	0.163	0.157
	25,000	1.237	1.172	0.869	0.842
	50,000	6.144	5.809	3.531	3.432

Bảng 3 thống kê *hệ số tăng tốc* (speed-up rate) của phiên bản song song pDTW_LBSC so với phiên bản tuần tự UCR_DTW trên năm bộ dữ liệu. Từ kết quả thực nghiệm trong Bảng 3, chúng ta có thể thấy trung bình, phiên bản song song pDTW_LBSC thực thi nhanh gấp **13** lần so với phiên bản tuần tự UCR_DTW.

V. KẾT LUẬN

Trong nghiên cứu trước đây [4], giải thuật EP-M kết chuỗi con trên chuỗi thời gian với độ đo DTW đã được đề xuất. Mặc dù giải thuật EP-M tương đối hữu hiệu, giải thuật này vẫn cần được cải tiến hơn nữa về độ hữu hiệu tính toán để nó có thể thích ứng với những trường hợp làm việc với chuỗi thời gian khá lớn. Bài báo này trình bày cách tiếp cận dựa vào GPU để tăng tốc việc tính toán độ đo khoảng cách DTW và nhờ đó giảm thiểu được tổng chi phí thực thi khi kết chuỗi con trên chuỗi thời gian bằng giải thuật EP-M. Trong giải thuật song song dựa vào GPU này, chúng tôi phân bổ một luồng (thread) phụ trách việc tính toán độ đo DTW giữa mỗi cặp chuỗi con trong quá trình so trùng chuỗi con.

Bảng 3. *Thời gian thực thi (tính bằng phút) của phiên bảng tuần tự có sử dụng UCR-DTW và phiên bản song song pDTW_LBSC cùng với độ tăng tốc giữa hai phiên bản trên mỗi bộ dữ liệu*

Bộ dữ liệu	Chiều dài	Phương pháp		
		UCR_DTWW	pDTW_LBSC	Độ tăng tốc
Power	10,000	6.975	0.312	22.355
	25,000	42.605	3.412	12.486
	35,450	83.551	7.849	10.644
Ecg	10,000	4.373	0.223	19.609
	25,000	28.378	1.943	14.673
	50,000	94.195	9.709	9.698
Chromosome	10,000	1.588	0.130	12.125
	25,000	4.855	0.423	11.477
	50,000	9.828	1.074	9.150
Stock	10,000	1.74	0.140	12.428
	25,000	14.923	0.918	16.255
	50,000	24.719	1.737	14.230
Eeg	10,000	1.476	0.157	9.401
	25,000	9.198	0.842	10.923
	50,000	29.874	3.432	8.704

Kết quả thực nghiệm với năm bộ dữ liệu mẫu cho thấy việc tính toán độ đo khoảng cách DTW có sử dụng dải Sakoe-Chiba và kỹ thuật cận dưới LB_Keogh trong giải thuật EP-M trên nền tảng GPU đem lại một sự tăng tốc nhanh lên gấp 13 lần so với phiên bản EP-M trên CPU mà có sử dụng bộ kỹ thuật UCR-DTW khi tính toán độ đo DTW. Những kết quả thực nghiệm này hàm ý rằng chúng ta nên kết hợp những kỹ thuật phần mềm được ưa chuộng với kỹ thuật song song hóa dựa vào GPU để cải thiện độ hữu hiệu tính toán của phương pháp kết chuỗi con này.

Trong tương lai, chúng tôi dự định hiện thực giải thuật EP-M trên môi trường song song và phân tán như Spark hoặc MapReduce ([18], [19]) để xem giải thuật này có thể được cải tiến nhiều hơn về độ hữu hiệu khi so sánh với cách song song hóa dựa vào GPU hay không.

VI. TÀI LIỆU THAM KHẢO

[1] Y. Lin, M. D. McCool, and A. Ghorbani (2010), Time Series Motif Discovery and Anomaly Detection Based on Subseries Join, IAENG *Int. Journal of Computer Science*, vol. 37, no. 3, pp.1-13.

[2] Y. Lin, M. D. McCool (2010), Subseries Join: A Similarity-based Time series Match Approach, *Proc. of PAKDD 2010*, Part I, LNAI 6118, pp. 238-245.

[3] A. Mueen, H. Hamooni and T. Estrada (2014), Time Series Join on Subsequence Correlation, *Proceedings of ICDM 2014*, pp. 450-459.

[4] V.D.Vinh, D.T. Anh (2016), Efficient Subsequence Join Over Time Series Under Dynamic Time Warping, In: *Recent Developments in Intelligent Information and Database Systems*, Studies in Computational Intelligence, D. Krol, L. Madeyski, N. T. Nguyen (Eds.), Vol. 642, Springer, pp. 41-52.

[5] E. Fink and H. S. Gandhi (2011), Compression of Time Series by Extracting Major Extrema, *Journal of Experimental & Theoretical Artificial Intelligence,* vol 23, no. 2, pp. 255-270.

