DOI: 10.15625/vap.2020.00217

THUẬT TOÁN HIỆU QUẢ KHAI THÁC MẪU PHỔ BIẾN TRÊN CƠ SỞ DỮ LIỆU ĐỊNH LƯỢNG TĂNG TRƯỞNG

Vương Đình Bắc, Nguyễn Duy Hàm
Trường Cao đẳng An ninh mạng iSPACE
bac.vuong@ispace.edu.vn, ham.nguyen@ispace.edu.vn

TÓM TẮT: *Khai thác mẫu phổ biến trên cơ sở dữ liệu giao dịch là một trong những bài toán quan trọng trong khai phá dữ liệu nói chung. Trong đó, khai thác mẫu phổ biến trên cơ sở dữ liệu tăng trưởng là bài toán có nhiều ứng dụng trong thực tế, do cơ sở dữ liệu thường xuyên có sự cập nhật bổ sung các giao dịch mới. Thời gian gần đây đã có nhiều nghiên cứu liên quan đến bài toán này trên cơ sở dữ liệu loại này, trên cả cơ sở dữ liệu nhị phân và cơ sở dữ liệu định lượng. Tuy nhiên, các giải pháp đề xuất trên cơ sở dữ liệu tăng trưởng hiện tại vẫn còn tốn khá nhiều thời gian trong khai thác. Bài báo này chúng tôi đề xuất một thuật toán hiệu quả khai thác tập phổ biến trên cơ sở dữ liệu định lượng tăng trưởng dựa trên cây ITW-Tree. Thuật toán do chúng tôi đề xuất đạt hiệu quả tốt hơn các phương pháp đã có trên các CSDL thử nghiệm.*

Từ khóa: *Mẫu phổ biến, cơ sở dữ liệu tăng trưởng, cơ sở dữ liệu định lượng, cơ sở dữ liệu định lượng tăng trưởng..*

I. GIỚI THIỆU

Bài toán khai thác mẫu phổ biến là bài toán cơ bản và quan trọng của lĩnh vực khai thác dữ liệu [1-4]. Từ bài toán này, một loạt các bài toán khai thác mẫu phổ biến trên các loại cơ sở dữ liệu (CSDL) khác nhau đã được quan tâm nghiên cứu. Trong đó có các CSDL là CSDL tĩnh, có nghĩa là CSDL cố định không có sự thay đổi hay thêm bớt các giao dịch trong quá trình khai thác và có cả CSDL động là CSDL tăng trưởng. Trên thực tế, CSDL giao dịch qua mỗi thời gian sẽ có sự thay đổi nhất định, và việc khai thác trên loại CSDL này (CSDL tăng trưởng) vẫn áp dụng phương pháp như trên CSDL tĩnh sẽ ít hiệu quả, tốn thời gian khai thác do đã duyệt lại CSDL đã duyệt ở những lần khai thác trước đó. Năm 2000, Hong và các cộng sự [5] đã đề xuất phương pháp khai thác mẫu phổ biến trên CSDL giao dịch tăng trưởng với sử dụng khái niệm pre-large itemset (tập gần phổ biến) để khai thác trên CSDL tăng trưởng. Tiếp sau đó có một số nghiên cứu đã cải tiến phương pháp này để đạt được hiệu quả cao hơn như [6], [7]. Trong đó, năm 2012 Ahmed [7] đề xuất phương pháp khai thác tập phổ biến trên CSDL định lượng (có trọng số) dựa trên tiếp cận FP-Growth. Tuy nhiên, giải pháp do Ahmed và cộng sự đề xuất chưa đạt được hiệu quả tốt nhất là trên các CSDL lớn, do phải đọc CSDL đến hai lần.

Trong bài báo này, chúng tôi đề xuất thuật toán khai thác mẫu phổ biến trên CSDL định lượng tăng trưởng dựa trên cây IT-Tree kết hợp khái niệm pre-large itemset [5], đồng thời trong biểu diễn tidset của các itemset chúng tôi sử dụng cấu trúc IWS [8] để nâng cao hiệu quả quả thuật toán.

Bài báo được trình bày trong 5 phần chính, tiếp theo mục II sẽ là phần các nghiên cứu liên quan; Thuật toán đề xuất sẽ được trình bày trong mục III; mục IV trình bày kết quả thực nghiệm trên 1 số CSDL mẫu; phần cuối cùng là Kết luận và Hướng nghiên cứu tiếp theo.

