Thuật toán song song khai thác nhanh các mẫu trọng số hữu ích phổ biến từ cơ sở dữ liệu định lượng động

Lê Hoàng Bình Nguyên[1], Nguyễn Duy Hàm[2], Nguyễn Thị Hồng Minh[3]
[1] Trường Cao đẳng An ninh mạng iSPACE, TP. Hồ Chí Minh, Việt Nam
[2] Trường Đại học Sài Gòn, TP. Hồ Chí Minh, Việt Nam
[3] Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội, Việt Nam

Tác giả liên hệ: Nguyễn Duy Hàm, ndham@sgu.edu.vn
Ngày nhận bài: 29/07/2023, ngày sửa chữa: 10/09/2023, ngày duyệt đăng: 26/09/2023
Định danh DOI: 10.32913/mic-ict-research-vn.v2023.n2.1238

Tóm tắt: Trong những năm gần đây, bài toán khai thác các mẫu trọng số hữu ích phổ biến (FWUP) đang được quan tâm nghiên cứu. Đây là một là một biến thể của bài toán khai thác mẫu. Một số phương pháp giải quyết khá hiệu quả bài toán này đã được đề xuất. Tuy nhiên, trong thực tế, khi dữ liệu có kích thước lớn và trọng số các mục thường xuyên thay đổi dẫn đến các thuật toán xử lý tốn nhiều thời gian trong quá trình khai thác mẫu. Nếu tận dụng khả năng tính toán song song của các hệ thống tính toán, hoặc của các bộ vi xử lý đa lõi (multi-core) thì có thể cải thiện được thời gian tính toán của các thuật toán. Trong bài báo này chúng tôi giới thiệu một giải pháp song song hoạt động trên bộ xử lý đa lõi, đặt tên là pdFWUNL, để khai thác các mẫu trọng số hữu ích phổ biến từ cơ sở dữ liệu định lượng động có trọng số các mục thay đổi. Kết quả thực nghiệm cho thấy thời gian khai thác mẫu của thuật toán song song pdFWUNL của chúng tôi hiệu quả hơn phương pháp tuần tự tốt nhất hiện có.

Từ khóa: *mẫu trọng số hữu ích phổ biến, cơ sở dữ liệu định lượng động, dữ liệu lớn, tính toán song song*

Title: **A Parallel Algorithm for Fast Mining Frequent Weighted Ultility Patterns from Dynamic Quantitative Databases**

Abstract: In recent years, the problem of mining frequent weighted utility patterns has been receiving research attention. This is a variation of the pattern mining problem. Many effective methods have been proposed to solve this problem. However, when the data is extensive and the weights of items change frequently, the algorithms take much time during the mining process. If we take advantage of the parallel computing capabilities of computing systems or multi-core processors, we can improve the mining time of algorithms. This paper presents a parallel solution operating on multi-core processors, named pdFWUNL, to exploit frequent weighted utility patterns from dynamic quantitative databases with variable item weight. The experimental results show that the mining time of our parallel algorithm, pdFWUNL, is more efficient than the best available sequential method.

Keywords: *frequent weighted utility patterns, dynamic quantitative database, big data, parallel computation*

I. GIỚI THIỆU

Công nghệ đang ngày càng phát triển và tác động đến tất cả các lĩnh vực của đời sống xã hội, mang lại những hiệu quả tích cực. Cùng với đó là lượng lớn dữ liệu đang liên tục được tạo ra trong mỗi phút giây. Những dữ liệu này có thể được tạo ra bởi con người hoặc máy móc, và được thu thập bằng nhiều phương thức, công nghệ khác nhau. Có thể kể đến như dữ liệu bán hàng, mạng xã hội, nhật ký hệ thống, dữ liệu đo đạc hiện tượng thiên nhiên... Nếu khai thác được những thông tin hữu ích từ dữ liệu này sẽ mang lại rất nhiều lợi thế, đặc biệt là trong kinh doanh, bán hàng, trong các hệ thống hỗ trợ ra quyết định, các hệ thống thông minh...

Trong lĩnh vực khai thác dữ liệu, khai thác mẫu là một trong những bài toán quan trọng [1]. Nhiệm vụ của khai thác mẫu là khám phá tất cả các mẫu phổ biến xuất hiện trong cơ sở dữ liệu (CSDL), ví dụ như tập sản phẩm được mua nhiều nhất bởi khách hàng. Khai thác mẫu trọng số hữu ích phổ biến (Frequent Weighted Utility Pattern - FWUP) [2, 3] là một biến thể của khai thác mẫu với mục tiêu tìm

ra các mẫu vừa xuất hiện thường xuyên, vừa có lợi ích và và tầm quan trọng (trọng số) cao. Đã có nhiều thuật toán được đề xuất để giải quyết hiệu quả bài toán này như MBiS [4], WUN-Miner [5], SWUNL-FWUP [6] trên CSDL định lượng tĩnh và dFWUT [7], dFWUNL [7] trên CSDL định lượng động. CSDL định lượng động (dQDB) là CSDL mà trọng số của các mục có thể thay đổi trong quá trình giao dịch chứ không cố định. Tuy nhiên, các thuật toán khai thác FWUP nêu trên đều chỉ chạy trên các bộ xử lý đơn lõi. Điều này có thể làm hạn chế hiệu suất, không thể mở rộng trên bộ xử lý đa lõi, cũng như gặp khó khăn trong việc khai thác trên các CSDL lớn, dữ liệu được tích luỹ nhanh chóng theo thời gian.

