MỘT THUẬT TOÁN HIỆU QUẢ CHO BÀI TOÁN KHAI THÁC MẪU TUẦN TỰ VỚI RÀNG BUỘC TRỌNG SỐ

PHẠM THỊ THIẾT
Khoa Công Nghệ Thông Tin, Trường Đại học Công nghiệp Thành phố Hồ Chí Minh
phamthithiet@iuh.edu.vn

Tóm tắt. Khai thác mẫu tuần tự có trọng số giúp tìm ra các mẫu có giá trị cao hơn nên có thể được áp dụng trong nhiều lĩnh vực hơn đồng thời giải quyết một số khó khăn về không gian lưu trữ và tài nguyên thực hiện trong bài toán khai thác mẫu tuần tự với độ hỗ trợ min_sup thấp. Bài báo đề xuất một tiếp cận mới trong khai thác mẫu tuần tự có trọng số bằng việc kết hợp giá trị trọng số thực của các item trong cơ sở dữ liệu chuỗi cùng với độ hỗ trợ của chúng để tìm ra tập mẫu phổ biến có giá trị hơn. Hơn nữa, thuật toán đề xuất sử dụng phương pháp tiếp cận dữ liệu theo chiều dọc nên thuật toán chỉ cần duyệt cơ sỡ dữ liệu một lần, do đó tiết kiệm được thời gian thực thi. Bên cạnh đó, để tăng hiệu suất tính toán, thuật toán áp dụng mã hóa khối nguyên tố trong các bước tính toán của quá trình phát triển mẫu. Kết quả thực nghiệm cho thấy thuật toán đề xuất có thời gian thực thi hiệu quả hơn.
Từ khóa. mẫu tuần tự, mẫu tuần tự có ràng buộc trọng số, CSDLchuỗi.

AN EFFICIENT ALGORITHM FOR MINING WEIGHTED SEQUENTIAL PATTERNS

Abstract. Mining weighted sequential patterns is to find higher-value patterns and these patterns can be applied in more fields, and at the same time it addresses some of the storage and resource limitations in the problem of mining sequential patterns with the low min_sup support. The paper proposes a new approach for mining weighted sequential patterns by combining the actual weight values of items in the sequence database with their support to find higher-value sequential patterns set. Moreover, the proposed algorithm uses a vertical database approach, so the algorithm only needs to scan the database once, thus saving execution time. In addition, to increase computational efficiency, the algorithm applies the prime block encoding approach in the computational steps of the extension pattern process. Experimental results show that the proposed algorithm has more effective execution time.
Keywords: sequential pattern, weighted sequential pattern, sequence database.

1 GIỚI THIỆU

Cho đến nay đã có rất nhiều công trình nghiên cứu về lĩnh vực khai thác dữ liệu nói chung, khai thác mẫu tuần tự phổ biến nói riêng. Việc khai thác mẫu tuần tự là một phần quan trọng của khai thác dữ liệu với các ứng dụng rộng rãi trong nhiều lĩnh vực kinh tế và khoa học như: phân tích quá trình mua bán hàng hóa, dự đoán thiên tai, phân tích chuỗi DNA, phân tích cấu trúc gen,… Bài toán khai thác mẫu tuần tự từ cơ sở dữ liệu (CSDL) chuỗi là để tìm ra tập các chuỗi con phổ biến thỏa mãn một ngưỡng hỗ trợ tối thiểu (min_sup) do người dùng đặt ra [1, 5, 10, 13, 16]. Đây là một trong những bài toán quan trọng trong lĩnh vực khai thác dữ liệu từ CSDL chuỗi và là nền tảng của nhiều nhiệm vụ khai thác dữ liệu khác như gom nhóm dữ liệu [3, 9], phân loại và dự đoán dữ liệu [9], phân loại dữ liệu dựa trên luật kết hợp [14]. Có rất nhiều thuật toán được đề xuất để cải thiện hiệu suất của quá trình khai thác mẫu tuần tự trên CSDL chuỗi như PSP [12], PrefixSpan [13], SPADE [22], SPAM [2], và PRISM [7], CM-SPADE [4], MCM-SPADE [11]. Tuy nhiên các thuật toán này chỉ sử dụng độ hỗ trợ để tìm ra các mẫu và khi khai thác mẫu tuần tự với độ hỗ trợ tối thiểu thấp sẽ phát sinh ra một lượng mẫu khổng lồ, điều này có thể làm cho không gian lưu trữ các mẫu bị quá tải. Để giải quyết vấn đề về không gian lưu trữ thì các phương pháp này cần phải tăng độ hỗ trợ tối thiểu min_sup [18, 21], tập các mẫu thu được giảm đi, tuy nhiên việc làm này có thể làm mất đi nhiều mẫu có tầm quan trọng cao nhưng lại có độ hỗ trợ chưa đủ lớn (tần suất xuất hiện trong các chuỗi trên toàn CSDL không nhiều). Hơn nữa, các thuật toán khai thác mẫu tuần tự trên đều thống nhất các mẫu tuần tự có tầm quan trọng là như nhau, trong khi đó, trong thực tế, mỗi thành phần trong CSDL có tầm quan trọng khác nhau. Những items nằm trong chuỗi có mức hỗ trợ thấp có thể có tầm quan

