Khai thác dữ liệu chuỗi theo mối quan tâm của người dùng

VĂN THỊ THIÊN TRANG *
Trường Đại học Kinh tế - Tài chính Thành phố Hồ Chí Minh
Ngày nhận: 07/09/2023 - Ngày chỉnh sửa: 26/09/2023 - Duyệt đăng: 09/10/2023
(*) Liên hệ: trangvtt@uef.edu.vn

Tóm tắt:

Khai thác dữ liệu chuỗi hay còn gọi là khai thác mẫu tuần tự là đi tìm những chuỗi con xuất hiện phổ biến (gọi là mẫu tuần tự) trong cơ sở dữ liệu chuỗi, ngưỡng phổ biến này do người dùng quy định. Trong những năm gần đây, do sự bùng nổ thông tin và dữ liệu lớn, bài toán này có xu hướng phát triển thành khai thác mẫu tuần tự có ràng buộc nhằm khắc phục cả hai thách thức về tính hiệu quả và hiệu suất thực thi vì ràng buộc đại diện cho mối quan tâm của người dùng. Bài báo này trình bày khảo sát chi tiết tất cả các phương pháp khai thác mẫu tuần tự và các loại ràng buộc đã được nghiên cứu. Phân loại các phương pháp khai thác, đồng thời phân tích ưu nhược điểm của chúng, từ đó chỉ ra hướng tiếp cận và phương pháp làm nền tảng cho các nghiên cứu về sau của bài toán này.

Từ khóa: Cơ sở dữ liệu chuỗi, khai thác dữ liệu, khai thác mẫu tuần tự, mẫu tuần tự phổ biến, ràng buộc.

Abstract:

Sequence data mining, also known as sequential pattern mining, is to find all frequent sub-sequences (called sequential patterns) in a sequence database, the threshold of frequency is specified by the user. In recent years, with the explosion growth of information and big data, this problem trends toward mining with constraints to overcome both effectiveness and efficiency challenges since that the constraints represent for the user's interest. This paper presents a detailed survey of recent studies on mining sequential pattern and the categories of the constraints. Moreover, it also classifies the main methods and analyzes their advantages and limitations, thereby main approaches and strategies to solve sequential pattern mining problems in the future are presented.

Keywords: Sequence database, data mining, sequential pattern mining, frequent sequential patterns, constraint.

1. Giới thiệu

Dữ liệu chuỗi là dữ liệu biểu diễn dưới dạng dãy sự kiện theo thứ tự thời gian, là loại dữ liệu quan trọng xuất hiện khá phổ biến trong nhiều ứng dụng khoa học, y tế, bảo mật, kinh doanh và các ứng dụng khác. Dữ liệu chuỗi chứa đựng những thông tin hữu ích, chẳng hạn chuỗi DNA mã hoá gen di truyền, chuỗi protein mô tả thành phần axit amin của protein và mã hoá cấu trúc và chức năng của protein. Ngoài ra, dữ liệu chuỗi có thể giúp nắm bắt thói quen hành xử của các cá thể như chuỗi lịch sử mua sắm của khách hàng, chuỗi truy cập web. Chuỗi còn có thể giúp mô tả cách thức các đơn vị hoạt động như chuỗi lịch sử bán hàng của một siêu thị, v.v... Việc khai thác dữ liệu chuỗi cung cấp các phương pháp và công cụ cần thiết để khám phá nguồn tri thức hữu ích tiềm ẩn trong các kho dữ liệu khổng lồ.

Từ chuỗi sự kiện, chúng ta có thể hiểu được cách thức các đối tượng hoạt động như thế nào, từ đó rút ra cách tốt nhất để giải quyết chúng. Dựa vào đặc điểm của các sự kiện trong chuỗi, có thể phân dữ liệu chuỗi ra thành hai loại: *Loại thứ nhất* là dạng chuỗi mà mỗi sự kiện trong chuỗi chỉ có một mục dữ liệu như chuỗi dữ liệu sinh học, chuỗi truy cập web (web log) v.v... *Loại*

thứ hai là mỗi sự kiện trong chuỗi là tập các mục dữ liệu như chuỗi giao dịch mua sắm của khách hàng, chuỗi lịch sử bán hàng, chuỗi các triệu chứng bệnh của bệnh nhân theo từng giai đoạn.

Khai thác chuỗi sự kiện hay còn gọi là khai thác mẫu tuần tự từ cơ sở dữ liệu (CSDL) chuỗi là lĩnh vực nghiên cứu ra đời từ năm 1995 được đề xuất đầu tiên bởi Agrawal và Srikant, thu hút nhiều nhà khoa học quan tâm và nghiên cứu rộng rãi. Khai thác mẫu tuần tự là bài toán khai thác mẫu khái quát nhất, với yêu cầu đi tìm những mẫu phổ biến là những chuỗi con trong CSDL chuỗi mà số lần xuất hiện của chúng lớn hơn ngưỡng phổ biến do người dùng chỉ ra. Khai thác mẫu tuần tự có ứng dụng rộng rãi và đa dạng như phân tích chuỗi gen sinh học, phân tích mối quan hệ giữa các tín hiệu mạng dưới dạng chuỗi tín hiệu viễn thông phổ biến, lấy thông tin từ các mẫu triệu chứng bệnh dùng cho chẩn đoán y khoa hoặc thuốc phòng bệnh, cải tiến cấu trúc siêu liên kết trong các trang web thương mại điện tử để gia tăng doanh thu nhờ vào khám phá các mẫu duyệt web của người dùng, dự đoán nhu cầu mua sắm của khách hàng...

Khảo sát các lĩnh vực ứng dụng, các nhà nghiên cứu đã nhận thấy rằng độ phổ biến không phải là độ đo tốt nhất để xác định ý nghĩa của một mẫu. Nếu sử dụng phương pháp khai thác mẫu theo ràng buộc truyền thống là độ phổ biến thì tập mẫu và luật tìm được thường rất đồ sộ, nhưng phần lớn chúng lại không có giá trị sử dụng. Do đó, một số nghiên cứu đưa vào các ràng buộc về độ hữu ích, ràng buộc về trọng số xuất hiện của sự kiện trong mẫu như khai thác mẫu hữu ích cao, mẫu có trọng số, mẫu tuần hoàn. Một số nghiên cứu khác tiến hành khai thác mẫu dựa trên các ràng buộc do người dùng yêu cầu tùy thuộc vào từng lĩnh vực ứng dụng như ràng buộc về item, về độ dài mẫu, về thời gian...