CSDL định lượng động xuất hiện nhiều trong thực tế, trọng số của mỗi mục dữ liệu có thể là lợi nhuận, giá cả hay lợi ích… của mặt hàng, các giá trị này có thể thay đổi theo từng giao dịch. Ví dụ giá của mặt hàng có thể thay đổi theo số lượng mua (khi mua với số lượng nhiều có thể được giá thấp hơn), có thể thay đổi theo thời gian (giá mùa hè có thể khác giá mùa đông với những mặt hàng điện lạnh)… Điều này dẫn đến bài toán có tính thực tiễn cao là khai thác mẫu phổ biến từ CSDL định lượng động với trọng số thay đổi.

Trong bài báo này, chúng tôi thực hiện một số nội dung sau: (1) Giới thiệu bài toán khai thác FWUP từ CSDL định lượng động; (2) Đề xuất một phương pháp song song, đặt tên là pdFWUNL để khai thác FWUP từ CSDL định lượng động, với khả năng chia tính toán thành nhiều tác nhiệm có thể thực hiện đồng thời, nhằm giảm chi phí xử lý; (3) Đề xuất cơ chế cân bằng tải động để phân phối công việc giữa các tiến trình khi có bộ xử lý nhàn rỗi trong quá trình tính toán; và (4) Tiến hành thực nghiệm, đánh giá và so sánh thuật toán đề xuất pdFWUNL với thuật toán gốc dFWUNL. Kết quả thử nghiệm cho thấy thuật toán pdFWUNL tốt hơn phiên bản gốc về thời gian thực hiện đối với tất cả các bộ dữ liệu thử nghiệm.

Nội dung bài viết được tổ chức như sau: Phần II giới thiệu các nghiên cứu liên quan. Phần III trình bày một số khái niệm và kiến thức cơ bản. Phần IV mô tả thuật toán dFWUNL và trình bày chi tiết về thuật toán song song được đề xuất. Phần V mô tả kết quả thực nghiệm. Cuối cùng, Phần VI rút ra kết luận và hướng nghiên cứu tiếp theo.

II. CÁC NGHIÊN CỨU LIÊN QUAN

Cơ sở dữ liệu định lượng thường xuất hiện trong dữ liệu bán hàng. Điểm đặc trưng của loại dữ liệu này là mỗi giao dịch lưu giữ số lượng các sản phẩm được mua và mỗi sản phẩm có một trọng số khác nhau. Trọng số này có thể là giá cả, lợi nhuận, hoặc tầm quan trọng của sản phẩm đó trong CSDL. Đã có nhiều hướng tiếp cận để khai thác thông tin có giá trị từ CSDL định lượng. Hướng nghiên cứu đầu tiên đó là khai thác các mẫu lợi ích cao (High Utility Pattern- HUP) do Yao và các đồng sự đề xuất [8]. Khai thác HUP tập trung vào việc tìm kiếm các mẫu có giá trị hữu ích không nhỏ hơn một ngưỡng hữu ích cho trước. Sau đó, nhiều thuật toán giải quyết bài toán này đã được đề xuất như Two-Phase [9], UP-Growth+ [10], HUI-Miner [11], EFIM [12], và các biến thể của HUP như khai thác HUP đóng [13], top-k HUP [14], HUP trung bình [15],...

Khai thác HUP chỉ quan tâm đến lợi ích của các mẫu, trong khi tần số xuất hiện của các mẫu cũng đóng vai trò quan trọng. Do đó, Khan và đồng sự đã đề xuất một hướng nghiên cứu mới đó là khai thác luật kết hợp trọng số hữu ích [2]. Tác giả đã đưa ra độ đo mới là trọng số hữu ích giao dịch, kết hợp hai yếu tố là tần số và tầm quan trọng để trích xuất được các FWUP, đồng thời phát triển một thuật toán dựa trên phương pháp Apriori. Tiếp đó, Vo và các đồng sự [3] đề xuất cấu trúc MWIT-tree với chỉ một lần quét CSDL để khai thác FWUP. Nguyen và các đồng sự [4] đã đề xuất cấu trúc MBiS để tối ưu việc lưu trữ tidset nhằm tăng hiệu quả khai thác. Tiếp theo, Bui và các đồng sự [5] đề xuất cấu trúc WUN-set cùng với thuật toán WUN-Miner, và sau đó, Nguyen và các đồng sự [6] phát triển thuật toán FWUP-SWUNL sử dụng cấu trúc SWUN-list và đề xuất các định lý tăng tốc quá trình khai thác. Kết quả thực nghiệm cho thấy FWUP-SWUNL là thuật toán khai thác FWUP tốt nhất hiện nay.

Các nghiên cứu khai thác FWUP ở trên đều tập trung vào CSDL định lượng tĩnh, trọng số các mục là cố định. Tuy nhiên trên thực tế, tầm quan trọng của mỗi sản phẩm chịu ảnh hưởng của nhiều yếu tố khác nhau như chiến lược giảm giá, thanh lý mặt hàng, thay đổi hành vi người dùng,... Điều này dẫn đến trọng số của từng sản phẩm cần được điều chỉnh cho phù hợp. Để giải quyết vấn đề này, Nguyen và các đồng sự [7] đã phát biểu bài toán và đề xuất phương pháp (thuật toán dFWUNL) khai thác FWUP từ CSDL định lượng động, trong đó trọng số của các mục có thể thay đổi theo thời gian. Thuật toán dFWUNL mặc dù giải quyết khá hiệu quả bài toán nhưng vẫn còn hạn chế trên các CSDL lớn, đặc biệt về mặt thời gian.