trọng hơn do tính năng trọng số của chính những items đó. Do vậy, nếu một mẫu có tầm quan trọng cao (hay trọng số cao) thì mẫu đó có giá trị cao và nên được lưu trữ lại để sử dụng. Chính vì thế, nếu một mẫu có độ hỗ trợ không thỏa điều kiện về ngưỡng hỗ trợ tối thiểu thì mẫu đó vẫn có thể được giữ lại sau quá trình khai thác mẫu nếu như trọng số của mẫu đó đủ lớn. Vấn đề tồn tại cần giải quyết là làm cách nào để không làm mất đi các mẫu có tầm quan trọng nhưng vẫn giữ được độ hỗ trợ tối thiểu ở mức hợp lý cho việc thu thập và lưu trữ tập các mẫu sau quá trình khai thác dữ liệu để không tạo ra tập mẫu quá lớn, gây dư thừa. Bài toán khai thác mẫu tuần tự dựa trên các ràng buộc trọng số được đề xuất để tìm ra những mẫu tuần tự không phổ biến (không thỏa ràng buộc về mức hỗ trợ tối thiểu) nhưng lại có các items xuất hiện trong mẫu có tầm quan trọng cao trong cơ sở dữ liệu, từ đó có thể tìm ra được một tập các mẫu tuần tự có trọng số hoàn chỉnh, có tầm quan trọng hơn, có lợi ích nhiều trong việc sử dụng các mẫu. Nhờ vậy quá trình khai thác được cải thiện hiệu quả và tập mẫu tìm thấy đáp ứng yêu cầu của người dùng tốt hơn.

Việc khai thác mẫu tuần tự có trọng số sẽ thu thập được tập mẫu tuần tự hoàn chỉnh hơn với những items có mức hỗ trợ thấp nhưng có tầm quan trọng cao, cần thiết cho quá trình sử dụng dữ liệu. Do đó, bài báo tập trung nghiên cứu và đề xuất một thuật toán cho bài toán khai thác mẫu tuần tự với ràng buộc trọng số bằng cách kết hợp cả ràng buộc về mức hỗ trợ tối thiểu lẫn trọng số của các item trong CSDL chuỗi để khai thác ra tập mẫu hoàn chỉnh với những tính năng chặt chẽ hơn của các mẫu tuần tự sau quá trình khai thác dữ liệu. Bên cạnh đó, để thuật toán được hiệu quả hơn, thuật toán còn sử dụng phương pháp mã hóa khối nguyên tố [7] trong các bước tính toán của quá trình phát triển mẫu. Kết quả thực nghiệm của thuật toán đề xuất được so sánh với kết quả thực nghiệm của thuật toán SPMW [15].

Bố cục tiếp theo của bài báo được tổ chức như sau: Mục 2 trình bày các công trình liên quan đến bài toán khai thác mẫu tuần tự và khai thác mẫu tuần tự với ràng buộc trọng số. Mục 3 trình bày các định nghĩa liên quan đến bài toán khai thác mẫu tuần tự với ràng buộc trọng số. Thuật toán đề xuất để khai thác mẫu tuần tự với ràng buộc trọng số được thảo luận trong Mục 4. Kết quả thực nghiệm được mô tả trong Mục 5 và Mục 6 là kết luận và hướng phát triển.

2 CÁC CÔNG TRÌNH LIÊN QUAN

AprioriAll [1] là thuật toán đầu tiên được thiết kế để giải quyết bài toán khai thác mẫu tuần tự trên CSDL chuỗi giao dịch. Để tìm mẫu tuần tự, thuật toán AprioriAll gồm 3 giai đoạn chính: Tìm itemset phổ biến, biến đổi CSDL và tìm mẫu tuần tự trên dữ liệu đã biến đổi. Để tìm được mẫu tuần tự, thuật toán AprioriAll phải phát sinh các ứng viên, nhưng số lượng ứng viên tạo ra rất lớn dễ dẫn đến tình trạng "nghẽn cổ chai". Mặt khác, để tìm tất cả các mẫu tuần tự, thuật toán AprioriAll phải duyệt CSDL nhiều lần để đếm độ hỗ trợ cho mỗi chuỗi ứng viên.

GSP [16] là một thuật toán được đề xuất vào năm 1996 bởi Agrawal và Srikant. Thuật toán này được mở rộng từ thuật toán AprioriAll [1]. GSP giải quyết bài toán khai thác mẫu tuần tự một cách tổng quát bằng cách bổ sung thêm các ràng buộc như: khoảng thời gian cực đại và cực tiểu giữa các thành phần trong một mẫu tuần tự. GSP sử dụng cấu trúc dữ liệu "cây băm" để giảm số ứng viên cần kiểm tra cho một chuỗi dữ liệu và biến đổi cách biểu diễn của chuỗi dữ liệu, do đó GSP khắc phục được tình trạng "nghẽn cổ chai" của AprioriAll và thực hiện nhanh hơn AprioriAll.

PSP [12] là một thuật toán khác được đề xuất bởi Masseglia và đồng sự vào năm 1998, nó cũng dựa trên mô hình Apriori giống như GSP. Điểm khác biệt của PSP so với GSP đó là PSP quản lý và lưu trữ các mẫu ứng viên bằng một cấu trúc dữ liệu cây tiền tố. PSP không thực hiện phép kết sinh Apriori. Cả ba phương pháp AprioriAll, GSP, PSP khi thực hiện đều phải duyệt CSDL nhiều lần và phải tải toàn bộ CSDL vào bộ nhớ chính, do đó những phương pháp này chỉ thật sự hiệu quả khi bộ nhớ chính có thể chứa hết toàn bộ CSDL. Tuy nhiên thuật toán PSP thực hiện hiệu quả hơn GSP.