Khai thác mẫu tuần tự phổ biến dựa trên ràng buộc là khám phá các mẫu phổ biến bằng cách kết hợp các ràng buộc do người dùng chỉ ra vào quá trình khai thác. Khai thác dựa trên ràng buộc có thể khắc phục cả hai thách thức về hiệu quả và hiệu suất thực hiện vì ràng buộc đại diện cho những gì mà người dùng quan tâm và yêu cầu, nó giới hạn các mẫu tìm được chỉ là một tập hợp con của tập tất cả các mẫu tuần tự phổ biến, tập con này chỉ chứa các mẫu thỏa các ràng buộc thỏa yêu cầu người dùng. Do đó, người dùng dễ phân tích tập mẫu kết quả tìm được vì nó có số lượng ít hơn. Hơn nữa, nếu có thể đưa các ràng buộc vào trong quá trình khai thác mẫu thì sẽ đạt được hiệu suất cao hơn vì thực hiện tìm kiếm có tập trung hơn, giảm không gian tìm kiếm. Đây chính là động lực thúc đẩy nghiên cứu bài toán khai thác mẫu tuần tự dựa trên ràng buộc phát triển gần đây.

Trong bài báo này, chúng tôi thực hiện khảo sát và tổng quan cơ sở lý thuyết cũng như các phương pháp giải quyết bài toán khai thác mẫu tuần tự dựa trên các ràng buộc đã có từ trước đến nay. Cấu trúc của bài báo như sau: Phần 2 trình bày một số khái niệm và định nghĩa liên quan đến bài toán. Phần 3 chỉ ra các đặc trưng của các thuật toán đã có. Phân loại các phương pháp khai thác, chỉ ra ưu nhược điểm của từng phương pháp được trình bày ở phần 4. Cuối cùng, một số kết luận được trình bày ở phần 5.

Định nghĩa bài toán và các loại ràng buộc

Một số định nghĩa cơ sở

Cho tập $I = \{i_1, i_2, ..., i_m\}$ gồm m phần tử còn gọi là các item.

Itemset: Một itemset là một tập không có thứ tự khác rỗng, gồm các item thuộc tập I. Itemset X ký hiệu là $(i_1, i_2, ..., i_k)$ với mỗi i_j $(1 \le j \le k)$ là một item. Nhằm đơn giản trong ký hiệu, đối với những itemset chỉ có một item đơn thì không cần cặp dấu ngoặc. Itemset có k item, ký hiệu là *k-itemset*.

Chuỗi: Một chuỗi (sequence) là một danh sách có thứ tự các itemset. Chuỗi S được ký hiệu là $\langle s_1 s_2 ... s_n \rangle$ với mỗi s_i $(1 \le i \le n)$ là một itemset. Chiều dài của chuỗi là tổng số item có trong chuỗi. Chuỗi có chiều dài k còn được gọi là *chuỗi-k*. Kích thước của chuỗi là số lượng itemset có trong chuỗi. Ví dụ, chuỗi $S = \langle BE(AC)A(ABC) \rangle$ có 8 item, tức S có chiều dài là 8 và được gọi là *chuỗi-8*. S có kích thước là 5, gồm 5 itemset là B, E, (AC), A, và (ABC).

Chuỗi con, chuỗi cha: Chuỗi $S_a = \langle a_1 a_2 ... a_n \rangle$ được gọi là chuỗi con của chuỗi $S_b = \langle b_1 b_2 ... b_m \rangle$, và S_b là chuỗi cha của S_a, ký hiệu $S_a \subseteq S_b$ hay $S_b \supseteq S_a$ nếu tồn tại các số nguyên $1 \le j_1 < j_2 < ... < j_n \le m$ sao cho $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, ..., $a_n \subseteq b_{jn}$.

Cơ sở dữ liệu chuỗi: Cơ sở dữ liệu chuỗi

(Sequence Database) dùng để khai thác là một tập hợp các chuỗi dữ liệu đầu vào, kí hiệu là *SDB* = $\{S_1, S_2, ... S_n\}$. Mỗi chuỗi dữ liệu có dạng có dạng (*SID, S*), trong đó *SID (Sequence Identify)* là định danh của chuỗi và *S* là chuỗi các itemset. Ví dụ, Bảng 1 minh họa một CSDL gồm 5 chuỗi có các *SID* lần lượt là 1, 2, 3, 4, và 5. Chẳng hạn, những chuỗi này có thể đại diện cho các lần mua sắm của 5 khách hàng, tập các mặt hàng (item) phân biệt là {A, B, C}.

Bảng 1. CSDL chuỗi minh họa SDB

SID	Chuỗi dữ liệu
1	⟨(AB)BB(AB)B(AC)⟩
2	⟨(AB)(BC)(BC)⟩
3	⟨B(AB)⟩
4	⟨BB(BC)⟩
5	⟨(AB)(AB)(AB)A(BC)⟩

Mẫu tuần tự: Mẫu tuần tự là một chuỗi con của chuỗi trong CSDL.

Độ phổ biến: Độ phổ biến (*support*) của một mẫu *p* là số lượng chuỗi trong *SDB* có chứa *p*, ký hiệu, $Sup_{SDB}(p) = |\{S|S \in SDB \wedge p \subseteq S\}|$

kí hiệu là *sup(p)*.

Mẫu tuần tự phổ biến: Cho trước một ngưỡng phổ biến tối thiểu, do người dùng qui định, kí hiệu là *minSup, minSup* ∈ (0, 1]. Một mẫu *p* được gọi là phổ biến nếu *sup(p)* ≥ *minSup*, khi đó *p* được gọi là mẫu tuần tự phổ biến. Như vậy, *mẫu tuần tự phổ biến là một chuỗi con phổ biến* tìm được trong CSDL.

Khai thác mẫu tuần tự: Cho trước CSDL chuỗi *SDB* và ngưỡng phổ biến tối thiểu *minSup* do người dùng qui định trước, bài toán khai thác mẫu tuần tự là tìm tất cả các chuỗi con phổ biến hay mẫu tuần tự phổ biến có trong *SDB*.

Gọi *FP* là tập các mẫu tuần tự phổ biến trong *SDB*, ta có:

$FP = \{p \in SDB \mid sup(p) \geq minSup\}$.

Ví dụ:

Xét CSDL chuỗi *SDB* ở Bảng 1, chuỗi s_1 = ⟨(AB)BB(AB)B(AC)⟩ có 6 itemset là: (AB), B, B, (AB), B, (AC) và có 9 item. Vậy s_1 có kích thước là 6 và có độ dài là 9. Trong chuỗi s_1, item A xuất hiện ba lần nhưng nếu tính độ phổ biến thì độ phổ biến của item A chỉ được tính là 1 đối với chuỗi s_1.

Chuỗi p = ⟨(AB)(C)⟩ là một chuỗi con của chuỗi s_1, vì vậy chuỗi con p được gọi là mẫu. Trong CSDL, chỉ có chuỗi s_1, s_2 và s_5 có chứa mẫu p, vậy $sup(p)$ = 3. Ta có $|SDB|$ = 5, nếu lấy $minSup$ = 3, nghĩa là một mẫu được gọi là phổ biến khi mẫu đó phải xuất hiện ít nhất 3 lần trong *SDB*. Trong trường hợp này, ta được p là một mẫu tuần tự phổ biến.