II. CÁC NGHIÊN CỨU LIÊN QUAN

2.1. Bài toán khai thác tập phổ biến có định lượng trên CSDL định lượng

Định nghĩa 1. Một CSDL trọng số (WD) được định nghĩa là 1 bộ ba $\langle T, I, W\rangle$, trong đó $T = \{t_1, t_2, .., t_m\}$ là tập các giao dịch, $I = \{i_1, i_2, .., i_n\}$ là tập các item, $W = \{w_1, w_2, \ldots, w_n\}$ là tập các trọng số của các item trong tập I.

Ví dụ 1: Cho CSDL trọng số như Bảng 1. Trong đó, tập các giao dịch $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ (Bảng 1A). Tập các item $I = \{A, B, C, D, E, F, G, H\}$. Tập trọng số của các item $W = \{0.8, 0.1, 0.5, 0.9, 0.2, 0.3, 0.4, 0.7\}$ (Bảng 1B).

Bảng 1. Ví dụ WD

A - Transaction Database	
ID	Items
1	*A, B, D, E*
2	*B, E, F*
3	*A, B, F*
4	*A, B, C, E*
5	*A, B, C, D, E*
6	*B, C, D, E, F*
7	*A, D*
8	*D, E, G*
9	*C, E, G*

B - Item Weight	
Item	Weight
A	0.8
B	0.1
C	0.5
D	0.9
E	0.2
F	0.3
G	0.4
H	0.7

Để khai thác FWI trên WD, F. Tao và các cộng sự [3] định nghĩa hai đại lượng là *ws* (weight support) và *tw* (transaction weight) như sau:

Định nghĩa 2. Trọng số giao dịch (*tw*) của một giao dịch t_k được định nghĩa như sau:

$$tw(t_k) = \frac{\sum_{i_j \in t_k} w_j}{|t_k|} \quad (1)$$

trong đó: - w_j là trọng số của item i_j;

- $|t_k|$ là số lượng item xuất hiện trong giao dịch t_k.

Ví dụ 2: $tw(t_2)$ của CSDL trong bảng 1 được tính theo công thức (1)

$$tw(t_2) = \frac{w(B) + W(E) + w(F)}{3} = \frac{0.1 + 0.2 + 0.3}{3} = 0.2$$

Tính toán tương tự, ta có *tw* của các giao dịch trong Ví dụ 1 được mô tả trong Bảng 2A.

Định nghĩa 3. Độ hỗ trợ trọng số (*ws*) của một itemset *X* được định nghĩa như sau:

$$ws(X) = \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)} \quad (2)$$

trong đó, - $t(X)$ là tập hợp các giao dịch có chứa itemset *X*.

Ví dụ 3: ws của itemset *AB* trong Ví dụ 1 được tính như sau:

$$ws(AB) = \frac{tw(t_1) + tw(t_3) + tw(t_4) + tw(t_5)}{\sum_{t_k \in T} tw(t_k)} = \frac{0.5 + 0.4 + 0.4 + 0.5}{4.12} = 0.44$$

Tính toán tương tự, ta có *ws* của các item trong Ví dụ 1 được thể hiện trong Bảng 2B.

Bảng 2. Trọng số của các giao dịch trong WD

A	
Transaction weights	
ID	*tw*
1	0.50
2	0.20
3	0.40
4	0.40
5	0.50
6	0.40
7	0.85
8	0.50
9	0.37
Sumtw	4.12

B	
Weighted support	
Item	WeightSupport *(ws)*
A	0.64
B	0.58
C	0.4
D	0.67
E	0.7
F	0.24
G	0.21
H	0

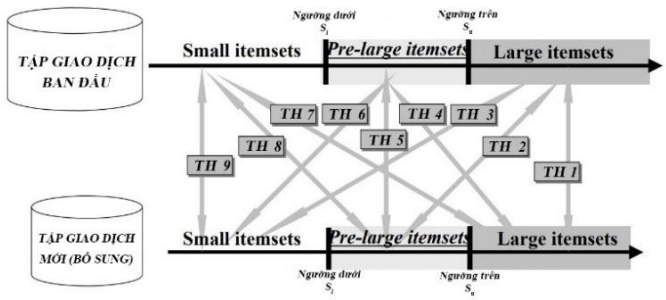
Để khai thác FWI, người ta sử dụng một ngưỡng gọi là *minws (minws ∈ [0, 1])*. Theo đó các itemset có *ws* thỏa mãn ngưỡng *minws* này được gọi là FWI theo ngưỡng *minws* đó. Bài toán khai thác FWI từ WD là bài toán tìm tất cả các FWI thỏa mãn ngưỡng *minws* cho trước.