PrefixSpan [13] là thuật toán được phát triển từ thuật toán FreeSpan [8]. FreeSpan là thuật toán đầu tiên thực hiện phép chiếu trên CSDL để giảm chi phí lưu trữ dữ liệu theo hướng tiếp cận theo chia nhỏ dữ liệu. Xuất phát từ tập mẫu tuần tự độ dài 1, PrefixSpan tạo ra CSDL được chiếu với mỗi mẫu đó. Trong CSDL chiếu, mỗi chuỗi dữ liệu chỉ giữ lại phần hậu tố đối với tiền tố đã chiếu. Mẫu được phát triển bằng những item phổ biến tìm được trong CSDL được chiếu. Quá trình này được lặp lại cho đến khi CSDL chiếu không còn item phổ biến nào. Tuy nhiên, khi phát triển mẫu, thuật toán FreeSpan và PrefixSpan đều phải thực hiện chiếu CSDL và duyệt CSDL chiếu để tìm item phổ biến.

Tất cả các phương pháp trên đều tiếp cận theo hướng biểu diễn thông tin dữ liệu theo chiều ngang. Để xác định độ hỗ trợ của một mẫu cần phải duyệt lại toàn bộ CSDL. Để khắc phục điều này, một số thuật toán như SPADE [22], SPAM [2], và PRISM [7] đã tiếp cận theo hướng biểu diễn thông tin dữ liệu theo

chiều dọc, đây là những thuật toán khá hiệu quả. Thay vì phải duyệt toàn bộ CSDL chuỗi, với mỗi mẫu ứng viên, chúng thực hiện lưu trữ thông tin cho biết mẫu đó có mặt trong những chuỗi dữ liệu nào, từ đó tính nhanh độ hỗ trợ. Mặt khác, mẫu mới được tạo ra lấy thông tin dựa trên những mẫu đã có và không cần phải duyệt lại CSDL.

Thuật toán SPADE [22] do Zaki đề xuất vào năm 2001. Thuật toán SPADE tổ chức dữ liệu theo chiều dọc, trong đó ứng với mỗi item sẽ lưu danh sách định danh của các chuỗi dữ liệu và định danh của các itemset có chứa item đó. Độ hỗ trợ của item được tính trực tiếp từ danh sách các định danh. Mặt khác, SPADE còn dựa trên lý thuyết dàn để chia nhỏ không gian tìm kiếm và thao tác kết đơn giản để tạo ra tập ứng viên. Thuật toán này gom nhóm các mẫu tuần tự dựa theo tiền tố thành các lớp tương đương. Với ngưỡng min_sup thấp, so với thuật toán GSP, thuật toán SPADE thực hiện nhanh gấp đôi [22].

Thuật toán SPAM [2] cũng tổ chức dữ liệu theo chiều dọc như thuật toán SPADE. Thông tin của các mẫu ứng viên được biểu diễn dưới dạng bảng bit dọc, mỗi bit ứng với một itemset của một chuỗi trong CSDL. Nếu item có mặt trong itemset j thì bit tương ứng itemset j được đánh dấu là 1, ngược lại là 0. Độ hỗ trợ của mẫu được xác định dựa trên bảng bit. Về tốc độ thực thi, trên các CSDL nhỏ, SPAM thực hiện nhanh hơn 2.5 lần so với SPADE, nhưng chưa thực hiện tốt bằng PrefixSpan. Với CSDL lớn, SPAM thực hiện tốt hơn nhiều so với SPADE và PrefixSpan vì tổ chức biểu diễn và lưu trữ dữ liệu dưới dạng bit nên thao tác phát sinh ứng viên và đếm độ hỗ trợ rất hiệu quả [2].

Thuật toán PRISM [7] được đề xuất bởi Gouda và các cộng sự vào năm 2010. Thuật toán PRISM tiếp cận theo hướng hoàn toàn khác biệt so với các thuật toán trước đây đó là sử dụng phương pháp mã hóa khối nguyên tố để biểu diễn thông tin của mẫu ứng viên. Thuật toán sử dụng cách tiếp cận dọc để liệt kê và đếm độ hỗ trợ, bên cạnh đó thuật toán sử dụng cấu trúc dữ liệu cây từ điển để lưu trữ các mẫu tuần tự tìm được. Thuật toán chỉ duyệt CSDL đúng một lần để tìm tập mẫu tuần tự độ dài 1 cùng với khối mã hóa thông tin tương ứng cho các mẫu đó. Sau đó, các mẫu được phát triển bằng cách thêm vào mẫu một item phổ biến. Thông tin của mẫu mới được xác đinh dựa trên khối mã hóa của mẫu cũ và của item thêm vào. Như vậy, thuật toán PRISM không phải duyệt CSDL nhiều lần. Đồng thời, thuật toán giảm chi phí tính toán bằng cách sử dụng bảng tra cho khối mã hóa thông tin dựa trên lý thuyết mã hóa nguyên tố.

Năm 2014, Fournier-Viger và cộng sự đã đề xuất thuật toán CM-SPADE [4] để khắc phục những điểm yếu của thuật toán SPADE bằng cách sử dụng cấu trúc dữ liệu gọi là CMAP CMAP (Co-occurrence MAP). Dữ liệu được lưu trữ trong CMAP là ánh xạ của từng item trong tập các item sao cho các item này có thể được mở rộng với 1 item khác để tạo thành một mẫu tuần tự mới. CM-SPADE cải thiện tốc độ xử lý và giảm mức sử dụng bộ nhớ bằng cách áp dụng thêm bước kiểm tra dựa trên cấu trúc dữ liệu của CMAP, điều này cho phép loại bỏ nhanh chóng các ứng viên không mong muốn và tránh chi phí cho việc xử lý các mẫu tuần tự dư thừa.

Thuật toán MCM-SPADE [11] do Huynh và các đồng sự đề xuất vào năm 2018 để khai thác mẫu tuần tự bằng cách sử dụng đa tiến trình. Để cải tiến hiệu suất của thuật tóan CM-SPADE, thuật toán MCM-SPADE đã đề xuất sử dụng định dạng dữ liệu theo chiều dọc kết hợp với cấu trúc [4] để sớm loại bỏ các ứng viên. Thuật toán MCM-SPADE đã phân chia các tác vụ liên quan đến từng nhân xử lý bằng cách sử dụng tính chất chia và trị dựa trên hệ thống kiến trúc đa nhân với kỹ thuật đa tiến trình.