Để giải quyết thách thức về thời gian tính toán khi dữ liệu lớn, ý tưởng song song hoá các thuật toán khai thác dữ liệu là một hướng tiếp cận. Một trong những mô hình phổ biến để áp dụng tính toán song song là sử dụng bộ xử lý đa lõi. Thiết kế đa lõi, công nghệ khá phổ biến hiện nay trong sản xuất chip xử lý, cho phép thực hiện nhiều tác nhiệm đồng thời để cải thiện hiệu suất của các quá trình tính toán. Tuy nhiên cần có thuật toán có khả năng phân rã các tiến trình và điều phối phù hợp. Đối với bài toán khai thác mẫu, đã có khá nhiều nghiên cứu sử dụng kiến trúc đa lõi và đạt được hiệu quả đáng kể [16–18].

III. CÁC KHÁI NIỆM CƠ BẢN

Một CSDL định lượng động là một bộ bao gồm ba thành phần (T, I, DW). Trong đó $T = \{t_1, t_2, ..., t_m\}$ là tập các giao dịch định lượng với m là số lượng giao dịch, $I = \{i_1, i_2, ..., i_n\}$ là tập mục với n là số lượng mục, và $DW = \{dw_1, dw_2, ..., dw_p\}$ là các tập trọng số với p là số lượng lô (batch). Mỗi giao dịch t_k có dạng $t_k = \{x_{k1}, x_{k2}, ..., x_{kn}\}$, $k = 1..m$, trong đó x_{ki} là số lượng của mục I trong giao dịch t_k (có thể hiểu như số lượng mặt hàng thứ i được mua trong giao dịch t_k), và tương ứng với một $dw_x \in DW$. Mỗi lô $dw_x = \{w_1, w_2, ..., w_n\}$, $x = 1..p$, là một tập trọng số của các mục trong I. Điểm khác biệt của CSDL định lượng động đó là trọng số của một mục có thể thay đổi trong các giao dịch khác nhau dựa trên tầm quan trọng của mục đó tại các thời điểm khác nhau. Mỗi lô giao dịch có thể có một hoặc nhiều giao dịch và tương ứng với một tập trọng số.

Bảng I mô tả về một CSDL định lượng động. CSDL có 5 giao dịch $T = \{t_1, t_2, t_3, t_4, t_5\}$ và năm mục $I = \{A, B, C, D, E\}$ với 2 lô $DW = \{dw1, dw2\}$. Trọng số của các mục trong giao dịch t_1, t_2, t_3 là $dw_1 = \{0.3, 0.5, 0.2, 0.4, 0.7\}$, và các giao dịch t_4, t_5 có trọng số là $dw_2 = \{0.4, 0.3, 0.4, 0.2, 0.3\}$.

Bảng I
VÍ DỤ VỀ CSDL ĐỊNH LƯỢNG ĐỘNG (dQDB$_E$)

Bảng giao dịch của CSDL						
Giao dịch	*A*	*B*	*C*	*D*	*E*	**Trọng số**
t_1	1	0	2	3	1	dw_1
t_2	0	2	1	0	2	
t_3	2	0	1	0	1	
t_4	1	0	3	2	0	dw_2
t_5	3	1	1	2	1	

Bảng trọng số của CSDL					
Trọng số	*A*	*B*	*C*	*D*	*E*
dw_1	0.3	0.5	0.2	0.4	0.7
dw_2	0.4	0.3	0.4	0.2	0.3

Định nghĩa 1. [7] Giá trị trọng số hữu ích giao dịch (twu) của một giao dịch t_k với tập trọng số $dw_p \in DW$ được xác định như sau:

$$twu\,(t_k) = \frac{\sum_{i_j \in t_k} w_{pi_j} \times x_{ki_j}}{|t_k|}$$

Trong đó, $w_{pi_j} \in dw_p$ là trọng số của mục i_j trong giao dịch t_k, x_{ki_j} là số lượng của mục i_j trong t_k, và $|t_k|$ là số lượng các mục trong t_k.

Ví dụ 1. Giao dịch t_2 trong dQDB$_E$ có giá trị twu được tính như sau:

$$twu(t_2) = (0.5 \times 2 + 0.4 + 0.7 \times 2)/3 \approx 0.87$$

Tương tự, giá trị twu của tất cả các giao dịch trong dQDB$_E$ được tính như trong Bảng II.

Bảng II
GIÁ TRỊ twu CỦA TẤT CẢ GIAO DỊCH TRONG (dQDB$_E$)

Giao dịch	**Trọng số**	**Công thức tính**	**twu**
t_1	dw_1	$(0.3 \times 1 + 0.2 \times 2 + 0.5 \times 3 + 0.7)/4$	0.65
t_2		$(0.5 \times 2 + 0.4 + 0.7 \times 2)/3$	0.87
t_3		$(0.3 \times 2 + 0.2 + 0.7)/3$	0.50
t_4	dw_2	$(0.4 + 0.4 \times 3 + 0.2 \times 2)/3$	0.67
t_5		$(0.4 \times 3 + 0.3 + 0.4 + 0.2 \times 2 + 0.3)/5$	0.52
Tổng giá trị twu ($sumtwu$)			3.21

Định nghĩa 2. [7] Gọi $sumtwu = \sum_{t_k \in T} twu(t_k)$ là tổng giá trị twu của các giao dịch trong dQDB. Độ hỗ trợ trọng số hữu ích (wus) của một mẫu X được xác định như sau:

$$wus\,(X) = \frac{\sum_{t_k \in t(X)} twu\,(t_k)}{sumtwu}$$

Trong đó, $t(X)$ là tập các giao dịch trong CSDL có chứa mẫu X.

