Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

FS-ALG: THUẬT TOÁN KHAI THÁC DÃY SỰ KIỆN PHỔ BIẾN

Nguyễn Thị Thu Hiền[1], Văn Thế Thành[2] và Quản Thành Thơ[3]

[1] *Khoa Công nghệ Thông tin, Trường Đại học Công nghệ Tp.HCM*
[2] *Trung tâm Công nghệ Thông tin, Trường Đại học Công nghiệp Thực phẩm Tp.HCM*
[3] *Khoa Khoa học và Kỹ thuật Máy tính, Trường Đại học Bách Khoa Tp.HCM*

Thông tin chung:
Ngày nhận: 19/09/2015
Ngày chấp nhận: 10/10/2015

Title:
FS-Alg: An algorithm for frequent event-based sequences mining

Từ khóa:
Sự kiện, dãy sự kiện, tập phổ biến

Keywords:
Event, Event-based sequence, Frequent item-set

ABSTRACT

Mining the frequent item-set with time factor from transaction database requires much execcution time. Thus, this paper presents an approach of tree structure known as FS-Tree (Frequent Sequence Tree) to store event sequences with appearance time. From there, we propose an algorithm known as FS-Alg (Frequent Sequence Algorithm) to mine the frequent sequence from FS-Tree. To illustrate the effectiveness of the algorithm, the paper compares it with the existing approach of TSET-Miner.

TÓM TẮT

Khai thác tập phổ biến có yếu tố thời gian từ cơ sở dữ liệu giao dịch đòi hỏi nhiều thời gian thực thi. Vì vậy, bài báo tiến hành xây dựng cấu trúc cây FS-Tree (Frequent Sequence Tree) để lưu trữ các dãy sự kiện tổ hợp với thời điểm xuất hiện tương ứng. Từ đó, bài báo đề xuất thuật toán FS-Alg (Frequent Sequence Algorithm) để khai thác các dãy sự kiện phổ biến từ cây FS-Tree. Để minh họa tính hiệu quả, bài báo đánh giá thuật toán đề xuất so với thuật toán tương đồng là TSET-Miner.

1 GIỚI THIỆU

Khai thác dữ liệu là quá trình tìm kiếm các tri thức tiềm ẩn có ích, các tập luật từ cơ sở dữ liệu giao dịch nhằm phục vụ cho các công việc dự báo, ra quyết định. Một số kỹ thuật khai thác dữ liệu ([15],[17],[18]) chỉ quan tâm đến tập phần tử mà bỏ qua yếu tố thời gian. Trong khi đó, thuộc tính thời gian có ý nghĩa rất quan trọng và có yếu tố quyết định đối với nhiều chiến lược dự đoán thuộc nhiều lĩnh vực như kinh doanh, thương mại, thị trường chứng khoán... Vì vậy, khai thác dữ liệu có yếu tố thời gian là một chủ đề có vai trò quan trọng trong khai thác dữ liệu.

Có nhiều kỹ thuật để khai thác dãy sự kiện phổ biến như Mô hình cơ sở dữ liệu thời gian và ràng buộc thời gian của Mkaouar M. *et al*. (2011) [2], Khai thác hiệu quả luật kết hợp liên giao dịch của Tung, A.K.H. *et al* (2003) [1], Truy vấn và thao tác trên cơ sở dữ liệu thời gian của Mkaouar M. *et al*. (2011) [3], Khai thác các đoạn phổ biến trong dãy sự kiện của H. Mannila *et al*. (1997) [4], Khai thác dãy sự kiện phổ biến sử dụng cây *Seq-Tree* (2015) [16], Một cấu trúc cây để khai thác dãy sự kiện của Francisco Guil *et al*. (2012) [7],... Trong đó, thuật toán *TSET-Miner* khai thác các dãy sự kiện phổ biến dựa trên cấu trúc cây *TSET* [7]. Thuật toán này phụ thuộc vào khung thời gian và độ phổ biến. Nếu khung thời gian hoặc độ phổ biến thay đổi thì phải xây dựng lại cây. Việc xây dựng cây *TSET* tốn kém rất nhiều thời gian, ứng mỗi nút con trong cây được sinh ra thì phải duyệt lại toàn bộ cơ sở dữ liệu.

Do đó, để khắc phục nhược điểm của thuật toán *TSET-Miner*, bài báo đề xuất cây *FS-Tree* để lưu trữ các dãy sự kiện tổ hợp ứng với từng thời điểm xuất hiện của tập phần tử. Việc xây dựng cây *FS-*

Tree chỉ cần duyệt cơ sở dữ liệu giao dịch đúng 1 lần. Thuật toán *FS-Alg* thực hiện trích xuất các dãy sự kiện phổ biến từ cây *FS-Tree* ứng với khung thời gian và độ phổ biến khác nhau do người dùng chỉ định. Vì vậy, khi thay đổi khung thời gian hoặc độ phổ biến thì chỉ thực hiện trích xuất mà không cần phải duyệt lại cơ sở dữ liệu. Thuật toán *FS-Alg* đã giải quyết được các yếu điểm còn tồn tại của thuật toán *TSET-Miner*, giúp rút ngắn thời gian thực thi.