Khai thác mẫu tuần tự trong *SDB* thu được *FP* = {⟨A⟩: 4, ⟨AB⟩: 3, ⟨ABB⟩: 3, ⟨ABC)⟩: 3, ⟨AC⟩: 3, ⟨(AB)⟩: 4, ⟨(AB)B⟩: 3, ⟨(AB)BB⟩: 3, ⟨(AB)BC⟩: 3, ⟨(AB)C⟩: 3, ⟨B⟩: 5, ⟨BA⟩: 3, ⟨B(AB)⟩: 3, ⟨BB⟩: 5, ⟨BBB⟩: 4, ⟨BB(BC)⟩: 3, ⟨BBC⟩: 4, ⟨B(BC)⟩: 3, ⟨BC⟩: 4, ⟨(BC)⟩: 3, ⟨C⟩: 4, với |*FP*| = 21.

Khai thác mẫu tuần tự dựa trên ràng buộc: Ràng buộc $\mathbb{C}$ trong khai thác mẫu tuần tự là một hàm Boolean $\mathbb{C}(p)$ trên các mẫu. Cho CSDL chuỗi *SDB*, ràng buộc $\mathbb{C}$ và ngưỡng phổ biến tối thiểu *minSup* do người dùng đưa ra. Bài toán khai thác mẫu tuần tự dựa trên ràng buộc là tìm tất cả các mẫu tuần tự phổ biến trong CSDL thỏa ràng buộc $\mathbb{C}$.

$FCP = \{p \in SDB \mid sup(p) \geq minSup \wedge \mathbb{C}(p) =$ true$\}$.

Các loại ràng buộc

Có thể xem xét và mô tả các ràng buộc từ nhiều góc độ khác nhau. Xét từ góc độ ứng dụng, dựa trên ngữ nghĩa và dạng thức của ràng buộc, Jian Pei và đồng sự (2007) đã khảo sát và đưa ra định nghĩa cho bảy loại ràng buộc xuất hiện phổ biến trong các lĩnh vực ứng dụng, bao gồm: ràng buộc item, ràng buộc độ dài, ràng buộc chuỗi con, ràng buộc kết hợp, ràng buộc biểu diễn dưới dạng biểu thức có quy tắc, ràng buộc về khoảng thời gian xảy ra của sự kiện đầu và cuối trong mẫu, ràng buộc về khoảng thời gian giữa hai sự kiện kề nhau trong mẫu. Mặc dù chưa hoàn toàn đầy đủ, nhưng hầu như đã khái quát nhiều ràng buộc thú vị trong các lĩnh vực ứng dụng.

Ràng buộc 1 (ràng buộc item – Item constraint): Ràng buộc item là ràng buộc yêu cầu một tập con item phải có mặt (hoặc không có mặt) trong mọi mẫu. Nó có dạng:

$C_{item}(p) \equiv (\varphi i: 1 \leq i \leq len(p), p[i]\ \theta\ V)$

Hoặc:

$C_{item}(p) \equiv (\varphi i: 1 \leq i \leq len(p), p[i] \cap V \neq \varnothing)$

Trong đó, V là tập con các item, $\varphi \in \{\forall, \exists\}$ và

$\theta \in \{\subseteq, \supseteq, \not\subset, \not\supset, \in, \notin\}$.

Ví dụ, trong khai thác mẫu truy cập web, người dùng có thể chỉ quan tâm tới những mẫu cho biết các lần vào cửa hàng sách trực tuyến. Gọi B là tập cửa hàng sách trực tuyến, khi đó ràng buộc item tương ứng là:

$C_{item}(p) \equiv (\forall i: 1 \le i \le len(p), \alpha[i] \subseteq B)$

Ràng buộc 2 (ràng buộc độ dài – Length constraint): Ràng buộc độ dài là ràng buộc yêu cầu về độ dài của mẫu. Độ dài ở đây có thể là số lần xuất hiện của các item hoặc số lượng itemset trong mỗi mẫu. Ràng buộc độ dài cũng có thể là yêu cầu về số lượng item phân biệt hoặc thậm chí là số item tối đa trong mỗi giao dịch của mẫu.

Ví dụ, người dùng chỉ muốn tìm những mẫu dài (chẳng hạn là những mẫu gồm ít nhất 50 itemset) trong phân tích chuỗi sinh học. Có thể biểu diễn yêu cầu này bởi ràng buộc độ dài: $C_{length}(p) \equiv (len(p) \ge 50)$.

Ràng buộc 3 (ràng buộc chuỗi con – Sub-pattern constraint): là ràng buộc yêu cầu mẫu phải chứa một trong số các chuỗi con thuộc một tập chuỗi cho trước. Ràng buộc chuỗi con có dạng:

$C_{sub\_pattern}(p) \equiv (\exists \gamma \in V: \gamma \subseteq p)$

Trong đó, V là một tập chuỗi cho trước.

Ví dụ, tìm những mẫu giao dịch có mua máy laptop trước sau đó mua máy ảnh kĩ thuật số.

Ràng buộc 4 (ràng buộc kết hợp – Aggregate constraint): là ràng buộc trong việc kết hợp các item trong mẫu, các hàm kết hợp có thể là min, max, sum, avg...

Ví dụ, một người phân tích thị trường muốn biết những mẫu giao dịch nào có giá trung bình của tất cả các item trong mẫu trên 100$.

Ràng buộc 5 (ràng buộc biểu thức có quy tắc – Regular expression constraint): là ràng buộc biểu diễn dưới dạng một biểu thức có quy tắc trên tập item, sử dụng các phép toán có quy tắc như phép nối rời và phép bao đóng Kleene.

Ví dụ, để tìm các mẫu tuần tự về một bệnh nhân nhận là mình bị bệnh sởi và bác sĩ đã cho một cách điều trị cụ thể, khi đó biểu thức có quy tắc có dạng *Nhận_là(Bệnh Sởi| Bệnh cùng họ với Sởi)(Điều trị A| Điều trị B| Điều trị C)*, trong đó *"|"* là phép tuyển (phép *OR*).

Ràng buộc 6 (ràng buộc khoảng thời gian xảy ra – Duration constraint): là ràng buộc chỉ áp dụng đối với các CSDL mà mỗi itemset trong chuỗi đều có kèm theo thời gian xảy ra. Ràng buộc này yêu cầu khoảng cách thời gian xảy ra giữa itemset đầu và cuối trong mẫu phải dài hoặc ngắn hơn một khoảng thời gian cho trước. Ràng buộc khoảng thời gian có dạng:

$C_{duration} \equiv Khoảng\_thời\_gian(p)\ \theta\ \Delta t$

Trong đó, $\theta \in \{\le, \ge\}$ và Δt là số nguyên cho trước. Một mẫu p thỏa ràng buộc này khi và chỉ khi $|\{\beta \in SDB | \exists 1 \le i_1 < \cdots < i_{len(p)} \le len(\beta)$ *sao cho* $p[1] \subseteq \beta[i_1], ..., p[len(p)] \subseteq \beta[i_{len(p)}]$, *và* $(\beta[i_{len(p)}].time - \beta[i_1].time)\ \theta\ \Delta t\}| \ge minsup$. Những thuật toán có đầu vào là chuỗi theo thời gian thì ràng buộc loại này thường cài đặt giới hạn thời gian trong phạm vi cửa sổ trượt *"sliding window"* trên các sự kiện của mẫu.