Tuy nhiên, các thuật toán trên chỉ quan tâm và sử dụng ràng buộc về độ hỗ trợ của các item để tìm ra các mẫu tuần tự mà chưa xem xét đến ràng buộc trọng số của các item trong CSDL, việc làm này sẽ phát sinh ra tập mẫu rất lớn nếu như độ hỗ trợ tối thiểu nhỏ, và có thể làm cho không gian lưu trữ các mẫu bị quá tải. Để giải quyết vấn đề về không gian lưu trữ thì các phương pháp này phải tăng độ hỗ trợ tối thiểu, tuy nhiên với cách giải quyết này có thể làm mất đi nhiều mẫu có tầm quan trọng cao nhưng lại có độ hỗ trợ chưa đủ lớn (tần suất xuất hiện trong các chuỗi trên toàn CSDL không nhiều). Trong khi đó, trọng số của một item, một mẫu là đặc trưng cho tầm quan trọng (hay giá trị) của item, mẫu đó Chính vì thế, trong quá trình khai thác nên giữ lại các mẫu có trọng số cao mặc dù các mẫu này có độ hỗ trợ không thỏa ngưỡng hỗ trợ tối thiểu.

Srikant và Agrawal [16] là những người đầu tiên khái quát bài toán khai thác mẫu tuần tự mà cho phép xử lý các ràng buộc thời gian. Họ đã đưa ra hai loại ràng buộc về thời gian là ràng buộc khoảng thời gian (time-gap) và ràng buộc thời gian - cửa sổ trượt (sliding time window). Trong đó, ràng buộc khoảng thời gian giới hạn thời gian xảy ra giữa hai thành phần liền kề phải nằm trong một khoảng hợp lý, còn ràng buộc thời gian - cửa sổ trượt qui định những thành phần có thời gian nằm trong phạm vi cửa sổ sẽ nằm

trong cùng một giao dịch. Ngoài ra, các tác giả trong [16] còn đưa ra ràng buộc phân cấp trên các mẫu do người dùng định nghĩa.

Garofalakis và đồng sự [6] đã đề xuất các ràng buộc trên mẫu tuần tự dưới dạng các biểu thức có qui tắc và đưa ra một bộ bốn thuật toán, gọi là bộ SPIRIT. Trong đó, mỗi thuật toán giải quyết bài toán khai thác mẫu tuần tự với một loại biểu thức ràng buộc cho trước.

Yun và cộng sự đã đề xuất thuật toán WFIM [19] vào năm 2005 để khai thác mẫu tuần tự có trọng số trong CSDL giao dịch lớn bằng cách sử dụng một dải trọng số (weight range) và trọng số tối thiểu (min_weight) cùng với độ hỗ trợ tối thiểu. Trong thuật toán này, các item riêng lẻ được gán các trọng số khác nhau trong phạm vi trọng số để phản ánh tầm quan trọng của chúng và các ràng buộc trọng số sau đó sẽ được đưa vào trong thuật toán tăng trưởng mẫu để giữ thuộc tính giảm bao đóng. WFIM sử dụng một cây tiền tố tiếp cận theo hướng từ dưới lên (bottom – up) được sắp xếp theo thứ tự tăng dần. Thuật toán cho phép điều chỉnh số lượng các itemset phổ biến có trọng số bằng cách thay đổi các thông số như một dải trọng số và trọng số tối thiểu mặc dù ngưỡng hỗ trợ tối thiểu thấp hơn trong CSDL dày hoặc CSDL dài. Trọng số và độ hỗ trợ của mỗi item trong WFIM được xem xét riêng để cắt tỉa không gian tìm kiếm. WFIM cho phép người dùng cân bằng độ hỗ trợ và trọng số của các itemset.

Năm 2006, Yun và cộng sự [20] đề xuất thuật toán WSpan để khai thác mẫu tuần tự hiệu quả bằng cách đưa các ràng buộc trọng số vào trong thuật toán tăng trưởng mẫu tuần tự trong khi vẫn bảo toàn thuộc tính giảm bao đóng bằng cách tiếp cận theo hướng CSDL chiếu. Tác giả định nghĩa bài toán khai thác mẫu tuần tự có trọng số là bài toán tìm tập hoàn chỉnh các mẫu tuần tự có trọng số trong CSDL với ràng buộc về độ hỗ trợ và ràng buộc về trọng số. Tác giả xem xét việc áp dụng ràng buộc trọng số vào khai thác mẫu tuần tự là thuộc tính giảm bao đóng có thể bị phá vỡ bằng việc áp dụng ràng buộc trọng số đơn giản: "Một chuỗi với trọng số thấp hơn có thể là một mẫu tuần tự phổ biến bằng việc kết hợp các items có trọng số cao hơn trong chuỗi". Trong thuật toán này, một dải trọng số được sử dụng và các items được gán các giá trị trọng số khác nhau bên trong dải trọng số, bên cạnh đó thuật toán sử dụng một giá trị ngưỡng hỗ trợ tối thiểu *min_sup* và ngưỡng trọng số lớn nhất (MaxW) để làm điều kiện kiểm tra độ phổ biến của các mẫu trong quá trình khai thác CSDL chuỗi. WSpan có thể điều chỉnh số lượng mẫu tuần tự bằng việc điều chỉnh dải trọng số của các thành phần trong CSDL chuỗi đầu vào.

