KHAI THÁC MẪU TƯƠNG ĐỒNG PHỔ BIẾN ĐÓNG TRÊN DỮ LIỆU PHÂN CẤP

Trần Cẩm Tú, Phạm Đức Thành

Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM

tutc@huflit.edu.vn, phamducthanh@huflit.edu.vn

TÓM TẮT— Khai phá mẫu phổ biến truyền thống thường sinh ra số lượng lớn mẫu dư thừa. Nghiên cứu này đề xuất GFCP (Generalized Frequent Closed Pattern miner), một phương pháp lọc nhiều giai đoạn để khai phá tập mẫu cô đọng từ cơ sở dữ liệu (CSDL) phân cấp. Phương pháp này kết hợp khai thác mẫu đóng (loại bỏ dư thừa cấu trúc) với lọc tương đồng (sử dụng Jaccard và Kulczynski) và xử lý dư thừa ngữ nghĩa trong CSDL phân cấp. Các thực nghiệm toàn diện, thay đổi ngưỡng hỗ trợ (minSup) và tương đồng (minSim), đã được tiến hành trên các CSDL chuẩn. Kết quả cho thấy các biến thể GFCP (BitSet và List) vượt trội đáng kể so với baseline CoGAR-C, vốn thất bại trên CSDL Mushroom (vượt quá 3600 giây). Cụ thể, GFCP-BitSet đạt hiệu suất thời gian vượt trội trên CSDL dày (Mushroom), trong khi GFCP-List nhanh hơn trên CSDL thưa (Fruithut, Product) và sử dụng bộ nhớ hiệu quả hơn ở hầu hết các kịch bản. Nghiên cứu khẳng định GFCP là một giải pháp hiệu quả, có khả năng mở rộng, cung cấp sự đánh đổi linh hoạt giữa tốc độ và bộ nhớ tùy thuộc vào cấu trúc dữ liệu đầu vào.

Từ khóa— Độ đo tương đồng, Jaccard, Kulczynski, mẫu tương đồng phổ biến, mẫu đóng phổ biến, CSDL phân cấp.

I. GIỚI THIỆU

Khai thác mẫu phổ biến [1]là nhiệm vụ nền tảng trong khai thác dữ liệu, cho phép khám phá các mẫu có liên quan từ các CSDL lớn bởi nó không chỉ giúp rút trích tri thức tiềm ẩn trong dữ liệu giao dịch mà còn đóng vai trò quan trọng trong nhiều ứng dụng khác như khai thác luật phân loại, phân tích giỏ hàng, dự đoán hành vi khách hàng, phân tích dữ liệu sinh học và khám phá tri thức từ dữ liệu mạng xã hội. Phần lớn các nghiên cứu truyền thống chỉ tập trung vào việc nhận diện các mẫu mà không phân tích mối liên hệ giữa chúng. Tuy nhiên, trong nhiều lĩnh vực ứng dụng thực tiễn như khoa học xã hội [2], y tế [3] hay địa chất [4], các đối tượng dữ liệu tuy không hoàn toàn giống nhau nhưng vẫn có thể chia sẻ những đặc điểm tương đồng đáng kể. Việc phân tích mức độ tương đồng giữa các kết quả giúp phát hiện ra những mẫu có liên hệ về mặt ngữ nghĩa hoặc phổ biến cùng xuất hiện [5]. Những mẫu này được xác định dựa trên ngưỡng tương đồng thay vì yêu cầu sự trùng khớp tuyệt đối, và thường mang tính chất tương tự nhau [6].

Mặc dù các phương pháp khai thác dựa trên độ đo tương đồng có thể mang lại những tri thức tiềm ẩn và có giá trị, song chúng thường vấp phải hạn chế là sinh ra một số lượng lớn mẫu [7]. Điều này không chỉ làm phức tạp quá trình phân tích mà còn làm gia tăng chi phí tính toán. Để khắc phục vấn đề này, nhiều nghiên cứu đã chuyển hướng sang khai thác tập đóng, một kỹ thuật có khả năng tái tạo đầy đủ tập mẫu phổ biến mà không làm mất thông tin, đồng thời giảm đáng kể chi phí bộ nhớ và thời gian xử lý so với các phương pháp truyền thống [8]. Bên cạnh sự dư thừa về mặt cấu trúc, tương đồng thì sự dư thừa về mặt ngữ nghĩa trong các bộ dữ liệu có cấu trúc phân cấp (taxonomy), ví dụ như hệ thống phân loại sản phẩm hoặc cây phả hệ sinh học cũng chưa từng được nghiên cứu trước đây. Bài toán đặt ra là làm thế nào để xây dựng một quy trình toàn diện có khả năng xử lý đồng thời tất cả các loại dư thừa này.

Để giải quyết thách thức này, bài báo đề xuất một phương pháp lọc các tập mẫu kết quả một cách có hệ thống với các bước sau:

- Loại bỏ dư thừa cấu trúc: lọc chỉ giữ lại những mẫu đóng phổ biến.
- Loại bỏ dư thừa ngữ nghĩa: Đối với các dữ liệu có cấu trúc phân cấp, sử dụng một bộ lọc loại bỏ các mẫu con nếu chúng không cung cấp thêm thông tin mới so với các mẫu cha tổng quát hơn.
- Loại bỏ dư thừa tương đồng: Các mẫu được so sánh với nhau dựa trên độ đo tương đồng. Các mẫu có độ tương đồng cao với một mẫu khác có độ hỗ trợ lớn hơn sẽ bị loại bỏ.

Những đóng góp chính của nghiên cứu này bao gồm:

- Xây dựng một khung hoàn chỉnh để khai thác một tập hợp mẫu chất lượng cao, giải quyết đồng thời cả sự dư thừa về cấu trúc, ngữ nghĩa phân cấp và tương đồng.
- Tiến hành các thực nghiệm đánh giá hiệu năng của kỹ thuật triển khai dựa trên BitSet so với List.
- Phân tích chi tiết ảnh hưởng của các tham số độ phổ biến tối thiểu và tương đồng tối thiểu cùng các độ đo tương đồng khác nhau lên số lượng và đặc tính của tập mẫu kết quả.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 trình bày các công trình liên quan; Phần 3 nêu các khái niệm cơ bản; Phần 4 mô tả chi tiết phương pháp đề xuất và các thuật toán cốt lõi; Phần 5 trình bày thiết lập thực nghiệm và phân tích kết quả. Cuối cùng, Phần 6 đưa ra kết luận và đề xuất các hướng phát triển trong tương lai.

II. NGHIÊN CỨU LIÊN QUAN

A. KHAI THÁC MẪU PHỔ BIẾN

Khai thác mẫu phổ biến từ lâu đã là chủ đề nghiên cứu quan trọng trong lĩnh vực khai phá dữ liệu. Bài toán này phát hiện các tập mẫu phổ biến xuất hiện phổ biến trong CSDL giao dịch [1]. Một vài thuật toán hiệu quả trong việc giải quyết việc này có thể kể đến là: Apriori [8], Eclat [9], HMine [10] và nhiều các biến thể khác của nó. Tuy nhiên thách thức lớn của bài toán này vẫn là thời gian xử lý và mức độ tiêu thụ bộ nhớ [11] đặc biệt trong thời đại quy mô và độ phức tạp của dữ liệu tăng chóng mặt như hiện nay. Do đó, các thuật toán khai thác mẫu phổ biến được thiết kế nhằm hạn chế số lượng mẫu sinh ra, đồng thời vẫn duy trì đầy đủ thông tin cần thiết, từ đó mang lại giải pháp hiệu quả cho hiện tượng bùng nổ mẫu.

B. KHAI THÁC MẪU PHỔ BIẾN ĐÓNG

Khai thác mẫu phổ biến đóng nhằm phát hiện các mẫu phổ biến không bị bao hàm trong bất kỳ tập nào khác có cùng độ hỗ trợ [12]. Các mẫu đóng này vẫn giữ nguyên toàn bộ thông tin, bao gồm cả tần suất chính xác của chúng, mà không cần thực hiện việc quét lại CSDL ban đầu. Trong số những thuật toán đầu tiên được phát triển cho hướng tiếp cận này có CLOSET [13], DCI-Closed[14] và CHARM [15]. Thuật toán Closet tổ chức các mẫu phổ biến trong cấu trúc cây, từ đó trích xuất các mẫu đóng mà không cần sinh thêm ứng viên, đồng thời sử dụng cơ chế chiếu phân vùng để mở rộng khả năng xử lý trên tập dữ liệu lớn.

Một trong những vấn đề phát sinh đối với bài toán khai phá tập mục phổ biến đóng là việc phát hiện trùng lặp. Sự xuất hiện của DCI-Closed[14] giúp phát hiện và loại bỏ kịp thời các tập mục đóng trùng lặp, mà không cần lưu giữ toàn bộ tập hợp các mẫu đóng trong bộ nhớ chính. Thuật toán này tận dụng biểu diễn CSDL dọc để tính nhanh độ hỗ trợ cũng như xét nhanh tính đóng của các mẫu. Thực nghiệm cho thấy DCI-Closed vượt trội hơn hẳn CLOSET.

Trong khi đó, CHARM [15] áp dụng cấu trúc diffset nhằm giảm dung lượng bộ nhớ trong quá trình khai thác. Đồng thời, thuật toán này còn sử dụng kỹ thuật băm để nhanh chóng loại bỏ các mẫu không đóng. Một thuật toán nổi bật khác là DCI_Closed [16], khai thác chiến lược tìm kiếm theo chiều sâu và loại bỏ hiệu quả các mẫu dư thừa mà không cần lưu trữ toàn bộ tập mẫu đóng trong bộ nhớ chính.