Ví dụ 2. Giá trị wus của mẫu AE được tính như sau:

$$wus\,(AE) = \frac{twu\,(t_1) + twu\,(t_3) + twu\,(t_5)}{sumtwu} = \frac{0.65 + 0.5 + 0.52}{3.21} \approx 0.52$$

Cho CSDL định lượng động và một ngưỡng hỗ trợ trọng số hữu ích tối thiểu (ký hiệu là $minwus$), bài toán khai thác FWUP là tìm tất cả các mẫu X có giá trị $wus(X)$ lớn hơn hoặc bằng $minwus$.

IV. THUẬT TOÁN PDFWUNL KHAI THÁC MẪU PHỔ BIẾN TRÊN CSDL ĐỊNH LƯỢNG ĐỘNG

1. Thuật toán dFWUNL

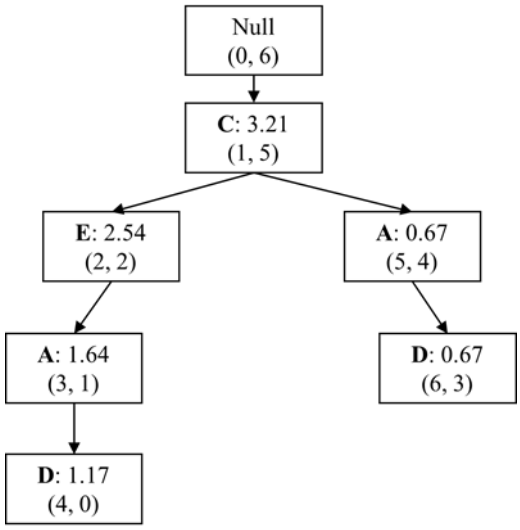
Phần này giới thiệu về thuật toán dFWUNL [7] và các thành phần của nó, đây là cơ sở để đề xuất thuật toán song song hóa của chúng tôi trong phần sau. Thuật toán dFWUNL khai thác các FWUP từ dQDB được đề xuất bởi Nguyen và đồng sự sử dụng cấu trúc WUNList, là mở rộng của cấu trúc N-list [19], được tạo thành từ cấu trúc WUN-tree. Về cơ bản, thuật toán dFWUNL gồm 3 giai đoạn:

1) Giai đoạn đầu tiên quét dQDB để xây dựng WUN-tree, tìm và sắp xếp giảm dần các 1-FWUP theo wus. Các 1-FWUP này được lưu trữ trong I_1. WUN-tree là cấu trúc cây, mỗi nút lưu trữ năm giá trị $name$, pre, $post$, $weight$, và $children$ tương ứng với tên của mục đại diện cho nút, thứ tự của nút khi duyệt pre-order (tiền thứ tự) và post-order (hậu thứ tự), tổng twu của các giao dịch đi qua nút này, và tập các nút con của nút đó.
2) Giai đoạn thứ hai tạo WUNList của các 1-FWUP trong I_1 bằng cách trích xuất thông tin từ WUN-tree. Cấu trúc dữ liệu WUNList của một mục X, ký hiệu là $wl(X)$ là một dãy các WUN-codes

$\{(pre_1, post_1, weight_1), (pre_2, post_2, weight_2), ..., (pre_n, post_n, weight_n)\}$ của các nút đại diện cho mục X trên WUN-tree. WUNList được sắp xếp tăng dần theo giá trị pre $(pre_1 < pre_2 < ... < pre_n)$. Đồng thời, tiến hành tính wus của 2-pattern thông qua WUN-tree và lưu trữ trong ma trận $F2$.

3) Giai đoạn cuối cùng tiến hành tìm kiếm tất cả các FWUP. Thuật toán dFWUNL sử dụng cấu trúc set-enumerations-tree [20] kết hợp với các định lý để tăng tốc quá trình khai thác. Trong quá trình tạo set-enumerations-tree, các nhánh của cây sẽ có cùng tiền tố. Do đó, chúng ta có thể xác định được k-FWUP từ hai $(k-1)$-FWUP và tính nhanh wus thông qua việc giao các WUNList (WUNList intersection). Kết thúc thuật toán, ta thu được đầy đủ tất cả các FWUP thoả mãn ngưỡng $minwus$.

Ví dụ 3. Hình 1 mô tả WUN-tree được tạo thành từ $dQDB_E$ với ngưỡng $minwus = 0.5$. Mục B bị loại bỏ vì $wus(B) = 0.43 < minwus$. Nút đại diện cho mục C có WUN-codes là $(1, 5, 3.21)$. Từ WUN-tree, ta tạo được WUNList của các 1-FWUP. Ví dụ, WUNList của mẫu A là $wl(A) = \{(3, 1, 1.64), (5, 4, 0.67)\}$. WUNList của các 1-FWUP được sử dụng để khai thác các FWUP tiếp theo và cho hiệu quả vượt trội.