Ràng buộc 7 (ràng buộc khoảng thời gian ngắt quãng – Gap constraint): ràng buộc này cũng chỉ áp dụng đối với dữ liệu chuỗi mà mỗi itemset trong chuỗi có kèm theo thời gian xảy ra. Ràng buộc này yêu cầu độ lệch thời gian xảy ra giữa hai itemset (kề nhau hoặc trong phạm vi quy định) trong mẫu phải dài hơn hoặc ngắn hơn ngưỡng thời gian cho trước. Ràng buộc có dạng:

$C_{gap} \equiv Gap(p)\ \theta\ \Delta t$

Trong đó, $\theta \in \{\le, \ge\}$ và Δt là số nguyên cho trước. Ví dụ, mẫu giao dịch tuần tự $\langle A, t_1, B, t_2, C\rangle$ thỏa ràng buộc này có nghĩa là item A được mua trước, sau khoảng thời gian t_1 thì B được mua và cuối cùng sau khoảng thời gian t_2 thì C được mua, với $t_1, t_2 > \Delta t$ cho trước.

Trong số những ràng buộc trên, chỉ có các ràng buộc liên quan đến thời gian (ràng buộc 6 và 7) là ảnh hưởng đến độ phổ biến vì các ràng buộc này quy định cách thức so khớp một mẫu với một chuỗi trong CSDL. Do đó, loại ràng buộc này ảnh hưởng đến quá trình đếm độ phổ biến của mẫu. Còn đối với những loại ràng buộc khác, có thể xác định mẫu có thỏa ràng buộc hay không bằng chính mẫu đó mà không ảnh hưởng đến việc đếm độ phổ biến.

Đặc trưng của các thuật toán khai thác mẫu tuần tự

Khi phát triển một thuật toán để khai thác mẫu tuần tự từ CSDL chuỗi, yếu tố đại diện cho hiệu suất khai thác là chi phí bộ nhớ sử dụng và tốc độ xử lý dữ liệu. Do đó, phải sử dụng cấu trúc dữ liệu thích hợp và thuật toán tối ưu. Các đặc trưng

ảnh hưởng đến hiệu suất của thuật toán là:

Cách tổ chức biểu diễn dữ liệu để lưu trữ vào bộ nhớ.

Các hướng tiếp cận để tìm và liệt kê mẫu tuần tự.

Kỹ thuật tạo mẫu ứng viên.

Phương pháp duyệt không gian tìm kiếm.

Ngoài ra, sử dụng một số đặc trưng khác như lý thuyết đồ thị, đưa ra những ràng buộc cho bài toán sẽ giúp thuật toán thực thi nhanh hơn, các mẫu phổ biến tìm được có giá trị hơn.

Các cách tổ chức dữ liệu

Có hai dạng tổ chức dữ liệu cơ bản gồm dạng biểu diễn ngang và biểu diễn dọc.

Bảng 2. CSDL gốc ban đầu

Đối tượng	Chuỗi sự kiện
1	A, B, C
2	A, D, E, F
3	B, E

Bảng 3. CSDL biểu diễn ngang

Đối tượng	Sự kiện
1	A, B, C
2	A, D, E, F
3	B, E

Bảng 4. CSDL biểu diễn dọc

Sự kiện	Đối tượng
A	1, 2
B	1, 3
C	1
D	2
E	2, 3
F	2

Biểu diễn ngang là dữ liệu được tổ chức theo chiều ngang, mỗi hàng đại diện cho dãy sự kiện (event) xảy ra tương ứng với đối tượng (object). Biểu diễn ngang thể hiện chuỗi sự kiện xảy ra của một đối tượng. Biểu diễn dọc là dữ liệu được tổ chức theo chiều dọc, mỗi hàng đại diện cho dãy đối tượng tương ứng với một sự kiện. Biểu diễn dọc thể hiện dãy đối tượng tham gia vào một sự kiện.

Trong hai cách tổ chức, thao tác đếm độ phổ biến của một sự kiện ở CSDL biểu diễn dọc đơn giản và nhanh hơn. Bởi vì theo cách biểu diễn này, có thể lấy được ngay các đối tượng ứng với sự kiện mà không phải duyệt toàn bộ CSDL. Hơn nữa, đối với CSDL lớn, việc biểu diễn theo chiều dọc mang tính cô đọng, giúp thực thi nhanh hơn và cho phép lặp lại việc tìm các mẫu tuần tự một cách dễ dàng. Tuy nhiên, dữ liệu gốc ban đầu thường được biểu diễn ngang, nếu muốn biểu diễn lại theo chiều dọc phải có bước tiền xử lý để chuyển đổi.

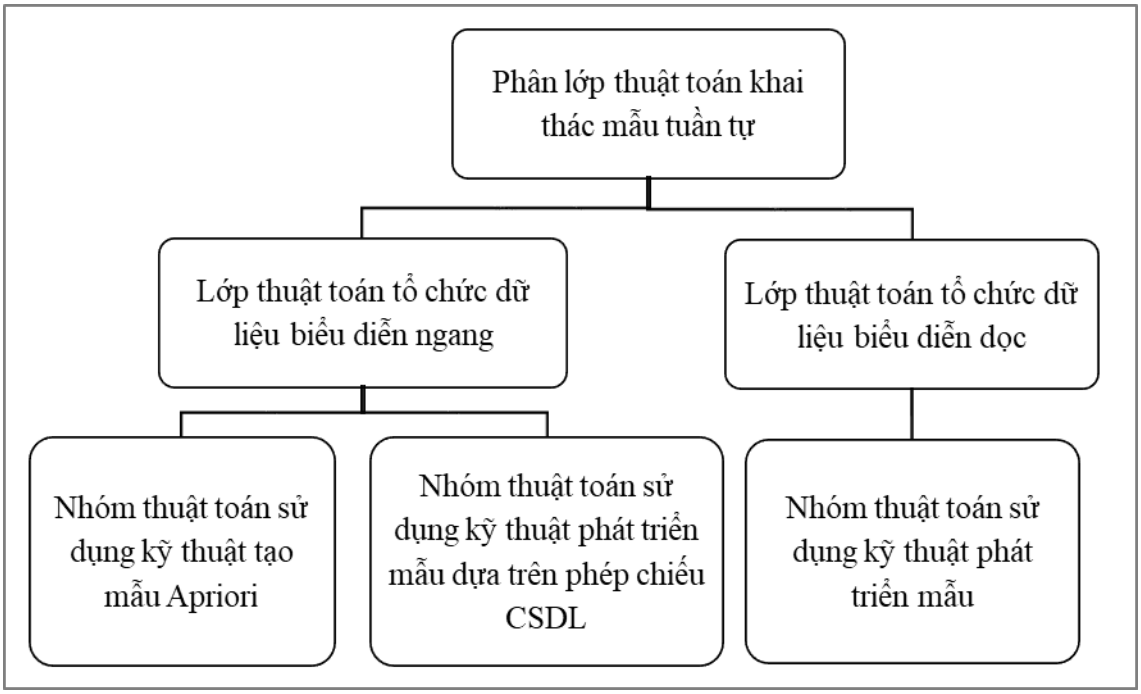
Các hướng tiếp cận để tìm và liệt kê mẫu tuần tự

Để tìm các mẫu tuần tự trong CSDL chuỗi, có hai bước cơ bản. Trước hết, tìm và liệt kê các chuỗi con (gọi là mẫu) có trong CSDL. Sau đó, xác định độ phổ biến của mỗi mẫu để kiểm tra có lớn hơn ngưỡng phổ biến tối thiểu hay không. Việc tìm và liệt kê các mẫu tuần tự có hai hướng tiếp cận: từ trên xuống và từ dưới lên.