Phần còn lại của bài báo được tổ chức như sau: ***Phần 2*** giới thiệu các công trình liên quan. ***Phần 3*** trình bày cơ sở lý thuyết xây dựng thuật toán *FS-Alg*. ***Phần 4*** trình bày thuật toán *FS-Alg*. ***Phần 5*** minh họa ví dụ của thuật toán *FS-Alg*. Kết luận và hướng phát triển được mô tả tại ***Phần 6***.

2 CÁC CÔNG TRÌNH LIÊN QUAN

Có rất nhiều công trình nghiên cứu về cách khai thác dữ liệu có yếu tố thời gian nhằm đưa ra các dãy sự kiện phổ biến phục vụ cho công việc dự báo. Một số công trình nghiên cứu lĩnh vực khai thác dữ liệu thời gian như Thuật toán dựa trên cấu trúc băm để khai thác luật kết hợp của Park *et al*. (1995) [8], Thuật toán hiệu quả để khai thác các dãy tuần tự phổ biến của M.J. Zaki *et al*. (2001) [10], Cây mẫu phổ biến mở rộng để khai thác luật kết hợp liên giao dịch của Lu *et al*. (2005) [9], Khai thác luật kết hợp từ dãy tuần tự thời gian của K. Bouandas *et al*. (2007) [11], Khai thác luật kết hợp theo thời gian phổ biến trên cơ sở dữ liệu sách xuất bản của C.H. Lee *et al*. (2001) [12],... Có nhiều cách tổ chức cấu trúc của cơ sở dữ liệu theo thời gian. Ứng với mỗi cách sẽ có các phương pháp khai thác khác nhau.

Gần đây, có một số công trình nghiên cứu như công trình nghiên cứu của Chun-Sheng Wang *et al*. (2009) về "Khai thác các mẫu liên giao dịch" [5]. Phương pháp khai thác theo đề xuất của bài báo thì tổng quát hơn phương pháp khai thác mẫu tuần tự hoặc phương pháp khai thác mẫu liên giao dịch. Thuật toán *EISP-Miner* có thể khai thác cả mẫu tuần tự trong phạm vi một giao dịch và các mẫu tuần tự xuyên suốt một số giao dịch khác nhau dựa trên cấu trúc cây *ISP-Tree*. Cơ sở dữ liệu giao dịch này được ấn định chính xác về thời gian, mỗi thời điểm ứng với một tập các dãy sự kiện. Phương pháp đề xuất có nhược điểm là sinh ra một số lượng lớn các mẫu phổ biến. Điều này có thể dẫn đến việc nhiều mẫu phổ biến sẽ bị dư thừa. Ngoài ra, việc xây dựng cây *ISP-Tree* phụ thuộc vào khung thời gian và độ phổ biến. Vì vậy, khi có sự thay đổi về khung thời gian hoặc độ phổ biến thì phải xây dựng lại cây *ISP-Tree*.

Công trình nghiên cứu để tìm ra "Thuật toán hiệu quả để khai thác sự gia tăng của các luật kết hợp theo thời gian" của Tarek F. Gharib *et al*. (2010) [14]. Công trình này đề cập đến việc xử lý chuỗi thời gian bằng cách thêm chuỗi thời gian vào các luật kết hợp. Nếu cơ sở dữ liệu thay đổi thì việc thực thi lại thuật toán sẽ bỏ qua các luật được tìm thấy trước đó. Vì vậy, thuật toán này chỉ xử lý trên những phần dữ liệu được chèn thêm hoặc cập nhật, đồng thời kết hợp với các dãy phổ biến được tìm thấy trước đó. Điều này giúp rút ngắn thời gian khai thác so với các phương pháp khai thác trên cơ sở dữ liệu cập nhật khác [9][12][6].

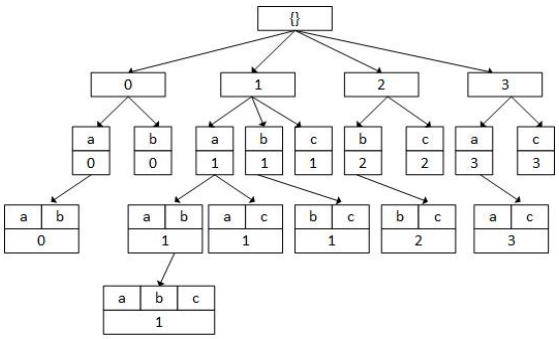
Một đóng góp khác là công trình nghiên cứu của Chun-Sheng Wang *et al*. (2013) [13], "Khai thác mẫu liên dãy tuần tự đóng". Bài báo nêu ra vấn đề là khi khai thác dãy tuần tự phổ biến sẽ tạo ra số lượng lớn các mẫu phổ biến hồi quy. Bài báo đề xuất thuật toán *CISP-Miner* để giải quyết vấn đề trên. Kết quả là tốt hơn trong một số trường hợp so với thuật toán *EISP-Miner* [5].