Hình 1. WUN-tree được tạo từ $dQDB_E$ với $minwus = 0.5$

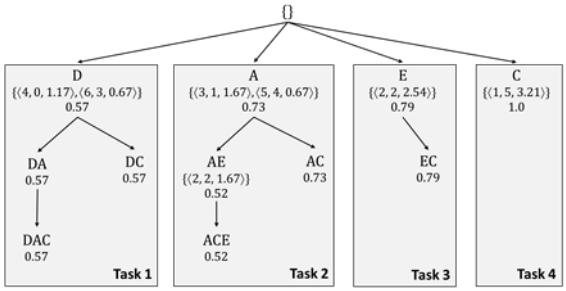
Thuật toán dFWUNL đã triển khai được việc khai thác mẫu trên CSDL định lượng động, tuy nhiên vẫn còn thách thức về thời gian khai thác trên các CSDL lớn và chưa có khả năng tận dụng sức mạnh của các hệ thống tính toán song song.

2. Đề xuất chiến lược song song cho thuật toán dFWUNL

Trong phần này, chúng tôi trình bày thuật toán có tên là pdFWUNL khai thác nhanh các FWUP trên dQDB. Thuật toán pdFWUNL là mở rộng của thuật toán dFWUNL [7], sử dụng kiến trúc bộ xử lý đa lõi để song song hoá quá trình tìm kiếm các mẫu trọng số hữu ích phổ biến.

Giai đoạn 3 của thuật toán dFWUNL tạo thành set-enumerations-tree với các nhánh khác nhau. Các nhánh này bắt đầu từ các 1-FWUP, sau đó được mở rộng dần. Quá trình khai thác trên các nhánh là hoàn toàn độc lập, không có sự phụ thuộc dữ liệu, tính toán lẫn nhau. Do đó, chúng ta có thể áp dụng chiến lược song song vào giai đoạn này. Thuật toán pdFWUNL áp dụng mô hình task-parallelism trên hệ thống bộ nhớ dùng chung. Mô hình này chia chương trình thành các tác vụ (task) riêng lẻ và sau đó tiến hành thực hiện song song các tác vụ độc lập. Thuật toán coi mỗi 1-FWUP trong I_1 là một task riêng biệt và phân phối các task này vào các lõi của bộ xử lý để khai thác độc lập. Mỗi lõi của bộ xử lý sau đó sẽ tiến hành tạo các cây con tương ứng với nút mà nó được chỉ định, sử dụng tìm kiếm theo chiều sâu (DFS).

Ví dụ 4. Hình 2 mô tả quá trình phân chia không gian tìm kiếm thành các công việc độc lập và tiến hành khai thác trên CSDL $dQDB_E$ với $minwus = 0.5$ của thuật toán pdFWUNL. Có bốn 1-FWUP trong I_1 nên có bốn task, và các task này được các lõi của bộ xử lý khai thác song song.



Hình 2. Phân chia không gian tìm kiếm của thuật toán pdFWUNL

Một khía cạnh quan trọng khi áp dụng khai thác song song đó là đảm bảo cân bằng tải. Làm sao phân bổ các công việc vào các bộ hoặc lõi xử lý để thời gian nhàn rỗi (không hoạt động) của các bộ hoặc lõi xử lý là ít nhất có thể. Thuật toán pdFWUNL sử dụng cơ chế cân bằng tải động áp dụng đối với các bộ xử lý đa lõi. Cụ thể như sau: 1-FWUP chưa được khai thác sẽ được đưa vào lõi xử lý khả dụng (đang nhàn rỗi) để tiến hành khai thác; nếu không có lõi xử lý nào, nó sẽ đợi cho đến khi có lõi xử lý thoả mãn để chỉ định 1-FWUP chưa xử lý tiếp theo.

Mã giả của pdFWUNL được hiển thị trong Thuật toán 1. Đầu tiên, thuật toán pdFWUNL tiến hành quét CSDL để tạo I_1, WUN-tree, WUNList của các mục trong I_1 và tính *wus* của 2-pattern lưu trữ trong ma trận F_2 (dòng 1-3). Tiếp theo, mỗi 1-FWUP là một task với nhiệm vụ tìm kiếm các FWUP theo nhánh của mục đó với thủ tục Find_FWUP (dòng 4-6).

Quá trình này được thực thi song song và liên tục được phân phối vào các lõi đang nhàn rỗi. Thủ tục Find_FWUP gồm các tham số $Prefix$, L, S, x và $level$ lần lượt là tiền tố chung, tập mục cuối cùng của các mẫu của lớp hiện tại, tập hợp các mục có thể tính nhanh *wus*, chỉ mục của mục đang xét và độ dài của mẫu. Trong thủ tục Find_FWUP, với độ dài ứng viên là 2 ($level = 2$), ta có thể tính nhanh các *wus* của 2-pattern này thông qua $F2$, nhờ đó nhanh chóng loại bỏ các mẫu không thoả mãn (dòng 10-22). Với các trường hợp khác ($level > 2$), ta tính WUNList của các ứng viên đó thông qua việc giao các WUNList để tìm ra FWUP (dòng 23-43). Thuật toán kết thúc khi không còn mẫu mới được tạo thành và ta thu được tất cả FWUP thoả mãn.