Đối với phương pháp tiếp cận từ trên xuống, dựa vào tập chuỗi của CSDL, thuật toán bắt đầu từ việc tạo ra một tập các chuỗi có độ dài lớn hơn hoặc bằng chuỗi dài nhất trong CSDL. Nếu chuỗi tạo ra là phổ biến thì dừng quá trình tìm đối với mẫu đó vì hiển nhiên tất cả các chuỗi con của nó mà phổ biến đều được chứa trong nó (theo tính chất Apriori đề xuất bởi Agrawal và đồng sự, 1995). Do đó, cách tiếp cận này thường áp dụng cho các thuật toán khai thác chuỗi phổ biến tối đại. Ngược lại, nếu chuỗi đã tạo không phổ biến, mỗi lần ta loại bỏ bớt một hoặc một số sự kiện để tạo ra chuỗi mới nhỏ hơn cho đến khi các chuỗi con thu được là phổ biến. Quá trình này dừng khi tất cả các chuỗi con phát sinh đều phổ biến hoặc khi chúng đạt đến độ dài tối thiểu cho trước.

Đối với phương pháp tiếp cận từ dưới lên, thuật toán xuất phát từ mẫu rỗng. Sau đó, mở rộng phát triển mẫu dần theo các sự kiện có thể, mỗi lần mở rộng mẫu là thêm vào một sự kiện. Nếu độ phổ biến của mẫu mới mở rộng nhỏ hơn ngưỡng phổ biến tối thiểu thì loại bỏ mẫu này và dừng mở rộng nó; ngược lại mẫu tiếp tục được mở rộng. Trong quá trình này, các mẫu ở cùng một mức xử lý sẽ có cùng độ dài. Quá trình kết thúc khi không tạo ra mẫu phổ biến mới.

Như vậy, để tìm và liệt kê mẫu tuần tự phổ biến dạng tổng quát, ta nên sử dụng phương pháp tiếp cận từ dưới lên.



Hình 1. Phân loại các thuật toán khai thác mẫu tuần tự

Các kỹ thuật tạo mẫu ứng viên

Có hai kỹ thuật cơ bản, bao gồm: Kỹ thuật tạo mẫu Apriori và tạo mẫu theo cách phát triển mẫu.

Kỹ thuật tạo mẫu Apriori: gọi L_k là tập các mẫu phổ biến độ dài *k*. Các mẫu ứng viên độ dài (k+1) được tạo ra bằng cách kết L_k và L_k.

Một chuỗi s_1 kết được với s_2 nếu chuỗi con thu được khi xóa item đầu tiên trong s_1 giống với chuỗi con thu được khi xóa item cuối cùng trong s_2. Chuỗi ứng viên tạo ra bởi phép kết s_1 với s_2 là chuỗi s_1 được mở rộng bởi item của của s_2. Khi thêm vào s_1, item đóng vai trò là một itemset mới hoặc được thêm vào itemset cuối phụ thuộc vào vị trí của item đó trong s_2. Lưu ý, khi kết L_1 với L_1, cần thêm item theo cả hai cách.

Kỹ thuật phát triển mẫu: mẫu mới (chuỗi mới) được tạo ra bằng cách mở rộng một mẫu phổ biến độ dài *k* ($k > 0$) đã có với một item phổ biến. Có hai cách mở rộng: mở rộng itemset và mở rộng sequence. Lấy $\alpha = \langle a_1 a_2 \ldots a_n \rangle$ là một mẫu phổ biến và *e* là một item phổ biến. Mở rộng mẫu theo item *e*, ta được mẫu mới α' như sau:

Mở rộng itemset: $\alpha' = \langle a_1 a_2 \ldots a'_n \rangle$ trong đó *e* được thêm vào itemset cuối a_n của chuỗi α, tức $a'_n = a_n \cup e$.

Mở rộng sequence: $\alpha' = \langle a_1 a_2 \ldots a_n e \rangle$ trong đó *e* được thêm vào chuỗi α với vai trò là một itemset mới.

Các phương pháp duyệt không gian tìm kiếm

Để tìm và liệt kê các mẫu tuần tự theo hướng tiếp cận từ dưới lên, có thể duyệt theo hai cách: Duyệt theo chiều rộng (Breadth -First Search - BFS) hoặc duyệt theo chiều sâu (Depth First Search - DFS). Chúng ta có thể coi không gian tìm kiếm như một đồ thị, là cấu trúc cây/dàn.

Đối với phương pháp BFS, ở mỗi lần duyệt, các mẫu được tạo ra luôn có cùng độ dài vì mỗi lần mở rộng ta thêm vào một item cho tất cả các mẫu trước khi sang mức tiếp theo.

Đối với phương pháp DFS, điểm di chuyển trên không gian tìm kiếm của một mẫu được mở rộng qua các mức cho đến khi mẫu hiện hành không phổ biến. Sau đó, giải thuật quay lui và thực hiện đối với mẫu được mở rộng ở mức bắt đầu. Quá trình này lặp đi lặp lại cho đến khi tất cả các mẫu mở rộng đều được duyệt.

Như vậy, theo BFS ta có được tất cả các mẫu mở rộng ở cùng một mức trên không gian tìm kiếm trước khi di chuyển sang mức kế tiếp. Nhờ vậy, ta có được nhiều thông tin hơn trước khi mở rộng chuỗi ở mức tiếp theo, còn DFS thì không như vậy. Tuy nhiên, DFS thuận lợi hơn BFS ở chỗ nó cần ít bộ nhớ hơn. Bởi vì DFS chỉ cần lưu

vết của một chuỗi mở rộng ở mức trước và chuỗi ở mức đang duyệt, còn BFS phải lưu tất cả các chuỗi mở rộng ở mức trước và mức đang xét. Do đó, trong trường hợp số lượng chuỗi quá lớn thì BFS không thích hợp..

Phân loại các phương pháp khai thác

Khai thác mẫu tuần tự là bài toán khá thông dụng, thu hút nhiều nghiên cứu. Cho đến nay, nhiều thuật toán đã được đề xuất để giải quyết bài toán này. Dựa vào cách thức tổ chức dữ liệu, có thể chia các thuật toán thành hai lớp như sau:

(1) Lớp thuật toán tổ chức dữ liệu biểu diễn ngang.