Gần đây, nhiều công trình đã đưa ra các cải tiến mới, chẳng hạn như tích hợp thuật toán di truyền [16] hoặc phát triển thuật toán CLS-Miner [17], vốn dựa trên cấu trúc utility-list để trực tiếp xác định giá trị lợi ích mà không sinh ứng viên, hay khai thác tính toán song song nhằm tăng tốc quá trình xử lý các mẫu sinh ra [18]. Thuật toán MLC-Miner[19] được đề xuất để khai thác các tập mẫu hữu ích cao đóng từ CSDL phân cấp, áp dụng các chiến lược hiệu quả để cải thiện hiệu năng khai thác.

C. KHAI THÁC MẪU PHỔ BIẾN ĐÓNG SỬ DỤNG ĐỘ ĐO TƯƠNG ĐỒNG VỚI CSDL PHÂN CẤP

1. ĐỘ ĐO TÍNH TƯƠNG ĐỒNG

Việc đánh giá mức độ tương đồng giữa các mẫu đóng vai trò then chốt trong khai thác mẫu phổ biến tương đồng. Việc lựa chọn độ đo tương đồng phù hợp là việc quan trọng. Một cách cơ bản để đo độ tương đồng giữa các đối tượng dữ liệu là xem chúng như những vector trong một không gian vector trên $\mathbb{R}$, và sử dụng tích nội, được định nghĩa như sau [20] $\langle x, y\rangle = \sum_{i=1}^{n} x_i\, y_i$, trong đó, $x = (x_1, x_2, \dots\ x_n)$; $y = (y_1, y_2, \dots\ y_n)$ là hai vector. Bằng cách tính toán kích thước của phần giao của hai vector ta có thể dẫn xuất ra nhiều thước đo độ tương đồng và khoảng cách khác nhau như độ đo tương đồng Jaccard, Dice, Cosine… [21]. Độ tương đồng Jaccard là một biến thể của tích trong chuẩn hóa khá đơn giản trong tính toán, thường được sử dụng để đo lường mức độ tương đồng và đa dạng của các tập mẫu bằng cách lấy kích thước phần giao giữa các tập chia cho kích thước của hợp của chúng.

2. CSDL PHÂN CẤP

CSDL phân cấp là một mô hình tổ chức dữ liệu theo cấu trúc phân cấp, trong đó các đối tượng hoặc khái niệm được sắp xếp thành các nhóm và nhóm con dựa trên mối quan hệ "thuộc loại" [22] . Nói cách khác, taxonomy là một hệ thống phân loại có cấu trúc cây, giúp tổ chức và quản lý thông tin theo từng cấp độ từ khái quát đến chi tiết. Khi áp dụng vào CSDL, CSDL phân cấp đóng vai trò như một bảng phân loại chuẩn để gắn nhãn, tìm kiếm và khai thác dữ liệu hiệu quả hơn. Các thuật toán khai thác mẫu tương đồng phổ biến hiện nay mới chỉ quan tâm tới các hạng mục dữ liệu ở mức trừu tượng cấp thấp mà thôi.

3. KHAI THÁC MẪU PHỔ BIẾN ĐÓNG SỬ DỤNG ĐỘ ĐO TƯƠNG ĐỒNG VỚI CSDL PHÂN CẤP

Hầu hết các nghiên cứu truyền thống đều hướng tới xác định các mẫu phổ biến mà không xét đến các mối tương quan giữa chúng. Tuy nhiên bằng cách xem xét sự tương đồng giữa các kết quả, ta có thể khám phá ra những mẫu có liên hệ ngữ nghĩa hoặc phổ biến cùng xuất hiện [5]. Các mẫu này, được phát hiện dựa trên ngưỡng tương đồng thay vì sự trùng khớp tuyệt đối, được gọi là các mẫu phổ biến tương đồng [6]. Kỹ thuật khai phá dựa trên độ tương đồng có thể khám phá ra những tri thức tiềm ẩn và có giá trị, chúng thường gặp phải vấn đề sinh ra một lượng lớn mẫu [7], điều này có thể làm giảm khả năng diễn giải của tập kết quả và làm tăng chi phí tính toán. Để khắc phục nhược điểm này, các nghiên cứu đã chuyển hướng sang khai phá tập mẫu đóng, phương pháp này cho phép tái cấu trúc hiệu quả tập mẫu phổ biến đầy đủ mà không làm mất thông tin, đồng thời giảm đáng kể thời gian chạy và

bộ nhớ sử dụng so với các phương pháp truyền thống [15]. Bài toán mở rộng khai thác mẫu tương đồng phổ biến đóng trên CSDL phân cấp được xem là khá quan trọng trong các ứng dụng thực tế và kết quả khai thác được mang lại nhiều ý nghĩa hơn cho người sử dụng. Không gian bài toán khi áp dụng mô hình dữ liệu phân cấp có độ phức tạp cao hơn đáng kể so với bài toán truyền thống vì không gian tìm kiếm được mở rộng. Tuy nhiên, mặc dù đã có các công trình áp dụng mô hình CSDL phân cấp trong bài toán khai thác mẫu phổ biến, mẫu hữu ích cao, bài toán khai thác mẫu tương đồng phổ biến đóng trên CSDL phân cấp đến nay vẫn chưa có một nghiên cứu đầy đủ nào được đề xuất giải quyết bài toán này.

III. KHÁI NIỆM CƠ BẢN

Trong phần giới thiệu các khái niệm cơ bản liên quan tới khai thác mẫu tương đồng phổ biến đóng trên CSDL phân cấp như: Mẫu tương đồng phổ biến, mẫu tương đồng phổ biến đóng, CSDL phân cấp.

A. CƠ SỞ LÝ THUYẾT

Bài toán khai thác tập mẫu phổ biến nhận vào ba tham số chính: 1. CSDL giao dịch D, 2. Ngưỡng hỗ trợ tối thiểu ($minSup$), 3. Ngưỡng tương đồng tối thiểu ($minSim$). Kết quả của bài toán là tập hợp đầy đủ các mẫu có tần suất xuất hiện lớn hơn hoặc bằng $minSup$. Trong biến thể mở rộng bài toán khai thác các mẫu phổ biến tương đồng ngưỡng $minSim$ được bổ sung để đánh giá mức độ tương đồng giữa các mẫu. Hai mẫu X và Y được xem là tương đương nếu độ tương đồng giữa chúng (tính theo một thước đo xác định trước) vượt qua ngưỡng $minSim$.

Định nghĩa 1 (CSDL giao dịch): Cho I là một tập hợp hữu hạn gồm n mục phân biệt, $I = \{i_1, i_2, \dots, i_n\}$. Một mẫu X là một tập hợp hữu hạn các mục sao cho $X \subseteq I$ và được gọi là mẫu kích thước k nếu $k = |X|$. Một CSDL giao dịch là một tập hợp các giao dịch, được định nghĩa là $D = \{T_1, T_2, \dots, T_m\}$. Mỗi giao dịch $T_k \in D$, $1 \leq k \leq m$, có một định danh giao dịch duy nhất (TID) là k và một tập hợp các mục xuất hiện trong giao dịch đó [14].

Định nghĩa 2 (Tidset và Hỗ trợ): Hỗ trợ của một mẫu X trong D là số giao dịch chứa X, được ký hiệu là $sup(X)$ và được xác định là $sup(X) = |TidSet(X)|$. Trong khi đó, $TidSet(X)$ là tập hợp các TID của các giao dịch chứa X [14].

Định nghĩa 3 (Mẫu đóng): Một mẫu X được gọi là mẫu đóng khi và chỉ khi không tồn tại mẫu Y lớn hơn trong D sao cho $Y \supset X$ và $sup(X) = sup(Y)$. Tập hợp đầy đủ các mẫu đóng được ký hiệu là C.

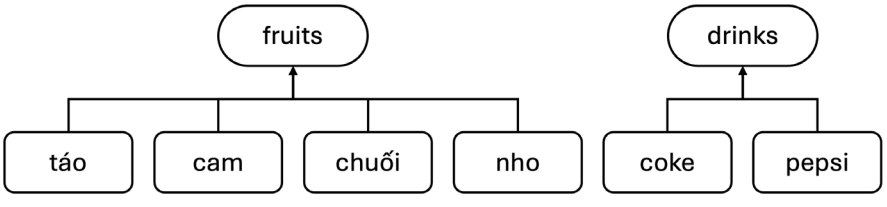
Định nghĩa 4 (Taxonomy). Một cấu trúc phân cấp các hạng mục trong CSDL D, ký hiệu τ, là một cấu trúc cây được định nghĩa trên D [23]. Trong τ, mỗi nút lá biểu diễn duy nhất một mục $i \in I$. Mỗi nút trong của τ biểu diễn một mục tổng quát ở mức trừu tượng cao hơn làm nhiệm vụ tổng hợp các nút con hoặc các mục tổng quát ở mức trừu tượng thấp hơn [23]. Giả sử rằng tất cả các mục $i \in I$ chỉ có thể được tổng quát hoá bởi τ vào duy nhất một mục tổng quát g mà thôi. Mỗi mục tổng quát g cũng có thể được tổng quát hoá thành một mục tổng quát khác ở mức trừu tượng cao hơn trong τ. Tập các mục tổng quát g trong τ được ký hiệu là G. Tất cả các nút con của g trong τ được ký hiệu là $\Delta(g, \tau)$, $\Delta(g, \tau) \subseteq I$ [23]. Chiều sâu tối đa của taxonomy τ được ký hiệu là δ. Hình 1 minh họa một cấu trúc phân cấp cho các mục thuộc CSDL D cho tại Bảng 1.