Thuật toán 1: Thuật toán pdFWUNL

```
   Input: CSDL định lượng động dQDB và ngưỡng minwus
   Output: FWUPs, tập tất cả các mẫu trọng số hữu ích phổ biến
1  Quét CSDL để tạo I1 và WUN-Tree
2  FWUPs ← I1
3  Duyệt WUN-tree để tạo WUNList của các mục trong I1, và tính wus
     của 2-pattern lưu trữ trong ma trận F2
4  for x ← |I1| − 1 downto 1 parallel do
5      Prefix2 ← {ix}, L2 ← ∅, S2 ← ∅
6      Find_FWUP(Prefix2, L2, S2, x, 2)
7  return FWUPs
8  Procedure Find_FWUP(Prefixk, Lk, Sk, x, level)
9      Prefixnext ← Prefixk
10     if level == 2 then
11         Prefixnext ← {ix}, Lnext ← ∅, Snext ← ∅
12         for y ← 0 to x − 1 do
13             wus(ixiy) ← F2[ix][iy]/sumtwu
14             if wus(ixiy) == wus(ix) then
15                 Snext ← Snext ∪ iy
16             else if wus(ixiy) ≥ minwus then
17                 wl(ixiy)  ← WUN-
                     List_intersect(wl(ix), wl(iy))
                   Lnext ← Lnext ∪ iy
18                 FWUPs ← FWUPs ∪ ixiy
19         if |Snext| > 0 then
20             Find_FWUP_sameWUS(Prefixnext, Snext)
21         if |Lnext| > 0 then
22             Find_FWUP(Prefixnext, Lnext, Snext, x, level+1)
23     else
24         for i ← |Lk| − 1 downto 1 do
25             Prefixnext ← Prefixnext ∪ Lk[i]
26             Lnext ← ∅, Snext ← Sk
27             for j ← 0 i − 1 do
28                 C ← Lk[j]
29                 wl(C)  ← WUN-
                     List_intersect(wl(Lk[i], wl(Lk[j])
30                 if wl(C) ≠ NULL then
31                     if wus(C) == wus(Lk[i]) then
32                         Snext ← Snext ∪ Lk[j]
33                     else
34                         Lnext ← Lnext ∪ C
35                         FWUPs ← FWUPs ∪ {Prefixk ∪
                             Lk[i] ∪ Lk[j]}
36             if |Snext| > 0 then
37                 Find_FWUP_sameWUS(Prefixnext, Snext)
38             if |Lnext| > 0 then
39                 Find_FWUP(Prefixnext, Lnext, Snext, x, level+
                     1)
40             remove last item of Prefixnext
41         Prefixnext ← Prefixnext ∪ Lk[0]
42         if |Sk| > 0 then
43             Find_FWUP_sameWUS(Prefixnext, Sk)
44 Procedure Find_FWUP_sameWUS(Prefix, S)
45     Sinh ra các tập con của S và lưu trữ vào Child
46     foreach c ∈ Child do
47         FWUPs ← FWUPs ∪ {Prefix ∪ c}
```

V. THỰC NGHIỆM

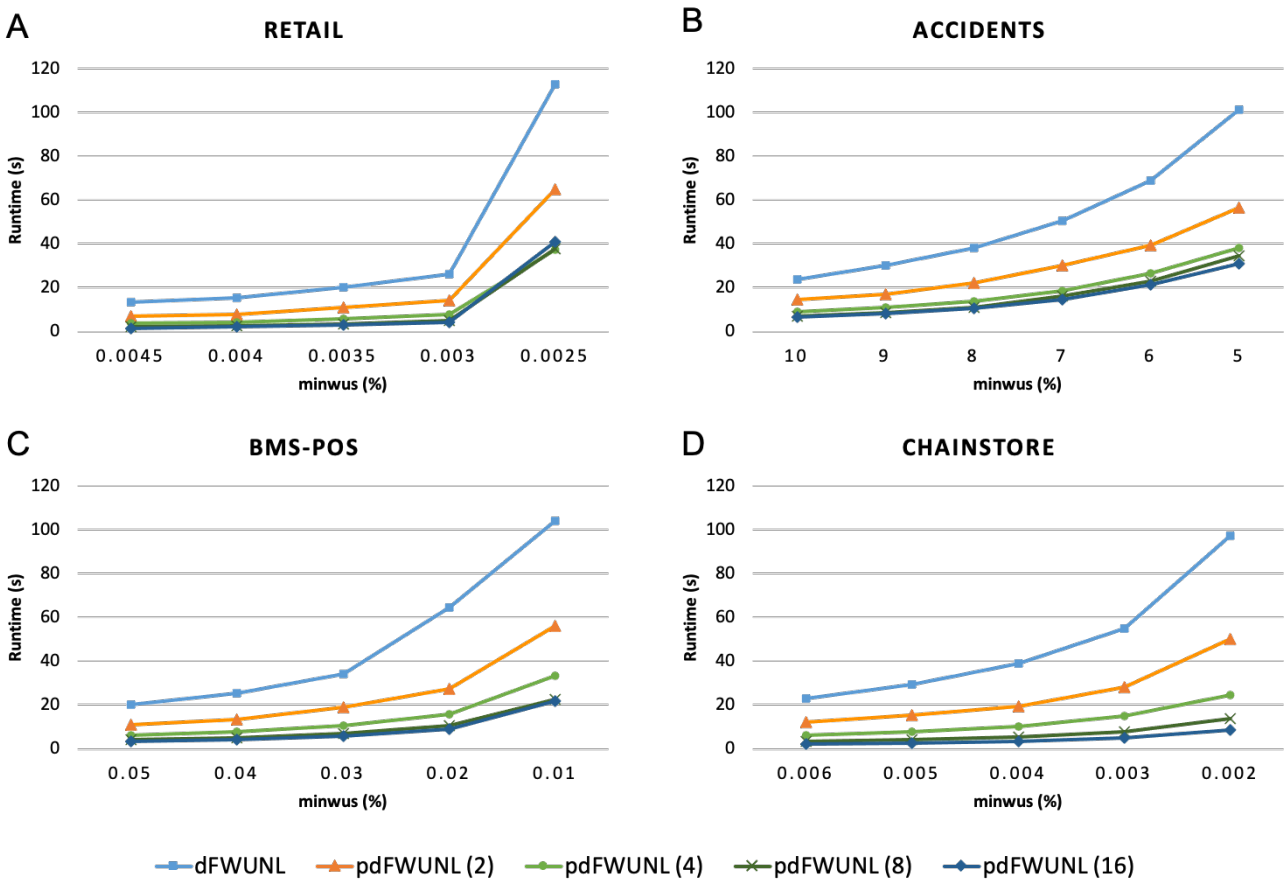
Phần này trình bày kết quả thực nghiệm thuật toán pdFWUNL và so sánh thời gian thực hiện với thuật toán dFWUNL trên một số bộ dữ liệu định lượng động khác nhau. Đồng thời, để kiểm tra khả năng mở rộng và tính hiệu quả của phương pháp song song, chúng tôi cũng thực nghiệm thuật toán pdFWUNL với số lượng lõi xử lý khác nhau, bao gồm: 2, 4, 8 và 16 lõi. Các thực nghiệm đều được tiến hành trên cùng hệ thống Dual CPU Intel Xeon Silver 4114 2.2 GHz (20 lõi), RAM 32GB, chạy hệ điều hành Windows 10. Các thuật toán được phát triển bằng ngôn ngữ C# trên nền tảng .NET Framework 4.7, sử dụng thư viện Task Parallel Library hỗ trợ lập trình song song.