Nghiên cứu về "Cách tiếp cận mới để khai thác các tập phần tử liên giao dịch" của S. Nandagopal *et al*. (2014) [6]. Thuật toán *IAR Miner* (Inter-transaction Association Rule Miner) được đề xuất để khai thác các tập phần tử liên giao dịch phổ biến trên cơ sở dữ liệu được ấn định chính xác về thời gian, mỗi thời điểm ứng với một tập các phần tử. Ban đầu, duyệt toàn bộ cơ sở dữ liệu để tính toán độ phổ biến của tập phần tử. Tiếp theo, xây dựng *Hash Table* và cây *ID-Tree* (Itemset – dataset tree). Mỗi nút của *ID-Tree* sẽ có cấu trúc là *ID-pair* (Itemset – dataset pair). Trong quá trình xây dựng cây *ID-Tree*, bài báo áp dụng chiến lược cắt tỉa nhằm giảm không gian tìm kiếm. Thuật toán *IAR Miner* khai thác hiệu quả tập liên giao dịch đóng phổ biến, không gian tìm kiếm được thu gọn, thời gian thực thi được rút ngắn và đòi hỏi bộ nhớ lưu trữ ít. Tuy nhiên, thuật toán *IAR Miner* có nhược điểm là nếu thay đổi độ phổ biến và khung thời gian thì phải duyệt lại toàn bộ cơ sở dữ liệu.

Trên cơ sở các công trình liên quan đã cho thấy sự phù hợp và tính quan trọng của quá trình khai thác dữ liệu có yếu tố thời gian. Các đóng góp của bài báo gồm: (1) Xây dựng cấu trúc cây *FS-Tree* để làm giảm thời gian tổ chức dữ liệu khi có sự thay đổi; (2) Đề xuất thuật toán *FS-Alg* để truy xuất hiệu quả các dãy sự kiện phổ biến có yếu tố thời gian.

3 CƠ SỞ LÝ THUYẾT

Nút gốc của cây *FS-Tree* là nút rỗng. Mỗi thời điểm xuất hiện trong cơ sở dữ liệu sẽ tạo thành 1 nút trong cây ở mức 1. Các nút ở mức 2 gồm các sự kiện ứng với thời điểm mà các sự kiện đó thuộc về, tức là các sự kiện có cùng thời điểm sẽ thuộc về cùng 1 nhánh. Trong cùng 1 nhánh, các nút sẽ tổ hợp lần lượt với nhau để tạo thành các nút ở các mức sâu hơn.



Hình 1: Một ví dụ về cây FS-Tree

Định nghĩa 2.1.1 (*Nút mức 2 trong cây*). Mỗi nút $Node$ trong cây *FS-Tree* bao gồm tập một dãy sự kiện $Node.seq$, tập các thời điểm xuất hiện dãy sự kiện $Node.time$ và tập các liên kết đến nút con $Node.link$

Định nghĩa 2.1.2 (*Nút mức 1 trong cây*). Mỗi nút $Ntime$ trong cây gồm thời điểm xuất hiện của dãy sự kiện $Ntime.time$ và tập liên kết $Ntime.link$ đến các nút mức 2.

Định nghĩa 2.1.3 (*Nút gốc trong cây*). Một nút gốc $RNode$ là nút rỗng có tập các liên kết $RNode.link$ đến các nút con.

Định nghĩa 2.1.4 (*Cây FS-Tree*). Một cây *FS-Tree* = $(RNode, S_{Node})$ gồm một nút gốc $RNode$ và một tập nút phân cấp trong cây $S_{Node} = \{Node_1, Node_2, \dots, Node_N\}$. Tập các nút phân cấp S_{Node} bao gồm các nút cha liên kết đến các nút con dựa trên tập các liên kết $Node.link$ tại mỗi nút $Node$ trong cây.

Nhánh trong cây là đường đi liên kết đến các nút con liên tiếp từ $Node_i$ đến $Node_j$ sao cho giữa 2 nút kề nhau đều có nhánh.

4 THUẬT TOÁN FS-ALG

4.1 Mô tả thuật toán

Thuật toán *FS-Alg* được áp dụng trên cơ sở dữ liệu DB có cấu trúc như sau:

Date	Item list
0	$a\ b$
1	$a\ b\ c$
2	$b\ c$
3	$a\ c$

Cây *FS-Tree* có nút gốc là nút rỗng và tập liên kết đến các nút con $link$. Các nút $Ntime$ ở mức 1 chứa các thời điểm xuất hiện của tập phần tử trong cơ sở dữ liệu DB. Nút này bao gồm thời điểm xuất hiện của sự kiện $time$, tập các liên kết đến nút con $link$. Mỗi thời điểm sẽ tạo thành một nhánh trong cây *FS-Tree*. Lần lượt chèn từng sự kiện vào cây *FS-Tree* với nút gốc là nút có cùng thời điểm với sự kiện đang xét. Các nút con ở mức 2 trong cây *FS-Tree* được tạo ra bằng cách tổ hợp với các nút đồng cấp và có cùng thời điểm, tức là thuộc cùng 1 nhánh. Mỗi nút $Node$ trong cây thuộc mức 2 xuống các mức sâu hơn gồm dãy sự kiện seq, thời điểm $time$, liên kết đến nút con $link$ của dãy sự kiện.