Năm 2016, Sirisha và các cộng sự [15] đã đề xuất một hướng tiếp cận mới để tìm ra mẫu tuần tự với giá trị trung bình trọng số (meanW) của mẫu tuần tự. Trong tiếp cận này, trước tiên các giá trị trọng số được gán vào các items, sau đó định nghĩa giá trị trung bình trọng số cho các mẫu tuần tự và dựa vào điều kiện $support * meanW < min\_sup$ để cắt tỉa mẫu. Với thuật toán này, một CSDL tuần tự được chiếu đệ quy vào một tập các CSDL chiếu có trọng số với kích thước nhỏ hơn và các mẫu tuần tự có trọng số được phát sinh trong mỗi CSDL chiếu có trọng số. Tuy nhiên, thuật toán này tiếp cận theo hướng tăng trưởng mẫu và chia nhỏ CSDL ra thành các CSDL chiếu các tiền tố làm cho thuật toán bị giới hạn về không gian lưu trữ và quá trình phát sinh CSDL chiếu các tiền tố mất chi phí cao, một số trường hợp không thể thực hiện được nếu CSDL chuỗi khai thác là các bộ CSDL dày và lớn.

Năm 20018, Van và các công sự [17] đã đề xuất một thuật toán gọi là MSPIC-DBV để khai thác các mẫu tuần tự dựa trên các ràng buộc itemset. Thuật toán này đã cải thiện đáng kể hiệu suất của bài toán khai thác mẫu tuần tự bằng cách sử dụng cấu trúc dữ liệu vector bit động kết hợp với cấu trúc cây tiền tố để biểu diễn chuỗi ứng viên. Tuy nhiên, thuật toán yêu cầu người sử dụng phải xác định trước tập các ràng buộc của các itemset mà không sử dụng các ràng buộc về trọng số thực tế của item trong CSDL.

Để cải thiện các vấn đề trên, bài báo này đề xuất một thuật toán WPM (**W**eighted **P**attern **M**ining) để khai thác mẫu tuần tự có trọng số bằng cách sử dụng phương pháp mã hóa khối nguyên tố và tính toán trọng số các item dựa trên trọng số thực tế của các thành phần trong CSDL.

3 CÁC ĐỊNH NGHĨA

CSDL chuỗi: Đặt $I = \{i_1, i_2, ..., i_m\}$ là một tập các mục (item). Một itemset là một tập các item con khác rỗng, Một itemset i được ký hiệu là $(i_1, i_2, ..., i_k)$, trong đó i_j là một item. Giả sử rằng các item trong một itemset được sắp xếp theo thứ tự từ điển, $S = \{s_1, s_2, ..., s_n\}$ là một tập các chuỗi, trong đó mỗi chuỗi s_x là một danh sách các item đã được sắp xếp theo thứ tự và $s_x = \{x_1, x_2, ..., x_p\}$ là 1 chuỗi trong đó x_i là một itemset và p là số lượng itemset sao cho $x_1, x_2, ..., x_p \subseteq I$. Trong s_x, x_1 xảy ra trước x_2, x_2 xảy ra trước x_3, v.v. Kích thước của một chuỗi là số lượng itemset có trong chuỗi đó. Số lượng các item có trong chuỗi

được gọi là độ dài của chuỗi, được xác định bởi $l = \sum_{j=1}^{n} s_j$. Một chuỗi có chiều dài l còn được gọi là *l-sequence*. Một CSDL chuỗi (*SD*) bao gồm một tập các chuỗi.

Độ hỗ trợ: độ hỗ trợ của chuỗi α được ký hiệu là *support*(α) trong *SD* là số các bộ trong *SD* có chứa α.

Mẫu tuần tự (chuỗi phổ biến): Cho trước một độ hỗ trợ tối thiểu *min_sup*, một chuỗi α trong *SD* là phổ biến nếu *support*(α) ≥ *min_sup*. Tức là, để chuỗi α là chuỗi phổ biến thì nó phải xuất hiện ít nhất *min_sup* lần trong *SD*. Một chuỗi phổ biến được gọi là một mẫu tuần tự. Mẫu tuần tự có chiều dài l được gọi là *l-pattern*.

Trọng số của item, itemset, sequence [15]:

Trọng số (*weight)* của một item $w(i)$ là một số thực không âm, phản ánh mức độ quan trọng của item trong *SD*.

Đặt i là một item đơn (single item), $s_1, s_2, ..., s_n$ là n chuỗi trong *SD*, trọng số của i được tính như sau:

$$W(i) = \frac{T(i)}{\sum_{1 \le j \le n} L(s_j)} \quad (1)$$

Với $T(i)$ là số lần xuất hiện của i trong *SD*, $L(s_j)$ là chiều dài của chuỗi s_j.

Đặt $\alpha = <t_1, t_2, ..., t_m>$ là một chuỗi, t_k ($1 \le k \le m$) là một thành phần của α, bao gồm n item đơn $i_1, i_2, ..., i_n$, trọng số của t_k được định nghĩa:

$$W(t_k) = \frac{\sum_{1 \le j \le n} W(i_j)}{n} \quad (2)$$

Và trọng số của α là:

$$W(\alpha) = \frac{\sum_{1 \le k \le m} W(t_k)}{m} \quad (3)$$

Trọng số trung bình:

Đặt *SD* là một CSDL chuỗi gồm n item đơn i_k ($1 \le k \le n$), ta định nghĩa trọng số lớn nhất *maxW* của *SD* là: $maxW = max_{1 \le k \le n}(W(i_k))$ và trọng số nhỏ nhất *minW* của *SD* là: $minW = min_{1 \le k \le n}(W(i_k))$. Khi đó, giá trị trọng số trung bình *MeanW* của *SD* được tính như sau:

$$meanW = (maxW + minW) / 2 \quad (4)$$

Item i là một item có trọng số phổ biến nếu *support(i)* * *meanW* ≥ *min_sup*

4 THUẬT TOÁN WPM ĐỂ KHAI THÁC MẪU TUẦN TỰ VỚI RÀNG BUỘC TRỌNG SỐ

Thuật toán khai thác mẫu tuần tự phổ biến với ràng buộc trọng số gọi là WPM (**W**eighted **P**attern **M**ining) do bài báo đề xuất được trình bày trong hình 1. Thuật toán WPM được xây dựng dựa trên sự kết hợp giữa giá trị độ hỗ trợ và trọng số của các item trong CSDL chuỗi để tìm ra tập các mẫu tuần tự có trọng số hoàn chỉnh và có giá trị cao. Các trọng số của các item trong thuật toán được tính toán dựa trên giá trị thực của item đó trong CSDL thay vì sử dụng một giá trị ước lượng do người dùng đặt. Bên cạnh đó, việc khai thác tập mẫu trên CSDL chuỗi trong thuật toán được thực hiện theo hướng tiếp cận cấu trúc dữ liệu được tổ chức theo chiều dọc và kết hợp sử dụng khối mã hóa nguyên tố [7] để biểu diễn thông tin ứng viên và tính toán độ hỗ trợ của các ứng viên khi phát triển các mẫu giúp nâng cao hiệu suất thực thi của phương pháp đề xuất.

Không gian tìm kiếm của thuật toán WPM là cây từ điển [2, 7], nghĩa là phát triển mẫu tuần tự phổ biến có trọng số theo DFS (Depth First Search). Nút gốc ở mức 0 của cây chứa chuỗi rỗng, các nút ở mức tiếp theo của cây gọi là mức thứ l và có kích thước là l. Nút con ở mức (l+1) được xây dựng bằng việc mở rộng chuỗi theo mức l và thu được chuỗi có kích thước (l+1), mở rộng mẫu được thực hiện bằng cách thêm một item phổ biến vào mẫu với 2 phương pháp là mở rộng mẫu theo chuỗi và mở rộng mẫu theo itemset. Thuật toán hoạt động như sau: Đầu vào của thuật toán gồm có một CSDL chuỗi biểu diễn theo chiều dọc, một ngưỡng hỗ trợ tối thiểu *min_sup*. Trước tiên, thuật toán duyệt CSDL đầu vào một lần và xác định trọng số của từng item trong CSDL. Từ đó xác định được trọng số lớn nhất và trọng số nhỏ nhất để tính được trọng số trung bình *meanW* từ hai giá trị này theo công thức (4) đã trình bày ở phần 3. Tiếp theo, thuật toán sẽ tìm ra tập các item phổ biến có chiều dài là 1 thỏa điều kiện: *support* (i) * *meanW* ≥ *min_sup*, đồng thời loại bỏ các item không phổ biến ra khỏi F_1. Với mỗi item phổ biến $i \in F_1$, thuật toán thực hiện phát triển mẫu theo chuỗi bằng thủ tục S_EXTEND và phát triển mẫu theo itemset bằng thủ tục I_EXTEND. Với một mẫu mới được phát sinh, thuật toán tiếp tục gọi đệ quy các thủ tục mở rộng mẫu theo chuỗi hay theo itemset để xây dựng không gian các mẫu tuần tự có trọng số theo chiều sâu. Các tham số đầu vào ứng với từng thủ tục để phát triển mẫu bao gồm tập S_n chứa các item được nối vào <*pat*> bằng

cách mở rộng theo chuỗi *s – extension* hoặc mở rộng theo itemset *i – extension*, *min_sup*. Với từng ứng viên *pat* được tạo bởi một lần mở rộng mẫu, thuật toán tính lại độ hỗ trợ của mẫu để kiểm tra xem mẫu đó có phổ biến hay không theo điều kiện *support* (α) * *meanW* ≥ *min_sup*. Việc này được thực hiện bằng toán tử kết và tính toán số các chuỗi mà các mẫu có xuất hiện dựa trên phương pháp khối mã hóa nguyên tố [7]. Nếu một mẫu là mẫu tuần tự phổ biến có trọng số, mẫu đó sẽ tiếp tục được sử dụng để gọi đệ quy hai thủ tục mở rộng mẫu theo chuỗi và theo itemset để tạo ra các mẫu ứng viên bắt đầu với tiền tố <*pat*>. Thuật toán cắt tỉa không gian tìm kiếm bằng việc không mở rộng các mẫu không phổ biến. Điều này thực hiện được dựa vào tính chất phổ biến của một mẫu tuần tự đó là mẫu tuần tự không phổ biến không thể được mở rộng để tạo thành một mẫu phổ biến.

WPM (*CSDL*, *min_sup*)

1. Duyệt *CSDL* để xác định meanW và F_1: danh sách các item phổ biến thỏa điều kiện *support (i)* * *meanW* ≥ *min_sup*
2. *pat_weight* :=∅
3. **FOREACH** item $i \in F_1$,
4. **S_EXTEND** (<*i*>, F_1, *min_sup*)
5. **I_EXTEND** (<*i*>, $\{e \in F_1 \mid e \succ_{lex} i\}$, *min_sup*)

S_EXTEND (*pat*, S_n, *min_sup*)

1. *pat_weight* ← *pat*;
2. S_{temp} :=∅
3. **FOREACH** item $j \in S_n$,
4. *pnew* = *s_extension*(*pat*, j);
5. **IF** *support*(*pnew*)**meanW* ≥ *min_sup*
6. **THEN** $S_{temp} := S_{temp} \cup (j)$
7. **FOREACH** *item* $j \in S_{temp}$,
8. **S_EXTEND** (*pnew*, S_{temp}, *min_sup*)