Bảng III mô tả các bộ dữ liệu thực nghiệm bao gồm: Accident, BMS-POS, Retail, và Chainstore được lấy từ thư viện mã nguồn mở SPMF [21]. Để có được CSDL định lượng động và không mất tính tổng quát, chúng tôi đã chia mỗi CSDL thử nghiệm thành nhiều lô. Mỗi lô có số lượng giao dịch bằng nhau, ngoại trừ lô cuối cùng chứa các giao dịch còn lại. Mỗi lô giao dịch tương ứng với một bộ trọng số được tạo ngẫu nhiên trong khoảng [1,10].

Bảng III
CƠ SỞ DỮ LIỆU THỰC NGHIỆM

CSDL	Số lượng mục	Số lượng giao dịch	Số lượng giao dịch mỗi lô
Retail	16,470	88,162	10000
Accidents	468	340,183	5000
BMS-POS	1657	515597	50000
Chainstore	46,086	1,112,949	80000

Hình 3 hiển thị thời gian thực hiện của thuật toán dFWUNL và thuật toán pdFWUNL trên các bộ dữ liệu thực nghiệm với các ngưỡng minwus khác nhau. Kết quả cho thấy thuật toán pdFWUNL vượt trội hoàn toàn so với thuật toán dFWUNL trên tất cả các thực nghiệm với số lượng lõi khác nhau. Điều này là do thuật toán song song có thể tận dụng được sức mạnh của kiến trúc bộ xử lý đa lõi. Khi ngưỡng minwus càng nhỏ, sự chênh lệch về thời gian khai thác càng thể hiện rõ. Ví dụ, CSDL Accidents với



Hình 3. Thời gian khai thác của thuật toán dFWUNL và pdFWUNL

minwus = 6%, thời gian khai thác của thuật toán dFWUNL là 69s, còn pdFWUNL(2) là 39.3s và pdFWUNL (8) là 23s. Bên cạnh đó, thuật toán pdFWUNL cũng cho thấy sự hiệu quả của mình khi số lượng lõi xử lý tăng lên. Ví dụ, CSDL Chainstore với *minwus* = 0.002%, thuật toán pdFWUNL chạy trên hai lõi (pdFWUNL(2)) mất 50s, còn chạy trên 16 lõi (pdFWUNL(16)) chỉ mất 8.6s. Tuy nhiên, sự gia tăng này là không tuyến tính và khi tăng lõi lên một mức độ nhất định, thời gian khai thác không giảm đi, thậm chí còn tăng lên. Điều này là cho thấy, thuật toán pdFWUNL còn phụ thuộc vào sự phân bố dữ liệu. Ví dụ, CSDL Retail với *minwus* = 0.0025%, thuật toán pdFWUNL chạy trên 16 lõi trong thời gian là 40.2s, trong khi chạy trên 8 lõi chỉ mất 38.6s.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đã giới thiệu một phương pháp sử dụng bộ xử lý đa lõi để song song hoá thuật toán dFWUNL khai thác FWUP từ CSDL định lượng động. Thuật toán song song được đề xuất có tên là pdFWUNL sử dụng cơ chế Task-parallel và cân bằng tải động để tận dụng sức mạnh tính toán của bộ xử lý. Các kết quả thực nghiệm cho thấy thuật toán song song vượt trội hoàn toàn về thời gian khai thác so với thuật toán tuần tự dFWUNL.