Duyệt nhánh ứng với nút chứa thời điểm 0, chèn các dãy sự kiện vào tập X_1. Xét nhánh chứa thời điểm 1, trích xuất các dãy sự kiện chứa trong nhánh này và chuyển vào tập Y. Thực hiên tích đề-các giữa tập X_1với tập Y và chuyển vào tập X_2. Sau đó, chuyển tập X_2 vào tiếp sau tập X_1. Thực hiện tương tự cho nhánh chứa thời điểm 2, 3 ứng với thời điểm gốc là 0.

Với thời điểm 1, sao chép các dãy sự kiện ở thời điểm 0 được lưu trong tập X_1 vào tập X_2. Khi sao chép, trong từng dãy sự kiện, chỉ lấy các sự kiện có thời điểm lớn hơn hoặc bằng 1. Nếu dãy sự kiện nào xuất hiện từ 2 lần trở lên thì chỉ giữ lại 1 dãy. Sao chép lần lượt từng dãy sự kiện trong tập X_2, đồng thời chuẩn hóa và chuyển sang tập X_1. Nếu dãy sự kiện sao chép được chuẩn hóa đã tồn tại trong tập X_1 thì tăng số lần xuất hiện. Ngược lại thì chuyển tiếp vào sau X_1.

Tiếp tục với thời điểm 2, xét tập X_2 đã có ở thời điểm 1. Duyệt tập X_2, chỉ giữ lại các sự kiện có thời điểm lớn hơn hoặc bằng 2. Nếu dãy sự kiện nào xuất hiện từ 2 lần trở lên thì chỉ giữ lại 1 dãy. Sao chép lần lượt từng dãy sự kiện trong tập X_2, đồng thời chuẩn hóa và chuyển sang tập X_1. Nếu dãy sự kiện sao chép được chuẩn hóa đã tồn tại trong tập X_1 thì tăng số lần xuất hiện. Ngược lại thì chuyển tiếp vào sau X_1.

Thực hiện tương tự như thời điểm 2 cho các thời điểm còn lại trên cây *FS-Tree* sẽ được tập X_1 hoàn chỉnh.

Tiến hành trích xuất trên tập X_1 này sẽ thu được các dãy sự kiện phổ biến ứng với các khung thời gian và các độ phổ biến khác nhau.

Nếu có sự thay đổi về cơ sở dữ liệu thì thực hiện thao tác cập nhật trên cây *FS-Tree*. Sau đó, áp dụng thuật toán *FS-Alg* để trích xuất các dãy sự kiện phổ biến từ cây *FS-Tree*.

4.2 Thuật toán FS-Alg

Tạo cây *FS-Tree* cần đối số đầu vào là tập các thời điểm và tập các sự kiện ứng với thời điểm xuất hiện thuộc cơ sở dữ liệu DB, đầu ra là cây *FS-Tree*. Đầu tiên, khởi tạo nút gốc là nút rỗng. Duyệt cơ sở dữ liệu DB, chèn thời điểm đầu tiên vào dưới nút gốc bằng thuật toán **insertNode**. Chèn sự kiện thứ nhất vào dưới nút có cùng thời điểm với sự kiện đang xét bằng thuật toán **insertNode**. Tạo nhánh cho nút này bằng thuật toán **CreateBranch**.

Thực hiện tương tự cho các sự kiện còn lại ứng với thời điểm đang xét. Sau đó, tiếp tục xét các thời điểm và tập các sự kiện còn lại trong cơ sở dữ liệu DB sẽ tạo được cây *FS-Tree* phù hợp với dữ liệu đầu vào.

Thuật toán 3.2.1: FS-Tree ($Date$, $Itemlist$)

Đầu vào: $(time_i \wedge Itemlist_i) \in DB$

Đầu ra: Cây *FS-Tree*

Begin

$RNode.seq = \emptyset$;

$RNode.link = NULL$;

$FS\text{-}Tree = RNode$;

Foreach $time_i \in Date$ **do**

Flag = **insertNode** ($time_i, NULL$);

Foreach $Item_k \in Itemlist_i$ **do**

Flag = **insertNode** ($time_i, Item_k$);

If (Flag = True) **then createBranch** ($Item_k, FS\text{-}Tree$);

End Foreach

End Foreach

End.