(2) Lớp thuật toán tổ chức dữ liệu biểu diễn dọc.

Các phương pháp khai thác mẫu tuần tự dựa trên ràng buộc đều phát triển từ các phương pháp khai thác mẫu tuần tự nói chung. Do đó, các phương pháp khai thác có ràng buộc cũng được xếp tương ứng vào các nhóm của phương pháp khai thác mẫu tuần tự tổng quát.

Lớp thuật toán tổ chức dữ liệu biểu diễn ngang

Dựa vào kỹ thuật tạo mẫu, có thể chia các thuật toán trong lớp này thành hai nhóm: nhóm thuật toán dựa trên k□ thuật tạo mẫu Apriori và nhóm thuật toán dựa trên kỹ thuật phát triển mẫu dùng phép chiếu CSDL.

Nhóm thuật toán sử dụng kỹ thuật tạo mẫu Apriori:

Các thuật toán sử dụng CSDL biểu diễn ngang có sẵn, tạo mẫu bằng phương pháp Apriori có những đặc điểm sau:

Chỉ sử dụng CSDL cho trước, với mỗi tập ứng viên mới tạo ra, phải duyệt lại CSDL để đếm độ phổ biến. Do đó, thuật toán phải duyệt CSDL nhiều lần.

Tạo tập mẫu ứng viên dựa trên phương pháp Apriori truyền thống *"xuất phát từ những mẫu ngắn hơn và kết hợp chúng thành những mẫu dài hơn"*. Phương pháp này sẽ tạo ra những ứng viên có thể không có mặt trong CSDL, vì quá trình tạo ứng viên này không truy cập gì đến CSDL. Do đó, số lượng mẫu ứng viên tạo ra rất lớn.

Duyệt không gian tìm kiếm theo chiều rộng.

Các thuật toán khai thác mẫu tuần tự đại diện điển hình của nhóm này là bộ ba thuật toán *AprioriAll, AprioriSome, DynamicSome*. Đây cũng là bộ ba thuật toán đầu tiên (Agrawal và đồng sự, 1995) đặt nền móng cho các nghiên cứu về sau.

Một số thuật toán khai thác mẫu tuần tự có ràng buộc điển hình theo phương pháp này gồm có *GSP* (Srikant & Agrawal, 1996), *CFR-Apriori* (Y. L. Chen, Y. H. Hu (2006), *MSP-Miner* (Sandra & Furtado, 2007), *PMPC (X. Wu, X. Zhu, Y. He, & A. N, 2013)*. Trong đó, *GSP* về cơ bản cũng tương tự bộ thuật toán -Apriori, chỉ khác ở chỗ *GSP* kết hợp thêm các ràng buộc thời gian và ràng buộc phân cấp vào quá trình khai thác mẫu. *CFR-Apriori* sử dụng ràng buộc về tính cô đọng của mẫu và tính mới xảy ra theo thời gian. *MSP-Miner* cũng là một thuật toán dựa trên Apriori, xử lý ràng buộc biểu thức có quy tắc nhưng dùng để khai thác mẫu tuần tự ưu tiên xảy ra theo thứ tự trước. *PMPC* tương tự *AprioriAll*, áp dụng khai thác chuỗi gen sinh học sử dụng ràng buộc trên số lượng kí tự đại diện cho một phân đoạn trong chuỗi gen.

Nhóm thuật toán sử dụng kỹ thuật phát triển mẫu và chiếu CSDL:

Các thuật toán thuộc lớp này có đặc điểm:

Các thuật toán thuộc nhóm này sử dụng một đại diện biểu diễn lại CSDL cũng theo chiều ngang như CSDL gốc, nhưng thực hiện chiếu CSDL theo mẫu hoặc theo tiền tố. Nhờ vậy, "*CSDL đã chiếu"* này sẽ có kích thước nhỏ hơn.

Mẫu mới được tạo ra bằng cách phát triển mẫu phổ biến ở bước trước. Phát triển mẫu bằng cách thêm vào một item phổ biến. Các item phổ biến dùng để phát triển mẫu được tìm trong CSDL đã chiếu. Do đó, kỹ thuật này tạo ra mẫu phổ biến trực tiếp, không cần sinh mẫu ứng viên rồi mới kiểm tra độ phổ biến.

Để tạo mẫu mới, phải duyệt CSDL chiếu để tìm item phổ biến. Do đó, vẫn phải duyệt CSDL nhiều lần, tuy nhiên không phải duyệt trên CSDL ban đầu mà duyệt trên các CSDL đã chiếu thu nhỏ dần. Khi xét mẫu càng dài thì CSDL được chiếu càng thu nhỏ.

Duyệt không gian tìm kiếm theo chiều sâu.

Thuật toán cơ sở của nhóm này bao gồm *FreeSpan* (J.Han và đồng sự, 2000) và *PrefixSpan* (J.Han và đồng sự, 2004). Trong đó PrefixSpan là dạng cải tiến của *FreeSpan*. Cả hai đều sử dụng đại diện biểu diễn ngang của CSDL song thực

Bảng 5. Các thuật toán khai thác có ràng buộc tổ chức dữ liệu biểu diễn ngang, sử dụng kĩ thuật phát triển mẫu và chiếu CSDL

Năm	Thuật toán	Cơ sở lý thuyết
2005	DELISP	Sử dụng ràng buộc khoảng thời gian ngắt quãng.
2007	PG	Mô hình chung để khai thác mẫu tuần tự có ràng buộc dựa vào tính chất đơn điệu và phản đơn điệu của các loại ràng buộc.
2008	PTAC	Sử dụng ràng buộc kết hợp.
2009	GTC	Sử dụng ràng buộc khoảng thời gian ngắt quãng.
2010	SMAC	Sử dụng ràng buộc kết hợp (phát triển từ PTAC).
2014	CFML-PrefixSpan	Sử dụng ràng buộc về độ dại mẫu, tính cô đọng của mẫu và tính mới xảy ra theo thời gian.
2015	Clo-SPEC	Sử dụng ràng buộc khoảng thời gian ngắt quãng.
2015	LIT-PrefixSpan	Sử dụng ràng buộc khoảng thời gian xảy ra
2017	PPICt	Sử dụng ràng buộc khoảng thời gian ngắt quãng.
2018	HFGSO	Sử dụng ràng buộc độ dài và ràng buộc biểu diễn dưới dạng biểu thức có quy tắc cho khai thác chuỗi gen DNA.

hiện chiếu CSDL lớn thành những CSDL nhỏ hơn dựa theo các item phổ biến và phát triển mẫu từ những CSDL đã chiếu này. Một CSDL được chiếu theo mẫu/tiền tố α là tập hợp các chuỗi con, là những hậu tố của các chuỗi trong CSDL ban đầu có tiền tố là α. Tại mỗi bước, thuật toán tìm các mẫu phổ biến theo tiền tố α và CSDL chiếu theo tiền tố tương ứng. Mẫu phổ biến mới được tạo ra từ tiền tố α (mẫu phổ biến tìm được ở bước trước) mở rộng theo các item phổ biến tìm được trong CSDL chiếu. Như vậy, các thuật toán dựa trên kỹ thuật chiếu mẫu không tạo tập mẫu ứng viên đồ sộ, không phải duyệt lại CSDL gốc để đếm độ phổ biến song vẫn phải duyệt các CSDL con đã được chiếu theo tiền tố. Do đó, đã có nhiều thuật toán khai thác mẫu có ràng buộc được phát triển từ nhóm thuật toán này được trình bày ở Bảng 5.