B. VÍ DỤ MINH HỌA

Bảng 1. *Một ví dụ về CSDL giao dịch D*

TID	Các mục
1	$\{táo, cam, coke\}$
2	$\{táo, cam, pepsi\}$
3	$\{táo, chuối, nho\}$
4	$\{cam, nho, coke\}$
5	$\{táo, cam\}$

Trên CSDL trên ta thấy tập mục $\{coke\}$ có độ hỗ trợ là 2, tuy nhiên, tập cha của nó là $\{cam,\ coke\}$ cũng có độ hỗ trợ bằng 2. Do có thể mở rộng tập mục mà không làm thay đổi tần suất xuất hiện, $\{coke\}$ là một tập không đóng. Ngược lại, một tập mục là đóng khi nó không có các tập cha có cùng độ hỗ trợ. Tập mục $\{táo,\ cam\}$ có độ hỗ trợ là 3. Phân tích cho thấy tất cả các tập cha của nó, ví dụ như $\{táo,\ cam,\ coke\}$, đều có độ hỗ trợ nhỏ hơn 3. Vì không tồn tại bất kỳ tập cha nào có cùng độ hỗ trợ, $\{táo,\ cam\}$ được kết luận là một tập đóng. Cấu trúc dữ liệu phân cấp được hiểu như sau: Trong một cửa hàng tạp hóa, ta có các mặt hàng như: $táo$, $chuối$, cam, $coke$, $pepsi$. Các mặt hàng này có thể được phân loại theo một cấu trúc phân cấp các mặt hàng như sau:



Hình 1. *Minh họa cấu trúc cây dữ liệu phân cấp cho CSDL D ở Bảng 1*

IV. PHƯƠNG PHÁP ĐỀ XUẤT

A. MỘT SỐ ĐỊNH NGHĨA

Định nghĩa 5 (Hàm tương đồng): Giả sử $s(X)$ và $s'(X')$ biểu diễn các mô tả của X và X' với $X, X' \in D$ và $S, S' \in I$. Tập hợp các đặc trưng khớp là $M = \{r \in T \mid X[r] = X'[r]\}$. Hàm tương đồng giữa X và X' được ký hiệu là $fs(X, X')$. Hàm tương đồng cho các phép đo tương đồng cụ thể sử dụng độ đo tương đồng Jaccard [2] được định nghĩa như sau: $fsJaccard(X, X') = \frac{}{|S \cup S'|}$ (1) và độ đo tương đồng Kulczynski: $fsKulc(X, X') = \frac{1}{2}\left(\frac{|M|}{|S|} + \frac{|M|}{|S'|}\right)$ (2)

Định nghĩa 6 (Các mẫu tương đồng): Giả sử $minSim$ là ngưỡng tương đồng tối thiểu, trong đó $0 \leq minSim \leq 1$. Hai mẫu X và X' được coi là tương đồng nếu $fs(X, X') \geq minSim$. Các mẫu X và X' được gọi chung là các mẫu tương đồng, ký hiệu là X'.

Định nghĩa 7 (Số lần xuất hiện của các mẫu tương đồng): Số lần xuất hiện của một mẫu X trong CSDL D đề cập đến các đối tượng X và các mẫu tương đồng của nó, ký hiệu là X', xuất hiện trong D.

Định nghĩa 8 (Độ phổ biến của các mẫu tương đồng): Tần suất của một mẫu tương tự X' trong CSDL D được ký hiệu là $sup(X')$ và đề cập đến số lượng giao dịch trong D mà mẫu tương tự X' xuất hiện. Nói cách khác, độ phổ biến của X' là số lượng phần tử của tập hợp các lần xuất hiện của X' trong D.

Định nghĩa 9 (Mẫu Tương đồng phổ biến): Một mẫu tương tự X' được coi là một mẫu tương đồng phổ biến trong D nếu $sup(X') \geq minSup$, trong đó $minSup$ là ngưỡng phổ biến tối thiểu.

Định nghĩa 10 (Mẫu Tương đồng phổ biến đóng): Một mẫu tương đồng phổ biến X được gọi là một mẫu tương tự đồng phổ biến đóng nếu nó là một mẫu phổ biến ($sup(X) \geq minSup$) và không tồn tại một tập cha Y của X ($Y \supset X$). Y là một mẫu phổ biến ($sup(Y) \geq minSup$) và Y tương đồng với X ($fs(X, Y) \geq minSim) \wedge sup(Y) \geq sup(X)$ (3).

Định nghĩa 11 (Khai thác mẫu tương đồng phổ biến đóng): Bài toán khai thác mẫu tương đồng phổ biến đóng là tìm tất cả các mẫu tương đồng phổ biến đóng X' thỏa mãn ngưỡng $minSim$ cho trước, với độ hỗ trợ $sup(X')$ thỏa ngưỡng $minSup$. Tập đầy đủ các mẫu tương đồng phổ biến đóng được ký hiệu là C'.

Khai thác mẫu tương đồng phổ biến đóng trên tập dữ liệu phân cấp là bài toán mở rộng tìm tất cả các mẫu tương đồng phổ biến X' trên CSDL phân cấp. Bằng cách xét các mẫu ở các mức trừu tượng của taxonomy τ, kết quả bài toán đảm bảo các mẫu vừa phổ biến, vừa tương đồng, vừa đóng tại mỗi mức phân cấp.

Trong nghiên cứu này, các mục được gọi là tương đồng theo một độ đo xác định dựa trên số lượng giao dịch mà chúng đồng xuất hiện. Các mục càng đồng xuất hiện trong càng nhiều giao dịch, điều đó chúng có tính tương đồng hoặc liên quan với nhau cao về mặt ngữ nghĩa. Xét một vài ví dụ sau sử dụng CSDL D tại Bảng 1 và các ngưỡng $minSup = 60\% = 3, minSim = 50\%$.

- Xét tập $\{\boldsymbol{táo}, \boldsymbol{cam}\}$: ta có $\{\boldsymbol{táo}\}$ xuất hiện trong giao dịch $\{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}\} \rightarrow \mathbf{sup}(\{\boldsymbol{táo}\}) = \mathbf{4}$; tương tự, $\{\boldsymbol{cam}\}$ xuất hiện trong giao dịch $\{\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{5}\} \rightarrow \mathbf{sup}(\{\boldsymbol{cam}\}) = \mathbf{4}$. Vậy $\mathbf{sup}(\{\boldsymbol{táo}, \boldsymbol{cam}\}) = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}\} \cap \{\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{5}\} = \{\mathbf{1}, \mathbf{2}, \mathbf{5}\} = \mathbf{3} \geq \boldsymbol{minSup} = \mathbf{3}$: $\{\boldsymbol{táo}, \boldsymbol{cam}\}$ là tập phổ biến. Độ tương đồng của $\{\boldsymbol{táo}, \boldsymbol{cam}\}$ theo Jaccard: $\frac{\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{5}\} \cap \{\mathbf{1},\mathbf{2},\mathbf{4},\mathbf{5}\}}{\{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{5}\} \cup \{\mathbf{1},\mathbf{2},\mathbf{4},\mathbf{5}\}} = \frac{\mathbf{3}}{\mathbf{5}} = \mathbf{0.6} \geq \boldsymbol{minSim}$. Điều này có nghĩa $\{\boldsymbol{táo}, \boldsymbol{cam}\}$ là tập mẫu tương đồng phổ biến, $\{\boldsymbol{táo}\}, \{\boldsymbol{cam}\}$ đồng xuất hiện trong 60% các giao dịch của $\boldsymbol{D}$.
- Xét $\{\boldsymbol{cam}, \boldsymbol{coke}\}$: $\mathbf{sup}(\{\boldsymbol{táo}, \boldsymbol{cam}\}) = \frac{\mathbf{2}}{\mathbf{5}} = \mathbf{0.4} < \boldsymbol{minSup} = \mathbf{60}\%$. Vậy $\{\boldsymbol{cam}, \boldsymbol{coke}\}$ không phổ biến (loại) và không được xét tiếp độ tương đồng.

Thuật toán 1: GFCP($D, \tau, minSup, minSim$)

Input: D – CSDL giao dịch
τ – Thông tin phân cấp các mục trong D
$minSup$ – ngưỡng độ hỗ trợ tối thiểu
$minSim$ – ngưỡng độ tương đồng tối thiểu

Output: R – tập đầy đủ các mẫu tương đồng phổ biến đóng

GIAI ĐOẠN 1: Khởi tạo và chuẩn bị dữ liệu

Đọc CSDL D, taxonomy τ
Xây dựng cấu trúc cây taxonomy để biểu diễn τ.