Các nút con của cây *FS-Tree* được xây dựng bằng cách tổ hợp các nút có chứa sự kiện thuộc cùng thời điểm ở mức 2, tức là thuộc cùng 1 nhánh. Thao tác này chính là tạo nhánh cho cây. Thực hiện tạo nhánh cho cây cần đối số đầu vào là nút $Node$ cần tạo nhánh, cây *FS-Tree*. Đầu ra là cây *FS-Tree* đã tạo nhánh. Lấy các nút có liên kết với nút có cùng thời điểm với nút $Node$ và đưa vào hàng đợi. Lấy nút thứ nhất trong hàng đợi tổ hợp với nút $Node$, chuyển vào tập con của nút đang xét, lấy các nút có liên kết với nút thứ nhất chuyển vào hàng đợi. Thực hiện tương tự cho các nút còn lại trong hàng đợi cho đến khi hàng đợi rỗng thì dừng lại.

Thuật toán 3.2.2: createBranch ($Node$, *FS-Tree*)

Đầu vào: $Node$, *FS-Tree*

Đầu ra: Cây *FS-Tree* đã tạo nhánh

Begin

Count1 = 0; Count2 = 0;

Queue = $\emptyset$; QueueNode = $\emptyset$;

Foreach ($Node_i \in Ntime.link$) **do**

Queue.push($Node_i$);

Count1 = Count1 + 1;

Count2 = Count2 + 1;

End Foreach

QueueNode.push($Node$);

While (QueueNode $\neq \emptyset$) **do**

Count1 = Count2; Count2 = 0;

For j=1..Count1 **do**

$Ntp1$ = Queue.pop();

$Ntp2$ = QueueNode.pop();

If ($Ntp_1.time = Ntp_2.time$) **then**

$UNode = Ntp1 \cup Ntp2$;

QueueNode.push($UNode$);

End If

End For

Foreach ($Node_i \in Node.link$) **do**

Queue.push($Node_i$);

Count2 = Count2 + 1;

End Foreach

End While

Return True;

End.

Thao tác chèn nút vào cây cần đối số đầu vào là thời điểm và tập các sự kiện thuộc thời điểm đang xét. Đầu ra là cây *FS-Tree* đã được cập nhật. Chèn nút vào cây có các trường hợp có thể xảy ra:

Trường hợp 1: Thời điểm thêm vào là mới. Thực hiện tạo nhánh mới ứng với nút chứa thời đang xét. Chèn lần lượt từng sự kiện thuộc thời điểm thêm đó vào cây ở mức tiếp theo.

Trường hợp 2: Thời điểm đã tồn tại trên cây, sự kiện thuộc thời điểm đó cũng đã có trên cây thì không thực hiện thao tác chèn vào cây.

Trường hợp 3: Thời điểm đã tồn tại trên cây, sự kiện là mới ứng với thời điểm đó. Thực hiện chèn sự kiện đó vào cây và thực hiện tổ hợp với các nút đã có trong nhánh đang xét.

Thuật toán 3.2.3: insertNode(Date, Itemlist)

Đầu vào: $time_i \wedge itemlist_i$, cây *FS-Tree*

Đầu ra: Cây *FS-Tree* đã được cập nhật

Begin

$S = \cup \{time \in level1\}$;

If ($time_i \notin S$) **then**

$S = S \cup time_i$;

createBranch($Ntime_i, FS-Tree$);

End If

If ($time_i \in S$) **then**

$S_1 = \cup \{Node_i \in Ntime_i\}$;

If ($item_k \notin S_1$) **then** return;

Else If

createBranch($Node_k, FS-Tree$);

End If

End If

Return True;

End.

Trích xuất các dãy sự kiện phổ biến từ cây *FS-Tree* cần đầu vào là cây *FS-Tree*, khung thời gian W, ngưỡng θ; đầu ra là tập các dãy sự kiện phổ biến $Fseq$ thỏa yêu cầu đầu vào. Lần lượt xét từng nhánh ứng với từng thời điểm. Duyệt nhánh chứa thời điểm 0 bằng thuật toán **getX** và chuyển vào tập X_1, duyệt nhánh chứa thời điểm 2 bằng thuật toán **getX** vào tập Y, thực hiện tích đề-các giữa X_1 và Y chuyển vào tập X_2. Thực hiện lần lượt cho các nhánh còn lại ứng với thời điểm gốc là 0.

Với thời điểm gốc là 1, gán tập X_1 vào tập X_2. Trong từng dãy sự kiện của tập X_2, chỉ giữ lại các sự kiện có thời điểm lớn hơn hoặc bằng 1. Nếu các dãy sự kiện trùng nhau thì chỉ giữ lại 1 dãy sự kiện. Sao chép và chuẩn hóa từng dãy sự kiện trong tập X_2 và chuyển vào tập X_1. Thực hiện tương tự cho các thời điểm còn lại trên cây *FS-Tree*. Trích xuất trên tập X_1 sẽ thu được các dãy sự kiện phổ biến thỏa yêu cầu.