2.2. Khai thác tập phổ biến trên Cơ sở dữ liệu định lượng tăng trưởng

Để khai thác tập phổ biến trên CSDL tăng trưởng, Hong và cộng sự [5] đã đề xuất khái niệm tập hợp cận phổ biến. Hong và cộng sự đã sử dụng hai ngưỡng được gọi là ngưỡng trên và ngưỡng dưới để xác các tập cận phổ biến và dựa trên chỉ số an toàn *f* của các giao dịch được chèn thêm để giảm nhu cầu quét lại CSDL ban đầu để duy trì hiệu quả các tập phổ biến. Công thức tính chỉ số an toàn *f* như sau:

$$f = \left\lceil \frac{(S_u - S_l)d}{1 - S_u} \right\rceil \quad (3)$$

trong đó: S_u là cận trên, S_l là cận dưới và *d* là số lượng giao dịch trong CSDL ban đầu và sẽ có 9 trường hợp như Hình 1 dưới đây:



Hình 1. Hình mô tả 09 trường hợp cận phổ biến

Bảng 3. 09 trường hợp và kết quả từng trường hợp

Trường hợp: CSDL Ban đầu - CSDL Mới	Kết quả
Trường hợp 1: Phổ biến - Phổ biến	Phổ biến (Thông thay đổi tập phổ biến)
Trường hợp 2: Phổ biến - Cận phổ biến	Phổ biến hoặc cận phổ biến, có thể xác định từ thông tin đã lưu
Trường hợp 3: Phổ biến - ít phổ biến	Phổ biến hoặc cận phổ biến hoặc ít phổ biến, có thể xác định từ thông tin đã lưu
Trường hợp 4: Cận phổ biến - Phổ biến	Cận phổ biến hoặc phổ biến, có thể xác định từ thông tin đã lưu
Trường hợp 5: Cận phổ biến - Cận phổ biến	Luôn cận phổ biến
Trường hợp 6: Cận phổ biến - Ít phổ biến	Cận phổ biến hoặc ít phổ biến, có thể xác định từ thông tin đã lưu
Trường hợp 7: Ít phổ biến - Phổ biến	Cận phổ biến hoặc ít phổ biến, khi số lượng giao dịch nhỏ
Trường hợp 8: Ít phổ biến - Cận phổ biến	Ít phổ biến hoặc cận phổ biến
Trường hợp 9: Ít phổ biến - Ít phổ biến	Luôn ít phổ biến

Từ Bảng 3, ta thấy các trường hợp 1, 5, 6, 8 và 9 sẽ không ảnh hưởng đến các tập phổ biến sau khi thêm giao dịch mới. Trường hợp 2 và 3 có thể làm bớt đi các tập phổ biến hiện có và trường hợp 4 và 7 có thể thêm các tập phổ biến mới. Nếu chúng ta giữ lại tất cả các tập phổ biến và cận phổ biến với độ hỗ trợ của chúng sau mỗi lần thay đổi, thì các trường hợp 2, 3 và 4 có thể được xử lý dễ dàng. Ngoài ra, trong giai đoạn cập nhật, tỷ lệ giữa số lượng giao dịch mới và số lượng giao dịch cũ thường rất nhỏ. Điều này rõ ràng hơn khi cơ sở dữ liệu phát triển lớn hơn. Về mặt lý thuyết, đã chỉ ra rằng một tập mục trong trường hợp 7 không thể là tập phổ biến cho toàn bộ cơ sở dữ liệu cập nhật khi số lượng giao dịch mới nhỏ hơn chỉ số an toàn *f*.

Công thức 3, Hong và cộng sự [5] đề xuất khi khai thác tập phổ biến trên CSDL nhị phân, trong đó *d* là số lượng giao dịch của CSDL, đại lượng *d* này đóng vai trò chi phối trong việc xác định *support* của các itemset, tương đương với đại lượng *d* này trên CSDL định lượng là *sumtw* - tổng trọng số tất cả các giao dịch, trong đó trọng số mỗi giao dịch được xác định theo công thức 1. Đối với CSDL định lượng đang xét, chúng tôi đề xuất sử dụng công thức 4, trong đó, đại lượng *d* trong công thức 3 được thay thế bằng *sumtw*

$$f = \left\lceil \frac{(S_u - S_l)sumtw}{1 - S_u} \right\rceil \quad (4)$$

Ví dụ 4: Giả sử *sumtw* = 4.12, S_l = 30 %, S_u = 40 % và một tập itemset *I* thuộc tập *Tr* (không thuộc tập phổ biến hay cận phổ biến) trong cơ sở dữ liệu ban đầu. Tổng trọng số tất cả giao dịch mới sẽ không làm thay đổi tập phổ biến trong đó cơ sở dữ liệu ban đầu được xác định bởi ngưỡng an toàn *f* dưới đây:

$$f = \left\lceil \frac{(S_u - S_l)sumtw}{1 - S_u} \right\rceil = \frac{(0.4 - 0.3)4.12}{1 - 0.4} = 0.69$$