[6] D. Sart, A. Mueen, W. Najjar, E. Keogh and V. Niennattrakul (2010), Accelerating Dynamic Time Warping SubsequenceSearch with GPUs and FPGAs, *International Conference on Data Mining*, pp. 1001-1006.

[7] K. W. Chang, B. Deka, W.W. Hwu, D. Roth (2012), Efficient Pattern-based Time Series Classification on GPU, *Proc. of IEEE 12th International Conference on Data Mining*, pp.131-140.
[8] Y. Zhu, Z. Zimmerman, N. S. Senobari, C. M. Yen, G. Funning, A. Mueen, P. Brisk, E. Keogh (2016), Matrix Profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins, *Proc. of IEEE 16 th Int. Conf. on Data Mining*, pp. 739-748.
[9] B. Zhu, Y. Jiang, Y. Deng (2021), A GPU acceleration framework for motif and discord based pattern mining, IEEE Trans. on Parallel and Distributed Systems, vol.32, no. 8, August 2021, pp. 1987-2004.
[10] E. Keogh, C.A. Ratanamahatana (2005), Exact Indexing of Dynamic Time Warping, *Knowledge and information systems*, vol. 7, no.3, 2005, pp. 358-386.
[11] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria and E. Keogh (2012), Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping, *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 262-270.
[12] H. Sakoe and S. Chiba (1978), Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol 26, no. 1, pp. 43-49.
[13] S. W. Kim (2001), An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases, *Proc. of 17th International Conference on Data Engineering*, pp. 607-614.
[14] J. Li, Y. Wang (2009), Early abandon to accelerate exact dynamic time warping, *Int. Arab J. Inf. Techno*l., vol 6, no. 3, pp. 144–152.
[15] NVIDIA (2017), CUDA Programming Guide Version 8.0, https://docs.nvidia.com/cuda/index.html
[16] NVIDIA (2017), CUDA Toolkit Documentation Version 8.0, https://docs.nvidia.com/cuda/index.html
[17] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen and G. Batista, "UCR Time Series Classification Archive", http://www.cs.ucr.edu/~eamonn/time_series_data/ (Accessed in 2018).
[18] K. Park and H. S. Won (2018), A study on subsequence similarity join in time series data using mapreduce, in: *Advances in Computer Science and Ubiquitous Computing*, Lecture Notes in Electrical Engineering 474, Springer, pp. 851-859.
[19] C. Rong, L. Chen, Y. N. Silva (2020), Parallel time series join using Spark, *Concurrency and Computation – Practice and Experience*, vol. 32, no. 9, May 2020.

ACCELERATING DYNAMIC TIME WARPING SUBSEQUENCE JOIN IN TIME SERIES WITH GPUS

Lam Truong An, Duong Tuan Anh

ABSTRACT— Subsequence join over time series is one of important time series data mining tasks. Here we utilize an efficient subsequence join method which is based on dynamic time warping (DTW) distance and segmentation. Given DTW's usefulness, there has been a large effect from research community to overcome its high computational cost. Proposed software speedup techniques for DTW computation in time series data mining include early abandoning, lower-bound based pruning and indexing. However, besides state-of-the art software solutions, we should turn to hardware-based solutions to improve further the efficiency of the current method. In this work we design and evaluate a hardware-based solution, Graphics Processing Unit (GPU), to accelerate our subsequence join method by applying parallelism. The experimental evaluation on several datasets shows the high efficiency of the proposed GPU-based method for time series subsequence join.

Keywords — time series, subsequence join, dynamic time warping, segmentation, GPU.

Lâm Trường An tốt nghiệp cử nhân ngành công nghệ thông tin năm 2012 tại trường Đại học Cần Thơ và thạc sĩ ngành Khoa học máy tính tại trường Đại học Bách Khoa, Tp. Hồ Chí Minh. Hiện nay anh là chuyên viên phần mềm tại Công Ty LUXOFT, Tp. Hồ Chí Minh.

Dương Tuấn Anh tốt nghiệp tiến sĩ ngành Khoa học máy tính tại Học viện Công gghệ Á Châu (Asian Institute of Technology), Bangkok, Thái Lan, năm 1998 và đó cũng là nơi mà ông tốt nghiệp thạc sĩ với cùng chuyên ngành. Ông đã là Phó Giáo Sư tại khoa Khoa học và Kỹ thuật máy ính, Trường Đại học Bách Khoa Tp. Hồ Chí Minh từ năm 2007. Hiện nay, ông là giảng viên khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ-Tin học Tp. Hồ Chí Minh. Lĩnh vực nghiên cứu chính của ông là metaheuristics, học máy và khai phá dữ liệu chuỗi thời gian. Ông là đồng tác giả của trên 100 bài báo khoa học.