Thuật toán 3.2.3: FS-Alg(*FS-Tree*,W, θ)

Đầu vào: Cây *FS-Tree*, khung thời gian W, ngưỡng θ

Đầu ra: Tập các dãy sự kiện phổ biến $Fseq$ thỏa khung thời gian W, ngưỡng θ

Begin

$X_1 = \emptyset; X_2 = \emptyset; X_3 = \emptyset; Y = \emptyset; i = 0;$

$X_1 =$ **getX** ($Ntime_i.time$, *FS-Tree*);

Foreach $Ntime_{i+1} \in RNode.link$ **do**

$Y =$ **getX** ($Ntime_{i+1}.time$, *FS-Tree*);

$X_2 = X_1 \times Y; X_1 = X_1 \oplus X_2;$

$X_2 = \emptyset; Y = \emptyset;$

End Foreach

$i = 1; X_2 = X_1;$

Foreach $Ntime_i \in RNode.link$ **do**

If ($X_2.time = Ntime_{i-1}$) **then**

Delete($X_2.event$);

End If

If (Countseq(X_2)≥ 2) **then** Delete($X_2.seq$);

If ($X_2.time = Ntime_i$) **then**

Delete($X_2.seq$);

End If

$X_3 = X_2;$

Normal(X_2); $X_1 = X_1 \oplus X_2;$

End Foreach

For $x_i \in X_1$ **do** ExtractFSeq(x_i, W, θ);

Return $Fseq$;

End.

Duyệt nhánh cây *FS-Tree* theo thời điểm cần đầu vào là cây *FS-Tree* và thời điểm $time$, đầu ra là tập X chứa các dãy sự kiện thỏa đầu vào. Lần lượt chèn từng nút có liên kết với nút chứa thời điểm $time$ vào hàng đợi. Xét nút đầu tiên trong hàng đợi, chuyển sự kiện trong nút này vào tập X, lấy các nút có liên kết với nút đầu tiên và chuyển vào hàng đợi. Thực hiện tương tự như vậy cho đến khi hàng đợi rỗng thì dừng lại.

Thuật toán 3.2.4: getX (*FS-Tree*, $time$)

Đầu vào: Cây *FS-Tree*, $time$

Đầu ra: Tập X chứa các dãy sự kiện thuộc nhánh có thời điểm $time$

Begin

$X = \emptyset$; count = 0; Queue = $\emptyset$;

Foreach $Node_i \in Ntime.link$ **do**

Queue.push($Node_i$);

count = count + 1;

End Foreach

While (Queue ≠ $\emptyset$) **do**

For j=1..count -1 **do**

$Node1$ = Queue.pop();

$X = X + Node_i.seq$;

Foreach $Node_k \in Node1.link$ **do**

Queue.push ($Node_k$);

count = count + 1;

End Foreach

End For

End while

Return X;

End.

Gọi N là số sự kiện có trong cơ sở dữ liệu. Mỗi nhánh của cây *FS-Tree* có n_i số sự kiện ($n_i \ll N$). Do đó, độ phức tạp của thuật toán FS-Alg là O(N).

5 VÍ DỤ MINH HỌA

Bài báo tiến hành ví dụ minh họa trên cơ sở dữ liệu *DB* được mô tả tại mục **4.1**. Thực hiện xây dựng cây *FS-Tree* bằng cách duyệt lần lượt từng thời điểm và $Itemlist$ ứng với thời điểm đó. Ở mức 0, tạo nút gốc $RNode$ rỗng và $RNode.link = NULL$.

Xét thời điểm 0, chèn thời điểm 0 vào dưới nút gốc (mức 1), $Ntime.time = 0$. Khi đó, $RNode$ sẽ trỏ đến các nút được tạo ra ở mức 1. Ở mức 2, chèn phần tử a vào dưới nút chứa thời điểm 0, $Node.seq = a$, $Node.time = 0$. Chèn phần tử b, $Node.seq = b$, $Node.time = 0$. Thực hiện tổ hợp từng cặp nút ở mức 2 sẽ sinh ra các nút ở mức 3 với $Node.seq = ab$, $Node.time = 0$.

Xét thời điểm 1, chèn thời điểm 1 vào dưới nút gốc. Ở mức 2, chèn lần lượt từng phần tử xuất hiện tại thời điểm 1 vào dưới nút chứa thời điểm 1. Các nút ở mức 3 được tạo thành bằng cách tổ hợp giữa các nút ở mức 2 thuộc cùng thời điểm, với nút gốc là nút chứa thời điểm đang xét. Thực hiện tương tự cho các thời điểm 2, 3 sẽ được cây *FS-Tree* hoàn chỉnh như Hình 1.