GIAI ĐOẠN 2: Khai thác đa mức

GIAI ĐOẠN 3: Biến đổi CSDL D thành CSDL D_k tại mức k, xây dựng CSDL dọc tương ứng

FOR EACH mức $k \in \tau$ **DO**
  $D_k = \emptyset$
  FOR EACH giao dịch $T_j \in D$ **DO**
    Thay thế các mục $i \in T_j$ bằng mục tổ tiên của nó trong τ tại mức k
    $D_k = D_k \cup T_j$.
  $VDB_k = \emptyset$ // xây dựng CSDL dọc VDB_k tại mức k
  FOR EACH $T_j \in D_k$ **DO**

FOR EACH $i \in T_j$ **DO**
$TIDset(i) = \{j \mid i \in T_j\}$
$support_i = |TIDset(i)|$
$VDB_k = VDB_k \cup TIDset(i)$
$F = \{i \mid support_i \geq minSup\}$ // tập các mục đơn phổ biến, sắp xếp F theo support tăng dần
GIAI ĐOẠN 4: Khai thác đệ quy sử dụng CSDL dọc VDB_k tại mức k.// biến thể từ DCI-Closed
$Closed = \emptyset, Post = F, Pre = \emptyset$
HÀM ĐỆ QUY Explore($Closed$, $Post$, Pre)
FOR EACH $v \in Post$ **DO**
$ext = Closed \cup \{v\}$
$TIDSet(ext) = \bigcap_{j \in ext} TIDset(j)$
IF $|TIDset(ext)| \geq minSup$ **THEN**
FOR EACH $w \in Pre$ **DO**
IF $TIDset(ext) \subseteq TIDset(w)$ **THEN CONTINUE**
$Closed' = ext, Post' = \emptyset$
FOR EACH $w \in Post, w \succ v$ **DO**
IF $TIDset(ext) \subset TIDset(w)$ **AND** $\sup(ext) = \sup(w)$ **THEN**
$Closed' = Closed' \cup \{w\}$
$TIDset(Closed') = \bigcap_{j \in Closed'} TIDset(j)$
ELSE
$Post' = Post' \cup \{w\}$
GIAI ĐOẠN 5: Lọc các mẫu tương đồng
FOR EACH pair $i_1, i_2 \in Closed'$ **DO**
$\mathbb{I} = TIDset(i_1) \cap TIDset(i_2), \mathbb{U} = TIDset(i_1) \cup TIDset(i_2)$
$Jaccard(i_1, i_2) = |\mathbb{I}|/|\mathbb{U}|$
IF $Jaccard(i_1, i_2) \geq minSim$ **AND** $support(\{i_1, i_2\}) \geq minSup$ **THEN**
$R = R \cup Closed'$
GIAI ĐOẠN 6: Đệ quy
Explore($Closed'$, $Post'$,$Pre \cup \{v\}$)
$Pre = Pre \cup \{v\}$
GIAI ĐOẠN 7: Kết thúc
RETURN R // tất cả các tập mục tương đồng phổ biến đóng đã khai thác

B. THUẬT TOÁN GFCP

Trong nghiên cứu này chúng tôi đề xuất thuật toán **GFCP** (Generalized Frequent Closed Pattern mining) để khai thác các tập mẫu đóng phổ biến tương đồng, sử dụng hai độ đo chính là Jaccard và Kulczynski, từ các CSDL phân cấp dựa trên một cây loại tương ứng. Thuật toán GFCP được xây dựng dựa trên việc mở rộng thuật toán DCI-Closed [14], để có thể đạt được các yêu cầu sau:

- Có khả năng hoạt động trên các CSDL phân cấp.
- Áp dụng hai độ đo tương đồng là Jaccard và Kulczynski để xác định các mẫu đóng phổ biến tương đồng.
- GFCP chỉ xét các mẫu được kết hợp từ các mục trên cùng một mức trong taxonomy (multi-level).

Vì dựa trên nền DCI-Closed, **GFCP** thừa hưởng các đặc điểm của DCI-Closed: là thuật toán một pha, không phát sinh tập ứng viên; áp dụng chiến lược tìm kiếm theo chiều sâu để mở rộng các mẫu đang xét. Quá trình xét tính đóng dựa hoàn toàn vào thuật toán DCI-Closed. Thuật toán **GFCP** được xây dựng dựa trên ba nguyên lý chính:

- Thứ nhất thay vì chỉ xét các item riêng lẻ, mỗi item trong một giao dịch được mở rộng để bao gồm chính nó và tất cả các item cha của nó điều này giúp thuật toán phát hiện các mối quan hệ ở các mức độ trừu tượng khác nhau.
- Thứ hai, thay vì tìm kiếm tất cả các mẫu, GFCP sử dụng một độ đo tương đồng để hướng tới các mẫu có ý nghĩa. Một ngưỡng tương đồng tối thiểu (*minSim*) được sử dụng làm điều kiện để cắt tỉa không gian tìm kiếm.
- Thứ ba, để giảm thiểu sự dư thừa trong tập kết quả, thuật toán chỉ tập trung vào việc khai phá các mẫu đóng. Một mẫu được coi là đóng nếu không tồn tại một tập cha nào có cùng độ hỗ trợ. Điều này giúp cho kết quả trả về là một biểu diễn nhỏ gọn và đầy đủ của tất cả các mẫu phổ biến.

Thuật toán **GFCP** được thể hiện tại Thuật toán 1.

C. ĐÁNH GIÁ ĐỘ PHỨC TẠP THUẬT TOÁN GFCP

Phần này trình bày các đánh giá để nhận định độ phức tạp của thuật toán GFCP đã đề xuất. Độ đo tương đồng được áp dụng là Jaccard. Đối với một độ đo tương đồng khác, độ phức tạp tương ứng của độ đo đó sẽ được áp dụng. Trước tiên, một số các tham số sau cần phải được định nghĩa để thuận tiện trong việc đánh giá: $L-$ số mức của cấu trúc taxonomy; $n = |D|, m = |I|, l_{avg} -$ độ dài giao dịch trung bình; $f^l -$ số lượng mục đơn phổ biến biến tại mức $l \in \tau$; $p^l -$ số lượng mẫu tìm thấy tại mức l; $\overline{s} -$ độ hỗ trợ trung bình của các mục đơn. Thời gian thực hiện chính của thuật toán tập trung hoàn toàn vào Giai đoạn 2 (Thuật toán 1). Vì vậy ta sẽ tập trung phân tích độ phức tạp của giai đoạn này.

Giai đoạn 2 (khai thác đa mức):

a. Chi phí biến đổi giao dịch tại các mức taxonomy: $\boldsymbol{O(n \times l_{avg} \times L)}$

b. Giai đoạn 3 (Xử lý không gian tìm kiếm tại từng mức): $\boldsymbol{O(n \times l_{avg} + m + mlogm)}$

Tính support: $\boldsymbol{O(n \times l_{avg})}$

Lọc các mục phổ biến: $\boldsymbol{O(m + mlogm)}$, trong đó thao tác sắp xếp giả định là tốt nhất, $\boldsymbol{O(mlogm)}$

Giai đoạn 4 (Đệ quy): $\boldsymbol{R = O(2^f)}$ trong trường hợp xấu nhất, thực tế quá trình tỉa ứng viên sẽ làm giảm đáng kể chi phí trong giai đoạn này. Trung bình là $\boldsymbol{R = O(p \times logf)}$.

Giai đoạn 5 (Lọc tương đồng): $\boldsymbol{O\left(p \times \overline{k}^2 \times \overline{s}\right)}$

Trong đó, chi phí xác định độ tương đồng của một mẫu kích thước trung bình là $\boldsymbol{k}$: $\boldsymbol{O\left(\overline{k}^2 \times \overline{s}\right)}$

Vậy, độ phức tạp về thời gian trong trường hợp xấu nhất của GFCP có thể được xác định như sau:

$$T_{total} = O\left(\underbrace{L \times n \times l_{avg}}_{\substack{Database \\ transformation}} + \overbrace{L \times m log m}^{Sorting\ per\ level} + \underbrace{L \times \boldsymbol{p} \times \boldsymbol{logf}}_{\substack{DCI-Closed \\ mining}} + \overbrace{L \times p \times \overline{k}^2 \times \overline{s}}^{\substack{Similiarity \\ calculation}}\right)$$

D. VÍ DỤ MINH HỌA

Để minh họa cho các thuật toán **GFCP**, CSDL giao dịch D tại Bảng 1 sẽ được sử dụng. Sau bước tổng quát hóa, biến đổi các mục chuyên biệt thành các mục tổng quát từ cây phân loại ta thu được kết quả là CSDL tổng quát hóa D_1 tại mức 1 của taxonomy, thể hiện trong Bảng 2.

Bảng 2. *CSDL tổng quát hóa D_1 tại mức trừu tượng thứ nhất*

TID	Giao dịch tổng quát
1	$\{fruit, drinks\}$
2	$\{fruit, drinks\}$
3	$\{fruit\}$
4	$\{fruit, drinks\}$
5	$\{fruit\}$

Các tham số khai thác được thiết lập như sau: ngưỡng $minSup = 60\% = 3$ và ngưỡng $minSim = 50\%$ sử dụng độ đo Jaccard. Với các ngưỡng đã cho, tại mức 0, ta có các mục sau là phổ biến: $\{táo\}$ và $\{cam\}$ với cùng độ hỗ trợ là 4. CSDL dọc tại mức 0 chứa $\{táo\}$ và $\{cam\}$ lần lượt là $VDB_0 = \{\langle táo, \{1,2,3,5\}\rangle;\ \langle cam, \{1,2,4,5\}\rangle\}$. Tương tự, $\{fruit\}$ có độ hỗ trợ là 5 và $\{drinks\}$ có độ hỗ trợ là 3, đều thỏa $minSup$. CSDL dọc tại mức 1 là $VDB_1 = \{\langle fruit, \{1,2,3,4,5\}\rangle; \langle drinks, \{1,2,4\}\rangle\}$.