I_EXTEND (*pat*, I_n, *min_sup*)

1. *pat_weight* ← *pat*;
2. I_{temp} :=∅
3. **FOREACH** *item* $j \in I_n$,
4. *pnew* = *i_extension*(*pat*, *j*);
5. **IF** *support*(*pnew*)**meanW* ≥ *min_sup*
6. **THEN** $I_{temp} := I_{temp} \cup (j)$
7. **FOREACH** *item* $j \in I_{temp}$,
8. **I_EXTEND** (*pnew*, $\{e \in I_{temp} \mid e \succ_{lex} j\}$, *min_sup*)

Hình 1: Thuật toán WPM để khai thác mẫu tuần tự với ràng buộc trọng số

5 KẾT QUẢ THỰC NGHIỆM

Kết quả thực nghiệm của thuật toán WPM để khai thác mẫu tuần tự với ràng buộc trọng số mà bài báo đề xuất được so sánh với thuật toán SPMW [15]. Các kết quả thực nghiệm được thực hiện trên máy tính Intel (R), Core (TM) i3-2370M CPU 2.40 GHz, 4Gb RAM trên hệ điều hành Windows 10 với ngôn ngữ lập trình Java trên nền tảng Eclipse. CSDLsử dụng trong thực nghiệm là các bộ dữ liệu chuẩn được tải trực tiếp từ http://fimi.ua.ac.be/data/. Đây là địa chỉ chứa các tập dữ liệu tin cậy được cộng đồng nghiên cứu khai thác mẫu tuần tự sử dụng để kiểm chứng thực nghiệm các thuật toán đề xuất. Đặc điểm của các bộ dữ liệu sử dụng trong thực nghiệm được trình bày trong bảng 1.

Bảng 1: Đặc điểm của các bộ CSDL chuẩn

CSDL	Số chuỗi	Số item phân biệt	Số itemset trung bình/chuỗi
Sign	730	267	51
Chess	3196	75	37
Mushroom	8124	119	23
T40I10D100k	100000	942	39

Các kết quả thực nghiệm đo lường về hiệu suất thời gian thực hiện của thuật toán đề xuất WPM so với thuật toán SPMW [15] trên các bộ dữ liệu chuẩn trong bảng 1 được liệt kê trình bày trong bảng 2.

Bảng 2: Kết quả thực nghiệm trên các bộ CSDL với các giá trị *min_sup* khác nhau

CSDL	min_sup (%)	Số mẫu tuần tự có trọng số	Thời gian thực thi (s)		Tỉ lệ (1)/(2)
			SPMW (1)	WPM (2)	
Sign	5	14	0.453	0.296	1.5
	4	56	0.829	0.451	1.8
	3	273	2.721	0.792	3.4
	2	1774	7.804	1.728	4.5
Chess	42	29	1.7	0.5	3.4
	39	552	9	2	4.5
	36	3980	44	7	6.3
	32	24550	Không đủ bộ nhớ	41	Không xác định
Mushroom	16	31	1.766	0.839	2.1
	14	39	2.047	0.854	2.4
	12	97	3.453	1.123	3.1
	10	263	5.989	1.676	3.6
T40I10D100k	75	3	Không đủ bộ nhớ	0.9	Không xác định
	65	9		1.10	
	55	19		1.11	
	45	42		1.75	

Theo kết quả thực nghiệm thống kê trong bảng 2 cho thấy trên cùng một CSDL, cùng ngưỡng hỗ trợ tối thiểu *min_sup*, tập các mẫu tuần tự thu được ở cả hai thuật toán là như nhau. Tuy nhiên, hiệu suất thực thi của thuật toán WPM mà bài báo đề xuất nhanh hơn hẳn so với thuật toán SPMW trong tất cả các trường hợp trên các bộ CSDL thực nghiệm. Đặc biệt khi giá trị *min_sup* càng nhỏ thì càng thấy rõ hơn khả năng thực thi nhanh của WPM so với SPMW. Ví dụ như, với bộ dữ liệu có kích thước nhỏ như Sign (730 chuỗi) thì khả năng thực thi của thuật toán WPM mà bài báo đề xuất có thời gian thực thi tốt hơn thuật toán SPMW trong tất cả các trường hợp. Đối với các CSDL chuỗi có kích thước lớn như Chess (3196 chuỗi) thì thuật toán SPMW chỉ có thể thực hiện với một vài giá trị *min_sup* nhất định nhưng với giá trị *min_sup* nhỏ khoảng 32% trở xuống thì thuật toán SPMW không thể thực thi trong khi đó thuật toán WPM vẫn thực thi tốt trong khoảng 41s. Và với CSDL lớn hơn như T40I10D100k thì thuật toán SPMW không thể thực hiện được do không gian dùng để xây dựng CSDL chiếu các tiền tố quá lớn dẫn đến không đủ bộ nhớ để lưu trữ số lượng các CSDL chiếu do thuật toán tạo ra, trong khi đó, thuật toán WPM thực hiện được trong tất cả trường hợp.