Do đó, nếu các giao dịch mới được chèn có tổng trọng số giao dịch bằng hoặc nhỏ hơn 0,6, thì sẽ không thể làm thay đổi tập phổ biến cho toàn bộ cơ sở dữ liệu cập nhật. Từ số an toàn *f*, số lượng giao dịch mới cần thiết để xử lý hiệu quả trường hợp 7 được xác định bởi S_u, S_l và *sumtw*. Nếu *sumtw* lớn hơn, thì *t* cũng có thể lớn hơn. Do đó, khi cơ sở dữ liệu phát triển, cách tiếp cận này ngày càng trở nên hiệu quả. Đặc tính này đặc biệt hữu ích cho các ứng dụng trong thực tế.

Bài toán khai thác tập phổ biến trên CSDL tăng trưởng được quan tâm từ khá sớm, ban đầu là các nghiên cứu trên CSLD nhị phân [3], [6], các nghiên cứu này sử dụng cấu trúc IT-Tree theo thuật toán Eclat. Năm 2012 nhóm Ahmed [7] đã đề xuất phương pháp khai thác tập phổ biến trên CSDL định lượng (CSDL có trọng số) theo tiếp cận FP-Growth [2]. Tuy nhiên tiếp cận này quét CSDL hai lần nên chưa hiệu quả về mặt thời gian nhất là đối với các CSDL lớn. Cũng theo tiếp cận này, một số nghiên cứu khai thác trên CSDL định lượng tăng trưởng như [9] khai thác tập phổ biến tối đại; hay [10] khai thác tập phổ biến trên trên cơ sở dữ liệu động (thêm xoá, thay đổi giao dịch).

III. THUẬT TOÁN ĐỀ XUẤT

3.1. Ký hiệu

Các ký hiệu được sử dụng trong thuật toán Pre-FWIs được đề xuất được đưa ra dưới đây:

D	Tập giao dịch ban đầu (CSLD Ban đầu)
T	Tập giao dịch mới bổ sung
U	Tập giao dịch sau khi đã cập nhật, $D \cup T$
$sumtw_d$	Tổng trọng số tất cả giao dịch trong tập ban đầu *D*
$sumtw_t$	Tổng trọng số tất cả giao dịch trong tập mới *T*
$sumtw_c$	Tổng trọng số các giao dịch đã được thêm vào *D* trước khi *D* được duyệt lại toàn bộ
S_l	Chỉ số cận dưới của tập cận phổ biến
S_u	Chỉ số cận trên của tập cận phổ biến $S_u > S_l$
X	Đại diện cho 1 phần tử
Tr	Cây IT-tree lưu toàn bộ tập cận phổ biến và tập phổ biến của tập giao dịch ban đầu *D*
T(X)	Tập chỉ số giao dịch tidset của phần tử *X* trong Tập giao dịch mới *T*
I_T	Tập itemset một phần tử trong tập giao dịch mới *T*

$ws^{Tr}(X)$	Độ hỗ trợ trọng số của phần tử *X* trong tập *Tr*
[*P*]	Lớp tương đương của nút P trong tập *Tr*
[∅]	Lớp tương đương với nút gốc trong tập *Tr*, đại diện là nút rỗng ∅