Trong thời gian tới, chúng tôi sẽ tiếp tục nghiên cứu các phương pháp để song song hoá hai giai đoạn còn lại của thuật toán pdFWUNL và tối ưu cơ chế cân bằng tải. Cùng với đó, nghiên cứu các phương pháp tính toán phân tán hoặc song song hoá trên các hệ thống khác để áp dụng khai thác FWUP trên những CSDL rất lớn.

TÀI LIỆU THAM KHẢO

[1] R. Agrawal, T. Imieliński, and A. Swami, “Mining association rules between sets of items in large databases,” vol. 22. ACM, 1993, pp. 207–216.

[2] M. S. Khan, M. Muyeba, and F. Coenen, “A weighted utility framework for mining association rules.” IEEE, 2008, pp. 87–92.

[3] B. Vo, B. Le, and J. J. Jung, “A tree-based approach for mining frequent weighted utility itemsets.” Springer, 2012, pp. 114–123.

[4] N. D. Ham, V. D. Bay, N. T. H. Minh, and T.-P. Hong, “Mbis: an efficient method for mining frequent weighted utility itemsets from quantitative databases,” *Journal of Computer Science and Cybernetics*, vol. 31, no. 1, pp. 17–30, 2015.

[5] H. Bui, B. Vo, and H. Nguyen, "Wun-miner: A new method for mining frequent weighted utility itemsets." IEEE, 2016, pp. 001 365–001 370.
[6] H. Nguyen, N. Le, H. Bui, and T. Le, "A new approach for efficiently mining frequent weighted utility patterns," *Applied Intelligence*, vol. 53, no. 1, pp. 121–140, 1 2023.
[7] H. Nguyen, N. Le, H. Bui, and T. Le, "Mining frequent weighted utility patterns with dynamic weighted items from quantitative databases," *Applied Intelligence*, 3 2023, [Online; accessed 2023-06-15]. [Online]. Available: https://link.springer.com/10.1007/s10489-023-04554-z
[8] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases." SIAM, 2004, pp. 482–486.
[9] Y. Liu, W.-k. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets." Springer, 2005, pp. 689–695.
[10] V. S. Tseng, B.-E. Shie, C.-W. Wu, and S. Y. Philip, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 8, pp. 1772–1786, 2012.
[11] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," 2012, pp. 55–64.
[12] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "Efim: a highly efficient algorithm for high-utility itemset mining." Springer, 2015, pp. 530–546.
[13] L. T. Nguyen, V. V. Vu, M. T. Lam, T. T. Duong, L. T. Manh, T. T. Nguyen, B. Vo, and H. Fujita, "An efficient method for mining high utility closed itemsets," *Information Sciences*, vol. 495, pp. 78–99, 8 2019.
[14] S. Krishnamoorthy, "Mining top-k high utility itemsets with effective threshold raising strategies," *Expert Systems with Applications*, vol. 117, pp. 148–165, 3 2019.
[15] H. Kim, U. Yun, Y. Baek, J. Kim, B. Vo, E. Yoon, and H. Fujita, "Efficient list based mining of high average utility patterns with maximum average pruning strategies," *Information Sciences*, vol. 543, pp. 85–105, 1 2021.
[16] U. Huynh, B. Le, D.-T. Dinh, and H. Fujita, "Multi-core parallel algorithms for hiding high-utility sequential patterns," *Knowledge-Based Systems*, vol. 237, p. 107793, 2 2022.
[17] H. M. Huynh, L. T. Nguyen, B. Vo, Z. K. Oplatková, P. Fournier-Viger, and U. Yun, "An efficient parallel algorithm for mining weighted clickstream patterns," *Information Sciences*, vol. 582, pp. 349–368, 1 2022.
[18] S. Kumar and K. K. Mohbey, "A review on big data based parallel and distributed approaches of pattern mining," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 1639–1662, 5 2022.
[19] Z. Deng, Z. Wang, and J. Jiang, "A new algorithm for fast mining frequent itemsets using n-lists," *Science China Information Sciences*, vol. 55, pp. 2008–2030, 2012.
[20] R. Rymon, "Search through systematic set enumeration," in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 1992, pp. 539–550.
[21] "Spmf: A java open-source data mining library, retrieved from http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php (accessed: 20 august 2020)."

Lê Hoàng Bình Nguyên đang làm Nghiên cứu sinh chuyên ngành Cơ sở toán học cho tin học tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội. Hiện công tác tại Khoa Công nghệ thông tin, Trường Cao đẳng An ninh mạng iSPACE, TP. Hồ Chí Minh.
Email: nguyen.le@ispace.edu.vn

Nguyễn Duy Hàm nhận học vị Tiến sĩ ngành Cơ sở toán học cho tin học vào năm 2017 tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội. Hiện là giảng viên Khoa Công nghệ thông tin của Trường Đại học Sài Gòn, TP. Hồ Chí Minh.
Email: ndham@sgu.edu.vn

Nguyễn Thị Hồng Minh Nhận bằng Thạc sĩ và Tiến sĩ về Toán - Tin của Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội vào năm 1996 và 2001. Hiện là Phó Giáo sư và công tác tại Khoa Toán - Cơ - Tin học, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội.
Email: minhnth@hus.edu.vn