Lớp thuật toán tổ chức dữ liệu biểu diễn dọc

Các thuật toán thuộc lớp này sử dụng một đại diện biểu diễn lại CSDL theo chiều dọc.

Với mỗi item *i* có mặt trong CSDL, đại diện biểu diễn dọc cho biết ngay các itemset nào trong mỗi chuỗi dữ liệu có chứa item *i* đó.

Tạo mẫu dựa trên phương pháp phát triển mẫu: mẫu ứng viên được tạo ra bằng cách mở rộng các mẫu phổ biến đã tìm được ở bước trước. Mở rộng mẫu là thêm vào một item phổ biến, các item này được lấy từ tập F_1. Tại mỗi mức phát triển mẫu, item sẽ bị loại bỏ nếu tạo ra mẫu ứng viên không phổ biến.

Độ phổ biến của ứng viên được xác định trực tiếp từ dại điện dọc của CSDL. Do đó, chỉ duyệt CSDL một lần (nếu bỏ qua bước tiền xử lý biến đổi CSDL từ ngang thành dọc).

Duyệt không gian tìm kiếm theo chiều sâu.

Các thuật toán tiêu biểu thuộc nhóm này là *SPADE* (M.J. Zaki, 2000), *SPAM* (J. Ayres, 2002), *PRISM* (Gouda và đồng sự, 2010). Với mỗi item, *SPADE* lưu danh sách ID của các chuỗi dữ liệu chứa item đó (SID) và ID của các itemset trong chuỗi có chứa item đó (TID), gọi là danh sách ID (ID-List). Cách tổ chức này giúp cho việc tạo ra danh sách các ID-List cho mỗi mẫu trở nên dễ dàng hơn, nhờ đó các mẫu mới được tạo ra theo phép kết đơn giản, đó là kết các ID-List với nhau. Độ phổ biến của mẫu được tính trực tiếp từ ID-List, do đó thuật toán không phải duyệt CSDL quá nhiều lần mà chỉ phải duyệt ba lần hoặc chỉ một lần đối với dữ liệu đã qua bước tiền xử lý. *SPADE* áp dụng được cả hai cách tìm kiếm chuỗi trên dàn là DFS và BFS.

Thay vì dùng ID-List như *SPADE*, *SPAM* dùng bitmap. Mỗi bit (*'1'* hoặc *'0'*) trong bitmap tương ứng với một itemset của chuỗi dữ liệu trong CSDL. *SPAM* sử dụng cấu trúc cây từ điển để thu nhỏ CSDL đại diện cho các chuỗi phổ biến. Mẫu ứng viên mới được phát triển từ mẫu phổ biến tìm được ở bước trước kết hợp mở rộng theo các item phổ biến. *SPAM* thực hiện nhanh hơn *SPADE* nhưng xử lý bộ nhớ không hiệu quả bằng *SPADE*, vì bitmap lưu những bit đại diện cho itemset của tất cả các chuỗi trong CSDL dù chuỗi không chứa mẫu.

Gần đây, hai thuật toán cải tiến của *SPAM* và *SPADE* là *CM-SPAM* và *CM-SPADE* (P. Fournier-

Bảng 6. Các thuật toán khai thác có ràng buộc tổ chức dữ liệu biểu diễn dọc

Năm	Thuật toán	Cơ sở lý thuyết
2000	cSPADE	Phát triển từ SPADE, kết hợp sử dụng nhiều ràng buộc: ràng buộc khoảng thời gian ngắt quảng, ràng buộc item và ràng buộc độ dài.
2003	GoSpec	Phát triển từ SPADE, sử dụng ràng buộc khoảng thời gian xảy ra và khoảng thời gian ngắt quãng.
2004	CCSM	Phát triển từ SPADE, sử dụng ràng buộc khoảng thời gian ngắt quãng.
2004	Querry	Phát triển từ SPAM, sử dụng ràng buộc item, phát biểu dưới dạng câu truy vấn.
2005	Pex-SPAM	Phát triển từ SPAM, sử dụng ràng buộc về khoảng cách item và ràng buộc biểu thức có quy tắc để khai thác chuỗi Protein.
2018	MSPIC-DBV	Kế thừa PRISM, áp dụng cấu trúc DBV, sử dụng ràng buộc itemset
2018	EMWAPC	Kế thừa PRISM, áp dụng cấu trúc DBV, sử dụng ràng buộc chuỗi con
2019	FARPAM	Phát triển từ SPAM, sử dụng ràng buộc về dấu thời gian.
2023	WSPM_PreTree	Kế thừa PRISM, áp dụng cấu trúc DBV, sử dụng ràng buộc trọng số của item trong mẫu

Viger và đồng sự, 2014) dùng thêm cấu trúc *CMAP (the Co-occurrence Map)* để lưu thông tin về sự xuất hiện cùng nhau của các item trong CSDL. Dựa vào *CMAP*, thuật toán cải tiến có thể thực hiện tỉa các ứng viên từ sớm để giảm không gian tìm kiếm. Nhờ đó, chúng vượt trội hơn các phương pháp kinh điển đã có (*GSP*, *PrefixSpan*, *SPADE* và *SPAM*).

Để khắc phục nhược điểm này của *SPAM*, *PRISM* sử dụng phương pháp mã hóa nguyên tố theo từng khối bit để nén bitmap của *SPAM*. Mỗi giá trị nguyên tố tương ứng một khối 8 bit (hoặc 16 bit). Để xác định độ phổ biến, mỗi mẫu có hai thông tin đi kèm gồm: dãy khối mã hóa chuỗi (cho biết những chuỗi nào trong CSDL có chứa mẫu) và dãy khối mã hóa vị trí (chỉ ra vị trí xuất hiện của mẫu trong mỗi chuỗi dữ liệu). Tuy nhiên, *PRISM* chỉ loại bỏ được những khối vị trí rỗng mà không bỏ đi được các khối chuỗi rỗng dù có rất nhiều chuỗi dữ liệu không chứa mẫu.

Một phương pháp mới có thể khắc phục hạn chế của *SPAM* lẫn *PRISM*, đó là dùng vector bit động (*Dynamic Bit Vector – DBV*). Một vector bit động *DBV* gồm: v*ector bit* là dãy các byte sau khi xóa các byte *'0'* ở đầu và cuối dãy và biến chỉ mục chỉ ra vị trí xuất hiện đầu tiên của byte khác *'0'* trong dãy vector bit. *DBV* sẽ áp dụng cho cả dãy khối mã hóa chuỗi và dãy khối mã hóa vị trí, nhờ vậy có thể loại bỏ các khối chuỗi rỗng mà *PRISM* không làm được. Hơn nữa, thao tác xử lý dãy vector bit cũng đơn giản và nhanh hơn so với dãy mã hóa nguyên tố.