Áp dụng thuật toán *FS-Alg* để trích xuất các dãy sự kiện phổ biến từ cây *FS-Tree*. Duyệt nhánh chứa thời điểm 0 và chèn các sự kiện tổ hợp vào tập $X_1 = \{(a_0, 1); (b_0, 1); (a_0b_0, 1)\}$. Xét nhánh chứa thời điểm 1 và chèn vào tập $Y = \{(a_1, 1); (b_1, 1); (c_1, 1); (a_1b_1, 1); (a_1c_1, 1); (b_1c_1, 1); (a_1b_1c_1, 1)\}$. Tạo tập X_2 chứa kết quả thực hiện tích đề - các giữa X_1 và Y, $X_2 = \{(a_0a_1, 1); (a_0b_1, 1); (a_0c_1, 1); (a_0a_1b_1, 1); (a_0a_1c_1, 1); (a_0b_1c_1, 0); (a_0a_1b_1c_1); (b_0c_1, 1); (a_0c_2, 1) \ldots\}$. Chuyển tập X_2 vào tiếp sau tập $X_1 = \{(a_0, 1); (b_0, 1); (a_0b_0, 1); (a_0a_1, 1); (a_0b_1, 1); (a_0c_1, 1); (a_0a_1b_1, 1); (a_0a_1c_1, 1); (a_0b_1c_1, 1); (a_0a_1b_1c_1, 1); (b_0c_1, 1); \ldots$Thực hiện tương tự cho nhánh chứa thời điểm 2, 3 ứng với thời điểm 0 là thời điểm ban đầu.

Xét thời điểm 1, gán tập X_1 cho tập X_2. Mỗi dãy sự kiện của tập X_2, chỉ lấy các sự kiện có thời điểm lớn hơn hoặc bằng 1. Khi đó, dãy sự kiện $(c_1, 1)$ xuất hiện 2 lần nên chỉ giữ lại 1 dãy sự kiện. Tập $X_2 = \{(a_1, 1); (b_1, 1); (c_1, 1); (a_1b_1, 1); (a_1c_1, 1); (b_1c_1, 1); (a_1b_1c_1, 1) \ldots\}$.

Sao chép và chuẩn hóa từng dãy sự kiện trong tập X_2 và chuyển vào X_1.

$X_1 = \{(a_0, 2); (b_0, 2); (a_0b_0, 2); (a_0a_1, 1); (a_0b_1, 1); (a_0c_1, 1); (a_0a_1b_1, 1); (a_0a_1c_1, 1); (a_0b_1c_1, 1); (a_0a_1b_1c_1, 1); (b_0c_1, 1); (c_0, 1); (a_0c_0, 1); (b_0c_0, 1); (a_0b_0c_0, 1) \ldots\}$

Các thời điểm 2, 3 thực hiện tương tự như thời điểm 1 trên tập X_2.

Với khung thời gian $W = 0$, $\theta = 2$, trích xuất trên tập X_1 sẽ thu được 6 dãy sự kiện phổ biến.

Khung thời gian $W = 1$ sẽ bao gồm khung thời gian $W = 0$. Vì vậy, số lượng dãy sự kiện phổ biến được trích xuất với khung thời gian $W = 1$ sẽ bằng 19 dãy sự kiện phổ biến.

Khung thời gian $W = 2$ sẽ bao gồm khung thời gian $W = 0, 1$. Do đó, số lượng dãy sự kiện phổ biến sẽ là 31 dãy sự kiện phổ biến.

Bảng 1: Các dãy sự kiện phổ biến được trích xuất bằng thuật toán FS-Alg với $\theta = 2, W = 0, 1, 2$

W	Số lượng $Fseq$	$Fseq$
0	6	$(a_0, 3); (b_0, 3); (a_0b_0, 2);\ \ (c_0, 3); (a_0c_0, 2); (b_0c_0, 2)$
1	19	$(a_0, 3); (b_0, 3); (a_0b_0, 2);\ \ (c_0, 3); (a_0c_0, 2); (b_0c_0, 2)$ $(a_0b_1, 2); (a_0c_1, 2); (a_0b_1c_1, 2); (b_0a_1, 2);$ $(b_0b_1, 2); (b_0c_1, 2);$ $(b_0a_1c_1, 2); (b_0b_1c_1, 2);$ $(a_0b_0b_1, 2); (a_0b_0c_1, 2);$ $(a_0b_0b_1c_1, 2); (c_0c_1, 2);$ $(b_0c_0c_1, 2)$
2	31	$(a_0, 3); (b_0, 3); (a_0b_0, 2);\ \ (c_0, 3); (a_0c_0, 2); (b_0c_0, 2)$ $(a_0b_1, 2); (a_0c_1, 2);$ $(a_0b_1c_1, 2); (b_0a_1, 2);$ $(b_0b_1, 2); (b_0c_1, 2);$ $(b_0a_1c_1, 2); (b_0b_1c_1, 2);$ $(a_0b_0b_1, 2); (a_0b_0c_1, 2);$ $(a_0b_0b_1c_1, 2); (c_0c_1, 2);$ $(b_0c_0c_1, 2); (a_0c_2, 2); (b_0c_2, 2);\ (a_0b_0c_2, 2); (a_0b_1c_2, 2); (a_0c_1c_2, 2);$ $(a_0b_1c_1c_2, 2);$ $(b_0b_1c_2, 2); (b_0c_1c_2, 2);$ $(b_0b_1c_1c_2, 2);$ $(a_0b_0b_1c_2, 2);$ $(a_0b_0c_1c_2, 2);$ $(a_0b_0b_1c_1c_2, 2)$

Như vậy, các dãy sự phổ biến được sinh ra bằng thuật toán *FS-Alg* hoàn toàn giống với các dãy sự kiến phổ biến được sinh ra bằng thuật toán *TSET-Miner*.