- Quá trình khai thác tại mức 0: $\boldsymbol{Post} = \{\langle \boldsymbol{táo}, \boldsymbol{4}\rangle, \langle \boldsymbol{cam}, \boldsymbol{4}\rangle\}$.
- Xử lý tiền tố $\boldsymbol{prefix} = \{\boldsymbol{táo}\}$ với support 4: ta xét mục $\{\boldsymbol{cam}\}$ từ $\boldsymbol{Post}$, khi thêm $\{\boldsymbol{cam}\}$ vào $\boldsymbol{prefix}$, khi đó $\boldsymbol{prefix} = \{\boldsymbol{táo}, \boldsymbol{cam}\}$ với support $\boldsymbol{3 \geq minSup}$. Vậy tập mục này là tiềm năng, ta giữ $\{\boldsymbol{cam}\}$ để xét trong lần đệ quy tiếp theo. Không còn mục nào trong $\boldsymbol{Post}$ có thể sử dụng để mở rộng $\boldsymbol{prefix} = \{\boldsymbol{táo}\}$ với support 4, vậy $\{\boldsymbol{táo}\}$ là tập mục phổ biến đóng. Hơn nữa vì $|\boldsymbol{prefix}| = \boldsymbol{1}$, độ đo tương đồng Jaccard mặc nhiên thỏa. Ta đưa $\langle \boldsymbol{táo}, \boldsymbol{4}\rangle$ vào tập $\boldsymbol{R}$.
- Đệ quy để mở rộng $\boldsymbol{prefix} = \{\boldsymbol{táo}\}$ sử dụng mục tiềm năng $\{\boldsymbol{cam}\}$ trong lần đệ quy bên trên. Ta có $\boldsymbol{prefix} = \{\boldsymbol{táo}, \boldsymbol{cam}\}$, support $\boldsymbol{3 \geq minSup}$. Vì không còn mục nào trong $\boldsymbol{Post}$ có thể mở rộng $\boldsymbol{prefix}$, $\{\boldsymbol{táo}, \boldsymbol{cam}\}$ là tập mục phổ biến đóng. Độ tương đồng Jaccard của $\{\boldsymbol{táo}, \boldsymbol{cam}\}$ là $\boldsymbol{60\% \geq minSim}$. Ta đưa $\langle\{\boldsymbol{táo}, \boldsymbol{cam}\}, \boldsymbol{3}\rangle$ vào $\boldsymbol{R}$.
- Quá trình đệ quy sử dụng $\boldsymbol{prefix} = \{\boldsymbol{táo}\}$ kết thúc vì $\boldsymbol{Post} = \emptyset$.
- Xử lý tiền tố $\boldsymbol{prefix} = \{\boldsymbol{cam}\}$ với support 4. Tương tự, vì tập $\boldsymbol{Post} = \emptyset$ nên không còn mục nào có thể mở rộng $\boldsymbol{prefix} = \{\boldsymbol{cam}\}$, vậy $\{\boldsymbol{cam}\}$ là tập mục phổ biến đóng. Tương tự $\{\boldsymbol{táo}\}$, vì $|\boldsymbol{prefix} = \{\boldsymbol{cam}\}| = \boldsymbol{1}$, $\{\boldsymbol{cam}\}$ thỏa độ đo tương đồng Jaccard. Ta đưa $\langle \boldsymbol{cam}, \boldsymbol{4}\rangle$ vào tập $\boldsymbol{R}$.

- Vậy tại mức 0, ta có các mẫu phổ biến đóng tương đồng sau: $\{\boldsymbol{L_0} = \{\langle \boldsymbol{t\acute{a}o}, \boldsymbol{4}\rangle;\ \langle \{\boldsymbol{t\acute{a}o}, \boldsymbol{cam}\}, \boldsymbol{3}\rangle;\ \langle \boldsymbol{cam}, \boldsymbol{4}\rangle\}\}$
- Quá trình khai thác tại mức 1: $\boldsymbol{Post} = \{\langle \boldsymbol{drinks}, \boldsymbol{3}\rangle, \langle \boldsymbol{fruits}, \boldsymbol{5}\rangle\}$
- Xử lý tiền tố $\boldsymbol{prefix} = \{\boldsymbol{drinks}\}$ với support = 3: kiểm tra bao đóng khi kết hợp với $\{\boldsymbol{fruits}\}$, ta có $\boldsymbol{TIDset}(\{\boldsymbol{drinks}\}) = \{\boldsymbol{1}, \boldsymbol{2}, \boldsymbol{4}\} \subset \boldsymbol{TIDset}(\{\boldsymbol{fruits}\}) = \{\boldsymbol{1}, \boldsymbol{2}, \boldsymbol{3}, \boldsymbol{4}, \boldsymbol{5}\}$. Vậy $\{\boldsymbol{drinks}\}$ không phải là tập đóng mặc dù phổ biến. Hơn nữa, do không còn mục nào trong $\boldsymbol{Post}$ có thể mở rộng $\{\boldsymbol{drinks}, \boldsymbol{fruits}\}$, $\{\boldsymbol{drinks}, \boldsymbol{fruits}\}$ trở thành tập phổ biến đóng với support 3. Độ đo Jaccard của $\{\boldsymbol{drinks}, \boldsymbol{fruits}\}$ là $\boldsymbol{60\%} \geq \boldsymbol{minSim}$, khi đó ta đưa $\langle \{\boldsymbol{drinks}, \boldsymbol{fruits}\}, \boldsymbol{3}\rangle$ vào $\boldsymbol{R}$.
- Xử lý tiền tố $\boldsymbol{prefix} = \{\boldsymbol{fruits}\}$. Thực hiện tương tự ta có $\langle \boldsymbol{fruits}, \boldsymbol{5}\rangle$ là một tập phổ biến đóng và thỏa độ đo tương đồng Jaccard. Ta đưa $\langle \boldsymbol{fruits}, \boldsymbol{5}\rangle$ vào $\boldsymbol{R}$.
- Vậy tại mức 1, ta có các mẫu phổ biến đóng tương đồng sau: $\{\boldsymbol{L_1} = \{\langle \{\boldsymbol{drinks}, \boldsymbol{fruits}\}, \boldsymbol{3}\rangle;\ \langle \boldsymbol{fruits}, \boldsymbol{5}\rangle\ \}\}$
- Thuật toán kết thúc, ta có tập $\boldsymbol{R}$ đầy đủ như sau: $\boldsymbol{R} = \begin{Bmatrix} \{\boldsymbol{L_0} = \{\langle \boldsymbol{t\acute{a}o}, \boldsymbol{4}\rangle;\ \langle \{\boldsymbol{t\acute{a}o}, \boldsymbol{cam}\}, \boldsymbol{3}\rangle;\ \langle \boldsymbol{cam}, \boldsymbol{4}\rangle\}\}, \\ \{\boldsymbol{L_1} = \{\langle \{\boldsymbol{drinks}, \boldsymbol{fruits}\}, \boldsymbol{3}\rangle;\ \langle \boldsymbol{fruits}, \boldsymbol{5}\rangle\ \}\} \end{Bmatrix}$

V. THỰC NGHIỆM VÀ KẾT QUẢ

A. MÔI TRƯỜNG VÀ THIẾT LẬP THỰC NGHIỆM

Để đánh giá hiệu quả của phương pháp đề xuất, chúng tôi tiến hành một loạt các thực nghiệm trên nhiều bộ dữ liệu khác nhau. Mục tiêu của chương này là phân tích hiệu năng của thuật toán về thời gian thực thi, mức độ sử dụng bộ nhớ, và đánh giá số lượng mẫu tương đồng phổ biến đóng được sinh ra dưới các tham số khác nhau. Các thực nghiệm được thực hiện trên môi trường máy tính CPU Intel® Core™ i7-7600U @ 2.80 GHz, 8GB RAM. Chúng tôi sử dụng ba bộ dữ liệu phổ biến trong lĩnh vực khai phá dữ liệu với các đặc điểm khác nhau: Fruithut, Mushroom, và Product. Chi tiết các CSDL được cho tại Bảng 3. Các CSDL này được cung cấp tại thư viện Khai thác Dữ liệu mã nguồn mở SPMF [24]cùng với taxonomy tương ứng*.

Các thí nghiệm khảo sát hai khía cạnh chính: thứ nhất so sánh hiệu năng triển khai đo lường sự khác biệt về tốc độ và bộ nhớ. Cách thực hiện phép so sánh này là đánh giá hai phiên bản của thuật toán **GFCP** sử dụng cấu trúc **Bitset** và **List**. Thuật toán cơ sở để đối sánh là **CoGAR-C**, phiên bản mở rộng của **CoGAR** [25] để khai thác mẫu tương đồng phổ biến đóng. Thứ hai đánh giá ảnh hưởng của độ đo tương đồng bằng cách phân tích số lượng mẫu kết quả được tạo ra bởi hai độ đo khác nhau là Jaccard và Kulczynski. Thực nghiệm được tiến hành với 2 kịch bản:

- Kịch bản 1: Giữ nguyên ngưỡng tương đồng *minSim* = 50% và thay đổi ngưỡng hỗ trợ tối thiểu minSup.
- Kịch bản 2: Giữ nguyên minSup ở một ngưỡng cố định thấp cho mỗi bộ dữ liệu và thay đổi ngưỡng tương đồng *minSim*.

Tất cả các thuật toán, GFCP và CoGAR-C, đều được triển khai bằng ngôn ngữ lập trình Java sử dụng JDK17.