6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Với sự gia tăng không ngừng của công nghệ thông tin và lượng dữ liệu hiện nay, việc khai thác các mẫu dữ liệu có giá trị là rất cần thiết. Khai thác mẫu tuần tự có trọng số giúp tìm ra các mẫu có giá trị cao hơn nên có thể được áp dụng trong nhiều lĩnh vực hơn đồng thời giải quyết một số khó khăn về không gian lưu trữ và tài nguyên thực hiện trong bài toán khai thác mẫu tuần với độ hỗ trợ *min_sup* thấp. Bài báo đã đề xuất một thuật toán hiệu quả để khai thác mẫu tuần tự có trọng số bằng việc kết hợp giá trị trọng số thực của các item trong CSDL chuỗi cùng với độ hỗ trợ của chúng. Bên cạnh đó, thuật toán sử dụng cấu trúc dữ liệu biểu diễn theo chiều dọc nên thuật toán chỉ cần duyệt CSDL một lần, do đó tiết kiệm được thời gian. Hơn nữa, thuật toán đã áp dụng khối mã hóa nguyên tố trong các bước tính toán của quá trình phát triển mẫu làm tăng hiệu suất thực thi của thuật toán so với các tiếp cận khác. Kết quả thực nghiệm cho thấy thuật toán WPM đề xuất trong bài báo có khả năng thực thi tốt và hiệu quả hơn trên các bộ dữ liệu thực có kích thước từ nhỏ như Sign, Chess đến bộ dữ liệu lớn, dày như T40I10D100k.

Trong tương lai, nhóm tác giả sẽ hướng tới việc tối ưu hơn thời gian thực thi và kết hợp thêm các ràng buộc khác để có thể khai thác tập mẫu hiệu quả hơn nữa hoặc đánh giá hiệu suất của thuật toán WPM với thuật toán cải tiến bằng cách áp dụng cách tiếp cận vector bit động [17]. Bên cạnh đó, nhóm tác giả cũng

hướng đến việc phát triển các thuật toán khai thác tập top-k mẫu tuần tự có trọng số với dữ liệu chuỗi ở một số lĩnh vực cụ thể như chuỗi dữ liệu giao dịch, chuỗi dữ liệu khách hàng, chuỗi lịch sử truy cập web,

LỜI CẢM ƠN

Bài báo được thực hiện với sự hỗ trợ từ quỹ đề tài nghiên cứu khoa học của trường đại học Công nghiệp TP HCM, mã số 20/1.6CNTT01

TÀI LIỆU THAM KHẢO

[1]. R. Agrawal, R. Srikant. Mining sequential patterns, *Proceedings of the 11th International Conference on Data Engineering,* pp. 3–14, 1995.

[2]. J. Ayres, J.E. Gehrke, T. Yiu, J. Flannick. Sequential pattern mining using a bitmap representation, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 429–435, 2002.

[3]. V. Chezhian V. Umadevi, J. Celin, S. Geetha. Hierarchical sequence clustering algorithm fordata mining, *Proceedings of the World Congress on Engineering*, pp. 21 – 25, 2011.

[4]. P. Fournier-Viger, A. Gomariz, M. Campos, R. Thomas. Fast vertical mining of sequential patterns using co-occurrence information. *In: PAKDD'14*, pp. 40–52, 2014.

[5]. W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, P. S. Yu. A Survey of Parallel Sequential Pattern Mining, *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, 2019.

[6]. M. N. Garofalakis, R. Rastogi, K. Shim. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints, *Proc. of the Very Large Data Bases Conf., Edinburgh, Scotland, UK*, pp. 223-234, 1999.

[7]. K. Gouda, M. Hassaan, M.J. Zaki. PRISM: a primal-encoding approach for frequent sequence mining, *Journal of Computer and System Sciences,* vol. 76, no. 1, pp. 88–102, 2010.

[8]. J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 355-359, 2000.

[9]. J. Han, M. Kamber. Data Mining: Concepts and Techniques 3nd Edition. *Morgan Kanufmann,* 2012.

[10]. B. Huynh, B. Vo, V. Snasel. An efcient method for mining frequent sequential patterns using multi-core processors, *Applied Intelligence,* vol. 46, no. 3, pp. 703–716, 2017.

[11]. B. Huynh, C. Trinh, H. Huynh, T.T. Van, B. Vo, V. Snasel. An efficient approach for mining sequential patterns using multiple threads on very large databases, *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 242–251, 2018.

[12]. F. Masseglia, F. Cathala, P. Poncelet. The PSP Approach for Mining Sequential Patterns, *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, *Nantes, France*, pp. 176-184, 1998.

[13]. J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach, *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 11, pp. 1424–1440, 2004.

[14]. B. Shim, K. Choi, Y. Suh. CRM strategies for a small-sized online shopping mall based on association rules and sequential patterns, *Expert Systems with Applications,* vol. 39, pp. 7736 – 7742, 2012.

[15]. A. Sirisha, S. Pabboju, G. Narsimha. Efficient mining of sequential patterns in a sequence database with weight constraint, *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4, no. 6, pp. 394 – 397, 2016.

[16]. R. Srikant, R. Agrawal. Mining sequential patterns: Generalizations and performance improvements, *In*: *5th Intl. Conf. Extending Database Technology*, pp. 3 – 17, 1996.

[17]. T. Van, B. Vo, B. Le. Mining sequential patterns with itemset constraints, *Knowledge and Information Systems*, vol. 57, no. 2, pp. 311-330, 2018.

[18]. W. Wang, J. Yang. Mining Sequential Patterns From Large Data Sets*, Springer*, 2005.

[19]. U. Yun, J. Leggett. WFIM: Weighted frequent itemset mining with a weight range and a minimum weight, *In: Fourth SIAM Int. Conf. on Data Mining, USA*, pp. 636–640, 2005.

[20]. U. Yun, J. Leggett. WSpan: Weighted sequential pattern mining in large sequence databases, *3rd International IEEE Conference on Intelligent Systems*, pp. 512 – 517, 2006.

[21]. U. Yun. A new framework for detecting weighted sequential patterns in large sequence databases, *Knowledge – base systems*, vol. 21, pp. 110 – 122, 2008.

[22]. M.J. Zaki. SPADE: an efficient algorithm for mining frequent sequences, *The Journal of Machine Learning Research,* vol. 42, pp. 31–60, 2001.

Ngày nhận bài: 02/10/2019
Ngày chấp nhận đăng: 09/01/2020