DBV được đề xuất đầu tiên trong khai thác tập itemset đóng (V. Bảy và đồng sự, 2012) và đã đưa vào áp dụng trong khai thác chuỗi phổ biến với các ràng buộc itemset (V.Trang, 2018), ràng buộc chuỗi con (V.Trang và đồng sự, 2018) và ràng buộc trọng số (P.T.Thiết và đồng sự, 2023). Hướng tiếp cận này cải thiện đáng kể về thời gian khai thác và bộ nhớ sử dụng.

Đánh giá chung

Đối với lớp thuật toán có dữ liệu biểu diễn ngang, ngay từ đầu các nhà nghiên cứu đã nhận thấy rằng nhóm thuật toán dựa trên Apriori có nhược điểm là phải tạo ra tập ứng viên với số lượng bùng nổ lũy thừa trong trường hợp xấu nhất. Chẳng hạn, để tìm được một chuỗi phổ biến có độ dài 100 thì phải tạo ra $2^{100} \approx 10^{30}$ ứng viên. Mặc dù, càng về sau các thuật toán thuộc lớp thuật toán dựa trên Apriori đã cải thiện, song vẫn không khắc phục được trong trường hợp CSDL lớn. Mặt khác, chúng phải duyệt CSDL nhiều lần để đếm độ phổ biến cho tập mẫu ứng viên. Do đó, về sau hướng tiếp cận này không còn được kế thừa phát triển.

Nhóm thuật toán dựa trên phương pháp phát triển mẫu chiếu CSDL đã bỏ đi bước phát sinh và tỉa ứng viên có ở các thuật toán kiểu - Apriori bằng cách chia nhỏ thành tập các cơ sở dữ liệu chiếu để khai thác riêng rẽ. Tuy nhiên, điểm bất lợi của nhóm thuật toán này là tốn chi phí cho việc thực hiện phép chiếu CSDL và vẫn phải duyệt CSDL nhiều lần. Khi so sánh với các thuật toán kiểu-Apriori, nhìn chung các thuật toán phát triển mẫu cài đặt, kiểm thử phức tạp hơn, song lại cho hiệu quả cao hơn cả về thời gian và bộ nhớ sử dụng.

Các thuật toán tổ chức dữ liệu biểu diễn dọc thực hiện tốt hơn so với biểu diễn ngang về thời gian thực thi và bộ nhớ sử dụng vì nhờ sử dụng

đại diện biểu diễn lại CSDL theo chiều dọc, dữ liệu thường trú trên bộ nhớ nên không phải duyệt lại CSDL nhiều lần như các thuật toán ngang, cũng không đòi hỏi nhiều xử lý tính toán để xây dựng các CSDL chiếu như các thuật toán phát triển mẫu dựa trên phép chiếu CSDL. Hơn nữa, việc tạo ra mẫu mới cũng dễ dàng, độ phổ biến của mẫu được xác định trực tiếp mà không phải duyệt lại CSDL. Sự thay thế cách định dạng dữ liệu này dẫn đến sự ra đời của các thuật toán dựa trên phương pháp duyệt không gian tìm kiếm theo chiều sâu; đồng thời cũng dẫn đến sự tăng cường tập trung nghiên cứu việc kết hợp các ràng buộc vào quá trình khai thác, nhằm cải thiện thời gian xử lý và đáp ứng nhu cầu thu gọn số lượng mẫu tìm được cô đọng theo mối quan tâm của người dùng (P. Fournier-Viger và đồng sự, 2017).

Như vậy, trong số những phương pháp đã có, cách tổ chức dữ liệu theo định dạng dọc, tạo mẫu dựa trên phương pháp phát triển mẫu, duyệt không gian tìm kiếm theo chiều sâu hiệu quả hơn, đây là hướng tiếp cận triển vọng nhất cho các nghiên cứu về sau.

Kết luận

Bài báo đã trình bày toàn bộ cơ sở lý thuyết nền tảng liên quan đến lĩnh vực khai thác chuỗi sự kiện dựa trên các ràng buộc theo mối quan tâm của người dùng. Trong đó, trọng tâm là tìm hiểu, phân loại và phân tích ưu nhược điểm của các phương pháp khai thác mẫu đã có. Từ đó, giúp định hướng cho các nghiên cứu về sau chọn lựa hướng tiếp cận và phương pháp giải quyết bài toán phù hợp từng lĩnh vực ứng dụng.

TÀI LIỆU THAM KHẢO

J. Ayres, J. Flannick, J. Gehrke, & T. Yiu (2002). Sequential pattern mining using a bitmap representation, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 429-435.

J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, & M. C. Hsu (2000). Freespan: Frequent pattern projected sequential pattern mining, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 355–359.

J. Han, J. Pei, B. Mortazavi-Asl., Q. Chen, U. Dayal, & M.C. Hsu (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", *IEEE Transactions on Knowledge and Data Engineering,* vol. 16(11), pp. 1424-1440.

J. Pei, J. Han, WeiWang (2007). Constraint-based sequential pattern mining: the pattern-growth methods", *Journal of Intelligent Information Systems*, vol. 28(2), pp. 133-160.

K. Gouda, M. Hassaan, & M.J. Zaki, (2010). Prism: A Primal-Encoding Approach for Frequent Sequence Mining, *Journal of Computer and System Sciences*, vol. 76(1), pp. 88-102.

M. J. Zaki (2000). SPADE: An Efficient Algorithm for Mining Frequent Sequences, *Journal of Machine Learning*, vol. 42(1/2), pp. 31-60.

P. Fournier-Viger, A. Gomariz, M. Campos, & R. Thomas (2014). Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information, *In Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference*, pp. 40-52.

P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, & R. Thomas (2017). A survey of sequential pattern mining. *Journal of Data Science and Pattern Recognition*, *1*(1), 54-77.

Pham Thi Thiet, Vu Thuy Duong, N. Ta -Du, Huynh Bao, & V. Trang (2023). Mining Weighted Sequential Patterns Based on Prefix-Tree and Prism Encoding. *Vietnam Journal of Computer Science*, 1-16.

R. Agrawal, & R. Srikant (1995). Mining sequential patterns, *Proceedings of the 11th International Conference on Data Engineering*, pp. 3-14.

V. Bay, T.P. Hong, & L. Bac (2012). DBV-Miner: A Dynamic Bit-Vector approach for fast mining frequent closed itemsets, *Journal of Expert Systems with Applications*, vol. 39(8), pp. 7196-7206.

V. Trang, Atsuo Yoshitaka, & L. Bac (2018). Mining web access patterns with super-pattern constraint. *Journal of Applied Intelligence* vol. 48(11): 3902-3914.

V. Trang, V. Bay, & L. Bac (2018). Mining sequential patterns with itemset constraints. *Journal of Knowledge and Information System*, vol 57(2): 311-330.