Bảng 3. *Đặc điểm của các CSDL thực nghiệm*

CSDL	#Giao dịch (1)	#Mục (2)	#Mức taxonomy (3)	Độ dài giao dịch trung bình (4)	Mật độ (5) = (4) / (2)
Mushroom	8,416	119	3	23.00	1.93E-01
Fruithut	181,970	1,265	4	3.59	2.86E-03
Product	54,537	1,559	6	8.22	5.27E-03

B. ĐÁNH GIÁ HIỆU NĂNG

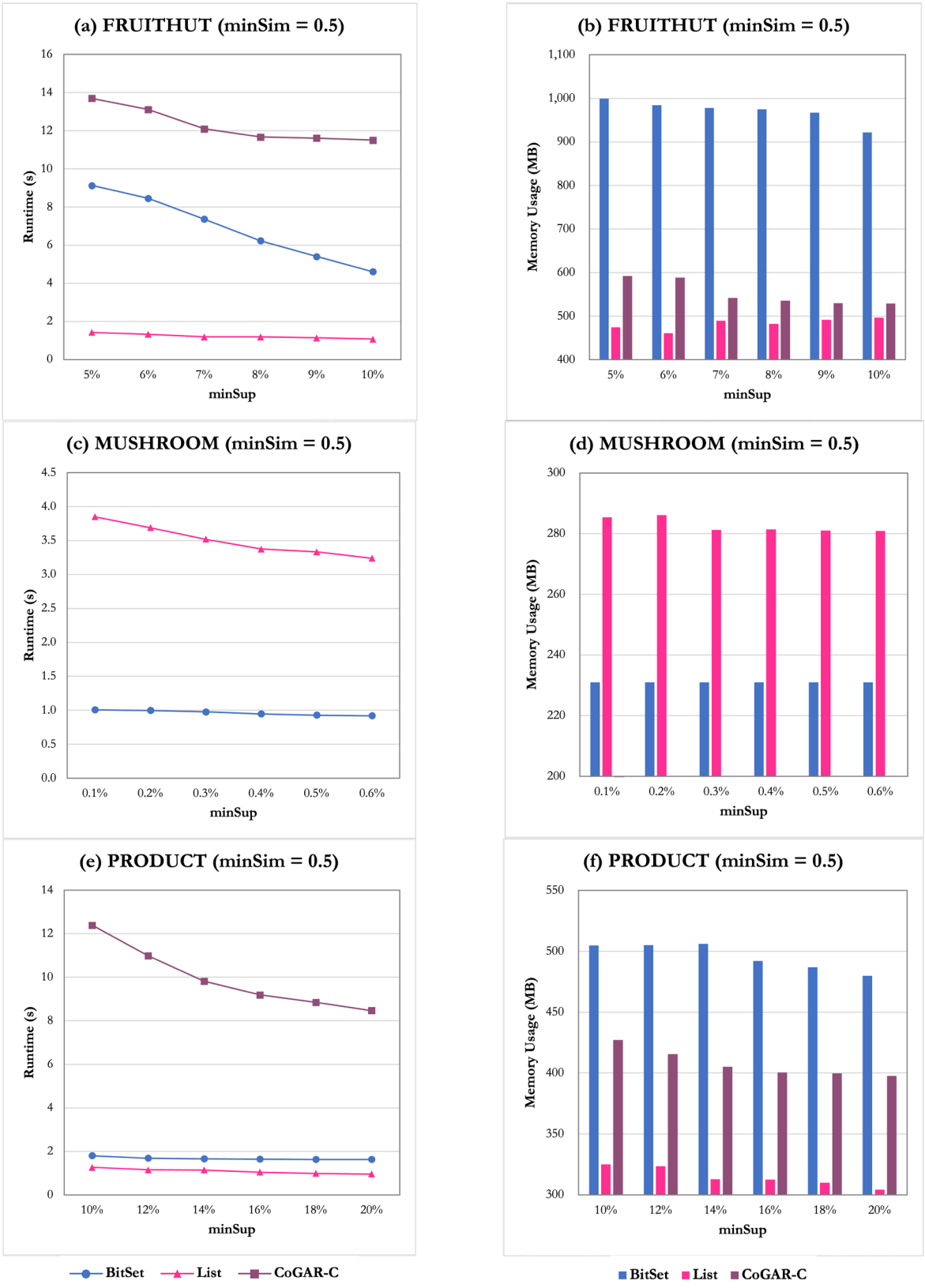
Chúng tôi tiến hành đánh giá hiệu quả của các thuật toán GFCP-BitSet, GFCP-List và CoGAR-C trên ba bộ CSDL chuẩn với các đặc tính khác nhau: Fruithut, Mushroom và Product. Hai loạt thực nghiệm được thiết kế: (1) thay đổi ngưỡng hỗ trợ tối thiểu ($minSup$) và giữ cố định ngưỡng tương đồng ($minSim = 50\%$) (Hình 2); (2) thay đổi $minSim$ và giữ cố định $minSup$. Các tiêu chí đánh giá chính là thời gian thực thi và bộ nhớ sử dụng (Hình 3).

Đối với thời gian thực thi, phân tích thời gian thực thi cho thấy hiệu suất của các biến thể GFCP phụ thuộc mạnh mẽ vào đặc tính (độ dày/thưa) của CSDL. Cụ thể như sau.

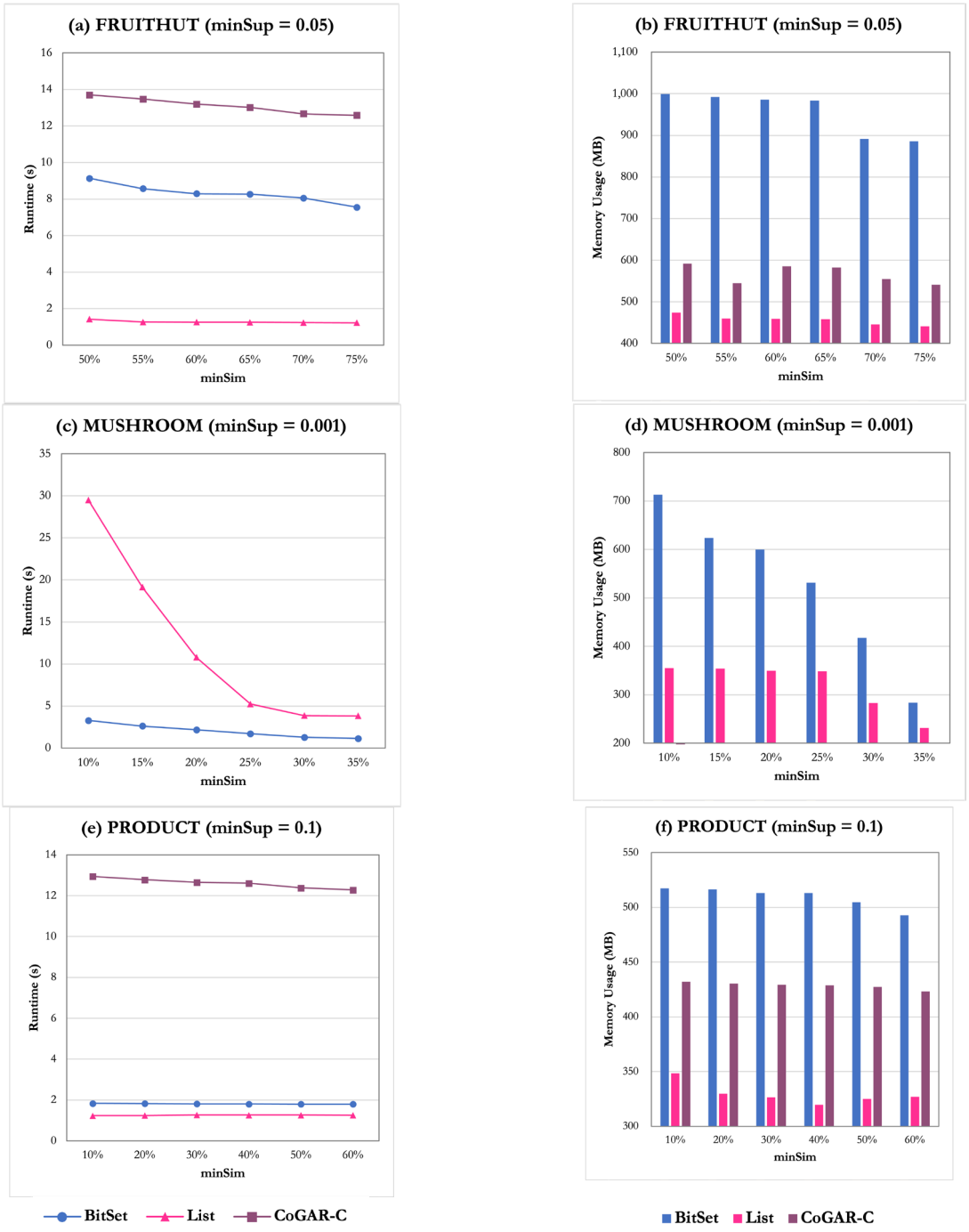
- So sánh với CoGAR-C: Trên tất cả các CSDL và mọi kịch bản (thay đổi $\boldsymbol{minSup}$ hoặc $\boldsymbol{minSim}$), CoGAR-C đều cho thấy thời gian thực thi chậm nhất một cách đáng kể (Hình 2, 3). Đáng chú ý nhất, CoGAR-C thất bại trong việc xử lý CSDL Mushroom, vượt quá giới hạn thời gian 3600 giây (Hình 2c và 2d, Hình 3c và 3d). Điều này cho thấy CoGAR-C có khả năng mở rộng kém và không phù hợp cho các CSDL phức tạp hoặc dày đặc.
- So sánh GFCP-BitSet và GFCP-List: Trên CSDL dày (Mushroom): GFCP-BitSet thể hiện sự vượt trội tuyệt đối. Khi thay đổi minSim (Hình 3c, minSim=10%), BitSet chỉ mất 3.28 giây, trong khi List cần tới 29.48 giây (gấp 9 lần). Tương tự khi thay đổi minSup (Hình 2c), BitSet (~0.9-1 giây) nhanh hơn đáng kể so với List (~3.2-3.8 giây). Điều này là do các phép toán bit cực kỳ hiệu quả khi xử lý các tập mục dày đặc. Trên CSDL thưa (Fruithut

* https://github.com/dzutrinh/MLC-Miner/tree/main/taxonomy

& Product): GFCP-List lại cho thấy hiệu suất tốt hơn. Ví dụ, trên Fruithut (Hình 2a, minSup=5%), List (1.42 giây) nhanh hơn nhiều so với BitSet (9.13 giây). Trên Product (Hình 2e, minSup=10%), List (1.26 giây) cũng nhanh hơn BitSet (1.80 giây).