3.2. Thuật toán Pre-FWIs

Ý tưởng chính của thuật toán được đề xuất là duyệt theo chiều sâu cây ITW-tree (là một mở rộng của IT-tree[2], với việc mỗi nút thêm một đại lượng là *ws* của itemset đó) để cập nhật độ hỗ trợ và các tập mục tidset được lưu trữ trong cây ITW-tree, trong đó sử dụng cấu trúc IWS, [8] để lưu trữ tidset nhằm tăng tốc tính toán trong quá trình tính giao tidset của các itemset. Thuật toán được đề xuất chỉ xử lý các giao dịch được chèn mới để cập nhật độ hỗ trợ sau cùng của các tập itemset bằng cách sử dụng kỹ thuật duyệt chiều sâu. Sự gia tăng về độ hỗ trợ của các tập itemset được tính toán nhanh chóng bằng cách sử dụng phép giao của các tập IWS. Thuật toán cũng sử dụng thuộc tính bao đóng giảm (downward-close) [1] để loại bớt các tập hợp itemset không thoả ngưỡng trong khi duyệt qua ITW-tree. Do đó, các tập phổ biến có thể được lấy trực tiếp từ quá trình duyệt cây. Các tập itemset có độ hỗ trợ nhỏ trong cơ sở dữ liệu ban đầu nhưng cận phổ biến hoặc phổ biến trong tập các giao dịch mới sẽ được chèn vào các tập hợp itemset được quét lại. Khi số lượng giao dịch mới được chèn vào vượt quá mức an toàn *f*, thuật toán phải quét lại cơ sở dữ liệu ban đầu để xử lý. Các chi tiết của thuật toán như sau.

INPUT: Ngưỡng hỗ trợ dưới S_l, ngưỡng trên S_u, cây IT-tree *Tr* lưu trữ các tập phổ biến và cận phổ biến bắt nguồn từ cơ sở dữ liệu gốc *D* bao gồm $(sumtw_d + sumtw_c)$ giao dịch ($sumtw_d$ là tổng trọng số tất cả các giao dịch trong cơ sở dữ liệu cuối cùng quét lại và $sumtw_c$ là tổng trọng số tất cả các giao dịch được chèn từ lần quét lại cuối cùng đến nay) và một tập hợp *t* giao dịch mới.

OUTPUT: Cây IT-tree *Tr* lưu trữ các tập phổ biến và cận phổ biến và các tập itemset được quét lại từ *U*.

```
BEGIN PROCEDURE
1.  f = ⌈((S_u − S_l) sumtw_d) / (1 − S_u)⌉ // theo công thức 4
2.  for each item X ∈ I_T { // I_T là tập itemset 1 phần tử từ tập T
3.    if (X ∉ Tr){ //Trường hợp 7, 8, and 9 trong Bảng 4
4.      if { Σ_{t_k ∈ T(X)} tw(t_k) / sumtw_t ≥ S_l
5.        Thêm X vào cây Tr
6.        wsup^Tr(X) = Σ_{t_k ∈ T(X)} tw(t_k) / (sumtw_d + sumtw_c + sumtw_t)
7.      if (wsup^Tr(X) ≤ S_u) {đánh dấu X (đánh dấu vào tập duyệt lại)} //end if
8.      } //end if
9.  }//end for
10. for each item X ∈ Tr / (X đã đánh dấu) { //Trường hợp 1, 2, 3, 4, 5, và 6 trong Bảng 4
11.   if (X ∈ I_T) {
12.     if ( (wsup^Tr(X) * (sumtw_d + sumtw_c) + Σ_{t_k ∈ T(X)} tw(t_k)) / (sumtw_d + sumtw_c + sumtw_t) ≥ S_l )
13.       wsup^Tr(X) = (sup^Tr(X) * (sumtw_d + sumtw_c) + Σ_{t_k ∈ T(X)} tw(t_k)) / (sumtw_d + sumtw_c + sumtw_t)
14.     else loại bỏ X khỏi Tr
15.   }else if ( (wsup^Tr(X) * (sumtw_d + sumtw_c)) / (sumtw_d + sumtw_c + sumtw_t) ≥ S_l ) {
16.       wsup^Tr(X) = (sup^Tr(X) * (sumtw_d + sumtw_c)) / (sumtw_d + sumtw_c + sumtw_t)
17.     } else loại bỏ X khỏi Tr
18. }//end for
19. ENUMERATE_FWIs (Tr, [∅]) // thủ tục đệ quy để cập nhật hỗ trợ và
                              // tidsets của các itemset dựa trên độ sâu của thứ tự đầu tiên duyệt
                              // và các tập tidset của các item trong các giao dịch thêm mới
20. if (sumtw_t + sumtw_c > f) {
21.   sumtw_d = sumtw_d + sumtw_t + sumtw_c
22.   sumtw_c = 0
23.   duyệt lại tập itemset được đánh dấu trong Tr với giao dịch U để chuẩn hóa tập Tr}
24. else sumtw_c = sumtw_c + sumtw_t
END PROCEDURE
PROCEDURE ENUMERATE_FWIs (Tr, [P])
BEGIN PROCEDURE
25. for each itemset X_i ∈ [P] and X_i không được đánh dấu{
```