6 KẾT LUẬN

Bài báo đã giải quyết được vấn đề trích xuất các dãy sự kiện phổ biến ứng với các khung thời gian khác nhau và với các độ phổ biến khác nhau trên cơ sở dữ liệu tăng trưởng. Thời gian trích xuất được rút ngắn rất nhiều so với khi thực hiện trích xuất bằng thuật toán *TSET-Miner*. Điều này đã được minh chứng trong phần kết quả thực nghiệm. Quá trình xây dựng cây *FS-Tree* chỉ cần duyệt cơ sở dữ liệu 1 lần, giúp tiết kiệm rất nhiều chi phí so với khi xây dựng cây *TSET*. Hướng phát triển của bài báo là đề xuất phương pháp để tối ưu việc lưu trữ dữ liệu thời gian sao cho việc truy xuất sẽ tiêu tốn ít thời gian nhất.

LỜI CẢM TẠ

Nhóm tác giả xin chân thành cảm ơn Khoa Khoa học và Kỹ thuật Máy tính, trường Đại học Bách khoa Tp.HCM và Trung tâm Công nghệ Thông tin, trường Đại học Công nghiệp Thực phẩm Tp.HCM là nơi bảo trợ để thực hiện nghiên cứu này.

TÀI LIỆU THAM KHẢO

1. Tung, A.K.H., H. Lu, J. Han and L. Feng, 2003. Efficient mining of inter-transaction association rules. IEEE Trans. Knowl. Data Eng., 15: 43-56.
2. Mkaouar M., Bouaziz R., Moalla M., 2011. Modelling temporal databases and temporal constraints, paper at the Third International Conference on Advances in Databases, Knowledge, and Data Applications.
3. Mohamed Mkaouar, Rafik Bouaziz, and Mohamed Moalla, 2011. Querying and manipulating temporal databases, International Journal of Database Management Systems, Vol.3, No.1.
4. H. Mannila, H. Toivonen, A.I. Verkamo, 1997. Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery 1 (3): 259–289.

5. Chun-Sheng Wang, Anthony J.T. Lee, 2009. Mining inter-sequence patterns, Expert Systems with Applications 36: 8649–8658.
6. S. Nandagopal, V.P. Arunachalam, S. Karthik, 2004. A novel approach for mining inter-transaction itemsets, vol. 8, No.14: 1857 – 7881.
7. Francisco Guil, Roque Marín, 2012. A tree structure for event-based sequence mining, Knowledge-Based Systems 35: 186–200.
8. Park, J.S., M.S. Chen, P.S. Yu, 1995. An effective hash-based algorithm for mining association rules. Proceedings of the ACM-SIGMOD International Conference on Management of Data, (ICMD' 95), ACM Press New York, NY, USA, pp: 175-186.
9. Lu, H.S., G. West and S. Venkatesh, 2005. An extended frequent pattern tree for intertransaction association rule mining. Curtin University of Technology, Perth, Western Australia.
10. M.J. Zaki, SPADE, 2001. An efficient algorithm for mining frequent sequences, Machine Learning 42 (1/2): 31–60.
11. K. Bouandas, A. Osmani, 2007. Mining association rules in temporal sequences, in: Proc. of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM'07), IEEE Computer Society.
12. C.H. Lee, C.R. Lin, M.S. Chen, 2001. On mining general temporal association rules in a publication database, in: Proc. of the IEEE Int. Conf. on Data Mining (ICDM'01), IEEE Computer Society.
13. Chun-Sheng Wang, Ying-Ho Liu, Kuo-Chung Chu, 2013. Closed inter-sequence pattern mining, Journal of Systems and Software Volume 86, Issue 6: 1603–1612
14. Tarek F. Ghari, Hamed Nassar, Mohamed Taha, Ajith Abraham, 2010. An efficient algorithm for incremental mining of temporal association rules, Data & Knowledge Engineering, Volume 69, Issue 8: 800–815.
15. Wei-Wen Wu, Yu-Ting Lee, Ming-Lang Tseng, Yi-Hui Chiang, 2010. Data mining for exploring hidden patterns between KM and its performance, Knowledge-Based Systems 23: 397–401.
16. Nguyễn Thị Thu Hiền, Lê Hữu Hà, Văn Thế Thành. Khai thác dãy sự kiện phổ biến sử dụng cây Seq-Tree, Tạp chí Khoa học – Đại học Huế, Tập 106, Số 07, 2015, Tr. 109-120
17. Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S. Yu, 2010. UP-Growth: An Efficient Algorithm for High Utility Itemset Mining, KDD'10, Washington, DC, USA.
18. Ming-Yen Lin, Tzer-Fu Tu, Sue-Chen Hsueh, 2012. High utility pattern mining using the maximal itemset property and lexicographic tree structures, Information Sciences, No. of Pages 14, Model 3G.