Hình 2. *Thời gian thực thi và mức độ sử dụng bộ nhớ trên các CSDL trong Bảng 3 (cố định minSim, thay đổi minSup).*



Hình 3. *Thời gian thực thi và mức độ sử dụng bộ nhớ trên các CSDL trong Bảng 3 (cố định minSup, thay đổi minSim).*

Tóm lại, cả hai biến thể GFCP đều vượt trội so với baseline CoGAR-C. GFCP-BitSet là lựa chọn tối ưu về tốc độ cho các CSDL dày, trong khi GFCP-List hiệu quả hơn cho các CSDL thưa.
Đối với yếu tố về mức độ sử dụng bộ nhớ, phân tích bộ nhớ cho thấy một sự đánh đổi rõ ràng giữa hai biến thể GFCP.

- Hiệu quả của GFCP-List: Trong hầu hết các thực nghiệm trên CSDL Fruithut và Product, GFCP-List là thuật toán tiết kiệm bộ nhớ nhất. Ví dụ, trên Fruithut (Hình 2b, 3b), List chỉ tiêu thụ trung bình ~470MB, so với ~980MB của BitSet và ~560MB của CoGAR-C. Tương tự trên Product (Hình 2f, 3f), List (~315MB) hiệu quả hơn BitSet (~490MB) và CoGAR-C (~400MB).
- Trường hợp đặc biệt (Mushroom): Một ngoại lệ thú vị xảy ra trên CSDL Mushroom (Hình 2d, 3d). GFCP-BitSet (~231MB) lại sử dụng ít bộ nhớ hơn GFCP-List (~280-286MB). Điều này củng cố nhận định rằng cấu trúc

BitSet phù hợp và được tối ưu hóa tốt hơn cho dữ liệu dày, không chỉ về thời gian mà còn về không gian lưu trữ trong trường hợp này.

Về tổng thể, GFCP-List là lựa chọn an toàn và hiệu quả nhất về bộ nhớ cho các CSDL thưa và trung bình. GFCP-BitSet tiêu thụ nhiều bộ nhớ hơn đáng kể trên CSDL thưa, nhưng lại hiệu quả bất ngờ trên CSDL dày (Mushroom). Tóm lại, thực nghiệm cho thấy không có một biến thể nào là tốt nhất cho mọi tình huống. GFCP-List nổi bật với khả năng tiết kiệm bộ nhớ và tốc độ nhanh trên CSDL thưa. Ngược lại, GFCP-BitSet là lựa chọn chuyên biệt, cung cấp tốc độ vượt trội trên CSDL dày (Mushroom), nhưng phải đánh đổi bằng việc tiêu thụ nhiều bộ nhớ hơn trên các CSDL thưa. Cả hai đều chứng minh hiệu quả và khả năng mở rộng vượt xa baseline CoGAR-C.

ĐÁNH GIÁ SỐ LƯỢNG MẪU KẾT QUẢ

Trong phần này chúng tôi đánh giá số lượng mẫu tương đồng phổ biến đóng được sinh ra, so sánh giữa hai độ đo Jaccard và Kulczynski.

Kết quả thực nghiệm trên cả hai kịch bản trong Bảng 3 và 4 đều chỉ ra rằng độ đo Kulczynski luôn sinh ra số lượng mẫu nhiều hơn đáng kể so với Jaccard. Trên bộ dữ liệu Mushroom với $minSup = 0.1\%$, Jaccard chỉ tìm thấy 821 mẫu trong khi Kulczynski phát hiện tới 12,152 mẫu. Điều này cho thấy hệ số Kulczynski là độ đo "thoáng" hơn. Xu hướng của nó là coi các cặp mẫu là tương đồng ngay cả khi kích thước của chúng chênh lệch lớn, miễn là một mẫu là tập con của mẫu kia.

Ngược lại, chỉ số Jaccard chặt chẽ hơn và đòi hỏi sự tương đồng cao về cả thành phần và kích thước. Phần tiếp theo khi so sánh sự ảnh hưởng của tham số *minSup* và *minSim* chúng tôi thấy rằng: Khi tăng *minSup*, số lượng mẫu kết quả của cả hai độ đo đều giảm. Điều này là tất yếu vì việc tăng ngưỡng hỗ trợ sẽ làm giảm số lượng mẫu đóng phổ biến ban đầu, vốn là đầu vào cho giai đoạn lọc tương đồng. Hoàn toàn tương tự, khi tăng $minSim$, số lượng mẫu cuối cùng cũng giảm đi. Một ngưỡng tương đồng cao hơn sẽ loại bỏ nhiều mẫu hơn, giúp tập kết quả trở nên cô đọng. Trên bộ dữ liệu hiệu ứng này càng rõ hơn cho thấy việc điều chỉnh $minSim$ là một công cụ mạnh mẽ để kiểm soát kích thước của tập kết quả đầu ra.

Bảng 4. *Số lượng mẫu tương đồng phổ biến sử dụng độ đo tương đồng Jaccard và Kulczynski đối với các CSDL trong Bảng 3 (cố định ngưỡng minSim, thay đổi ngưỡng minSup).*

FRUITHUT			**MUSHROOM**			**PRODUCT**		
minSup	**Jaccard**	**Kulczynski**	**minSup**	**Jaccard**	**Kulczynski**	**minSup**	**Jaccard**	**Kulczynski**
5%	37	40	0.1%	821	12,152	10%	69	115
6%	31	34	0.2%	821	12,152	12%	51	93
7%	27	30	0.3%	821	12,152	14%	40	74
8%	22	25	0.4%	821	12,152	16%	32	61
9%	19	22	0.5%	821	12,152	18%	29	57
10%	17	20	0.6%	821	12,151	20%	27	55

Bảng 5. *Số lượng mẫu tương đồng phổ biến sử dụng độ đo tương đồng Jaccard và Kulczynski đối với các CSDL trong Bảng 3 (cố định ngưỡng minSup, thay đổi ngưỡng minSim).*

FRUITHUT			**MUSHROOM**			**PRODUCT**		
minSim	**Jaccard**	**Kulczynski**	***minSim***	**Jaccard**	**Kulczynski**	***minSim***	**Jaccard**	**Kulczynski**
50%	37	40	10%	150,436	199,195	10%	199	199
55%	37	38	15%	112,449	185,343	20%	182	199
60%	37	37	20%	77,147	158,391	30%	112	199
65%	37	37	25%	42,934	132,125	40%	73	184
70%	37	37	30%	15,074	106,548	50%	69	115
75%	37	37	35%	6,681	89,955	60%	68	78

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo này đã trình bày GFCP (Generalized Frequent Closed Pattern miner), một phương pháp hiệu quả để khai phá tập mẫu phổ biến đóng cô đọng từ cơ sở dữ liệu (CSDL) phân cấp. Bằng cách kết hợp ba giai đoạn lọc—loại bỏ dư thừa cấu trúc (sử dụng mẫu đóng), dư thừa ngữ nghĩa (dựa trên phân cấp) và lọc tương đồng (sử dụng độ đo Jaccard và Kulczynski)—phương pháp của chúng tôi đã giải quyết thành công vấn đề bùng nổ số lượng mẫu kết quả, vốn là một thách thức lớn của các thuật toán truyền thống. Kết quả thực nghiệm toàn diện trên các bộ CSDL

chuẩn (Mushroom, Fruithut, Product) đã khẳng định tính hiệu quả và khả năng mở rộng của GFCP. Các biến thể GFCP (BitSet và List) đều cho thấy hiệu suất vượt trội so với thuật toán baseline CoGAR-C, đặc biệt CoGAR-C đã thất bại trong việc xử lý CSDL dày (Mushroom) do giới hạn về thời gian.
Nghiên cứu cũng chỉ ra một sự đánh đổi quan trọng:

- GFCP-BitSet tỏ ra cực kỳ hiệu quả về mặt thời gian trên CSDL dày.
- GFCP-List lại nhanh hơn và tiết kiệm bộ nhớ đáng kể trên các CSDL thưa.

Phát hiện này cung cấp một lựa chọn linh hoạt cho người dùng, cho phép họ chọn biến thể thuật toán phù hợp nhất dựa trên đặc tính của CSDL đầu vào để đạt được sự cân bằng tối ưu giữa tốc độ và tài nguyên. Dựa trên những kết quả đã đạt được, chúng tôi đề xuất một số hướng nghiên cứu tiếp theo để mở rộng và cải tiến phương pháp: Hiệu suất của thuật toán, đặc biệt là GFCP-BitSet trên CSDL thưa hoặc GFCP-List trên CSDL dày, vẫn có thể được cải thiện. Một hướng đi rõ ràng là thiết kế và triển khai một phiên bản GFCP song song sử dụng mô hình đa luồng. Việc song song hóa các tác vụ tính toán độ tương đồng hoặc các phép toán giao tập hợp có thể giảm đáng kể tổng thời gian thực thi trên các hệ thống đa lõi, cho phép xử lý các bộ CSDL quy mô rất lớn. Dữ liệu trong thế giới thực thường mang tính không chắc chắn (ví dụ: dữ liệu cảm biến, chẩn đoán y khoa). Chúng tôi dự định mở rộng GFCP để xử lý CSDL không chắc chắn để có thể khai phá các tri thức hữu ích trong bối cảnh dữ liệu không rõ ràng.

VII. LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi Trường Đại học Ngoại ngữ – Tin học Thành phố Hồ Chí Minh (HUFLIT) theo đề tài mã số H2024-03. Nhóm tác giả xin trân trọng cảm ơn sự hỗ trợ quý báu này, đã tạo điều kiện thuận lợi để nghiên cứu được hoàn thành.

VIII. TÀI LIỆU THAM KHẢO

[1] R. Agrawal, T. Imieliński, and A. Swami (1993), "Mining association rules between sets of items in large databases," ACM SIGMOD Record, vol. 22, no. 2, pp. 207–216.
[2] H. P. D. W. T. Leitgöb (2023), "Editorial: Big data and machine learning in sociology," Frontiers in Sociology, vol. 8, pp. 1173155–1173160.
[3] F. Behrad and M. Saniee Abadeh (2022), "An overview of deep learning methods for multimodal medical data mining," Expert Syst Appl, vol. 200, p. 117006, doi: 10.1016/J.ESWA.2022.117006.
[4] X. Guo et al (2023), "Deep seabed mining: Frontiers in engineering geology and environment," International Journal of Coal Science & Technology 10:1, vol. 10, no. 1, pp. 1–31, Apr. 2023, doi: 10.1007/S40789-023-00580-X.
[5] T. Li, A. Rezaeipanah, and E. S. M. Tag El Din (2022), "An ensemble agglomerative hierarchical clustering algorithm based on clusters clustering technique and the novel similarity measurement," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 6, pp. 3828–3842, doi: 10.1016/J.JKSUCI.2022.04.010.
[6] Y. Luo et al. (2022), "MOF Synthesis Prediction Enabled by Automatic Data Mining and Machine Learning**," Angewandte Chemie International Edition, vol. 61, no. 19, p. e202200242, doi: 10.1002/ANIE.202200242.
[7] A. Y. Rodríguez-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and J. Ruiz-Shulcloper (2023), "Mining frequent patterns and association rules using similarities," Expert Syst Appl, vol. 40, no. 17, pp. 6823–6836, doi: 10.1016/J.ESWA.2013.06.041.
[8] R. Agrawal and R. Srikant (1994), "Fast Algorithms for Mining Association Rules in Large Databases," in 20th International Conference on Very Large Data Bases (VLDB '94), Morgan Kaufmann Publishers Inc., pp. 487–499–487–499.
[9] M. J. Zaki (2000), "Scalable algorithms for association mining," IEEE Trans Knowl Data Eng, vol. 12, no. 3, pp. 372–390.
[10] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang (2001), "H-mine: Hyper-structure mining of frequent patterns in large databases," Proceedings - IEEE International Conference on Data Mining, ICDM, pp. 441–448, doi: 10.1109/ICDM.2001.989550.
[11] P. Fournier-Viger et al. (2022), "Pattern Mining: Current Challenges and Opportunities," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13248 LNCS, pp. 34–49, doi: 10.1007/978-3-031-11217-1_3.
[12] S., S. S. and D. K., Prabha (2013), "A survey on closed frequent pattern mining," Int J Comput Appl, vol. 63, no. 14.
[13] J. H. R. M. J Pei (2000), "Closet: An efficient algorithm for mining frequent closed itemsets," Acm sigmod workshop on research issues in data mining and knowledge discovery, vol. 4, pp. 21–32.
[14] C. Lucchese, S. Orlando, and R. Perego (2006), "Fast and memory efficient mining of frequent closed itemsets," IEEE Trans Knowl Data Eng, vol. 18, no. 1, pp. 21–36, Jan, doi: 10.1109/TKDE.2006.10.

[15] M. J. Zaki and C.-J. Hsiao (2002), "CHARM: An Efficient Algorithm for Closed Itemset Mining," in 2002 SIAM International Conference on Data Mining (SDM), pp. 457–473.
[16] D. A. U. N. V. S. R. Jashma Suresh Ponmudiyan Poovan (2023), "A genetic algorithm coupled with tree-based pruning for mining closed association rules," International Journal of Electrical and Computer Engineering (IJECE), vol. 13, no. 3, pp. 2876–2890.
[17] T. L. Dam, K. Li, P. Fournier-Viger, and Q. H. Duong (2019), "CLS-Miner: efficient and effective closed high-utility itemset mining," Front Comput Sci, vol. 13, no. 2, pp. 357–381.
[18] S. S. S. A. K. L. A. K. and B. Kuldeep Singh (2021), "Mining of Closed High Utility Itemsets: A Survey," Recent Advances in Computer Science and Communications, vol. 14, no. 1, pp. 6–12.
[19] T. D. D. Nguyen, N. T. Tung, L. T. T. Nguyen, T. T. Pham, and B. Vo (2024), "MLC-miner: Efficiently discovering multi-level closed high utility patterns from quantitative hierarchical transaction databases," Expert Syst Appl, vol. 254, p. 124383, doi: 10.1016/j.eswa.2024.124383.
[20] M. M. Deza and E. Deza (2009), "Encyclopedia of Distances," in Encyclopedia of Distances, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–583. doi: 10.1007/978-3-642-00234-2_1.
[21] A. Levy, B. R. Shalom, and M. Chalamish (2025), "A guide to similarity measures and their data science applications," J Big Data, vol. 12, no. 1, p. 188, doi: 10.1186/s40537-025-01227-1.
[22] L. Cagliero, S. Chiusano, P. Garza, and G. Ricupero (2017), "Discovering High-Utility Itemsets at Multiple Abstraction Levels," pp. 224–234. doi: 10.1007/978-3-319-67162-8_22.
[23] L. Cagliero, S. Chiusano, P. Garza, and G. Ricupero (2017), "Discovering high-utility itemsets at multiple abstraction levels," in European Conference on Advances in Databases and Information Systems, M. Kirikova, K. Nørvåg, G. A. Papadopoulos, J. Gamper, R. Wrembel, J. Darmont, and S. Rizzi, Eds., Cham: Springer International Publishing, pp. 224–234.
[24] P. Fournier-Viger et al. (2016), "The SPMF Open-Source Data Mining Library Version 2," pp. 36–40. doi: 10.1007/978-3-319-46131-1_8.
[25] E. Baralis, L. Cagliero, T. Cerquitelli, and P. Garza (2012), "Generalized association rule mining with constraints," Inf Sci (N Y), vol. 194, pp. 68–84, doi: 10.1016/j.ins.2011.05.016.

MINING FREQUENT SIMILARITY PATTERNS FROM TAXONOMY-BASED DATASETS

Trần Cẩm Tú, Phạm Đức Thành

Faculty of Information Technology, Ho Chi Minh City University of Foreign Languages - Information Technology

tutc@huflit.edu.vn, phamducthanh@huflit.edu.vn

ABSTRACT— Traditional frequent pattern mining often generates a large number of redundant patterns. This study proposes GFCP (Generalized Frequent Closed Pattern miner), a multi-stage filtering method for mining a concise set of patterns from hierarchical databases. This method combines closed pattern mining (to eliminate structural redundancy) with similarity-based filtering (using Jaccard and Kulczynski) and semantic redundancy pruning within the hierarchical database. Comprehensive experiments, varying the minimum support (minSup) and minimum similarity (minSim) thresholds, were conducted on standard datasets. The results demonstrate that the GFCP variants (BitSet and List) significantly outperform the CoGAR-C baseline, which failed to complete execution on the Mushroom dataset (exceeding 3600 seconds). Specifically, GFCP-BitSet achieves superior runtime performance on dense datasets (Mushroom), while GFCP-List is faster on sparse datasets (Fruithut, Product) and demonstrates greater memory efficiency in most scenarios. This research confirms that GFCP is an effective and scalable solution, offering a flexible trade-off between speed and memory depending on the input data structure.

Keywords — similarity measures, Jaccard, Kulczynski, correlated frequent closed pattern, frequent closed pattern, hierarchical database.

Trần Cẩm Tú là Thạc sĩ Khoa học máy tính từ năm 2017 tại Trường Đại học Khoa học tự nhiên, Đại học Quốc gia TP.HCM. Hiện cô là giảng viên của Khoa Công nghệ thông tin Trường Đại học Ngoại ngữ-Tin học Tp. Hồ Chí Minh. Lĩnh vực nghiên cứu quan tâm: khai thác dữ liệu, khai thác luật kết hợp, phân lớp.

Phạm Đức Thành nhận học vị Thạc sĩ Công nghệ thông tin năm 2006 tại Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh; hiện là giảng viên công tác tại Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ-Tin học Tp. Hồ Chí Minh. Lĩnh vực nghiên cứu quan tâm: khai thác dữ liệu, tối ưu hóa.