26. **for each** itemset $X_j \in [P]$ with $j > i$ and X_j không được đánh dấu {
27. $Z = X_i \cup X_j$
28. $T(Z) = T(X_i) \cap T(X_j)$
29. **if** (Z có trong Tr)
30. **if** $\left(\frac{wsup^{Tr}(Z)*(sumtw_d + sumtw_c) + \sum_{t_k \in T(Z)} tw(t_k)}{sumtw_d + sumtw_c + sumtw_t} \geq S_l\right)$
31. $wsup^{Tr}(Z) = \frac{sup^{Tr}(Z)*(sumtw_d + sumtw_c) + \sum_{t_k \in T(Z)} tw(t_k)}{sumtw_d + sumtw_c + sumtw_t}$
32. $[Pi] = [Pi] \cup \{Z \text{ x } T(Z)\}$
33. **else** loại bỏ Z khỏi Tr
34. **else if** $\left(\frac{\sum_{t_k \in T(Z)} tw(t_k)}{sumtw_d + sumtw_c + sumtw_t} \geq S_l\right)$ {
35. Thêm Z vào Tr và đánh dấu Z vào tập rescanned itemset
36. $wsup^{Tr}(Z) = \frac{+ \sum_{t_k \in T(Z)} tw(t_k)}{sumtw_d + sumtw_c + sumtw_t}$ }
37. }// end for
38. ENUMERATE_FWIs(Tr, $[Pi]$)
39. }//end for

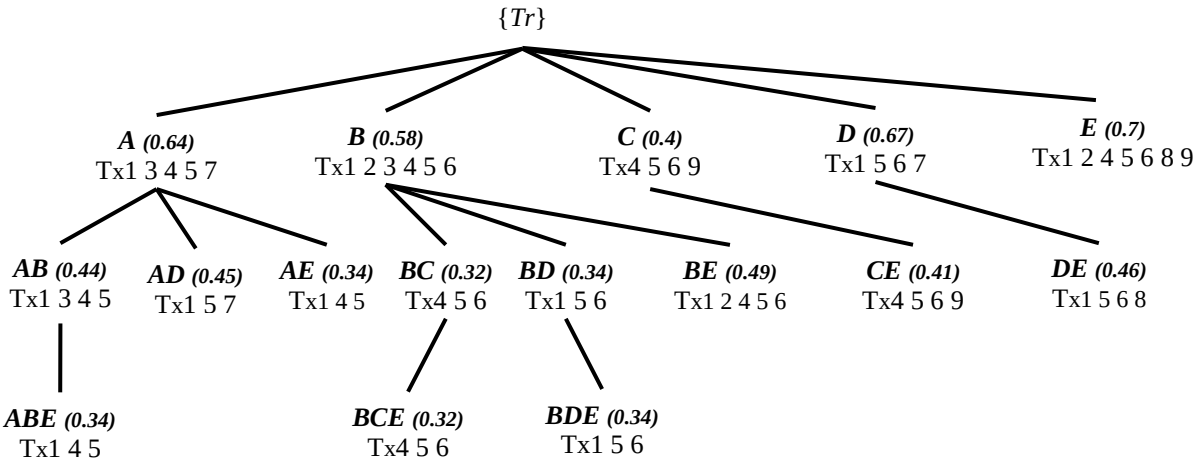
END PROCEDURE

Hình 2. Thuật toán khai thác tập phổ biến trên CSDL định lượng tăng trưởng

Thuật toán trong Hình 2 được đề xuất để xử lý cho các giao dịch được chèn tuần tự. Dòng 2 đến dòng 7 xử lý tập 1- itemset có trong tập giao dịch mới I_T nhưng thuộc tập có độ support nhỏ trong cơ sở dữ liệu ban đầu (Trường hợp 7, 8 và 9 trong Bảng 4) và dòng 8 đến dòng 16 xử lý tập itemset 1 phần tử có trong tập Tr trong cơ sở dữ liệu ban đầu và xuất hiện lại trong tập giao dịch mới I_T (Trường hợp 1, 2, 3, 4, 5 và 6 trong Bảng 3). Sau dòng 16, độ hỗ trợ và tập tidset của các itemset 1 phần tử trong cây Tr đã được cập nhật và các tập itemset chú ý cần duyệt lại đã được đánh dấu. Do đó, dòng 18 đến 22 kiểm tra xem số tổng trọng số tất cả giao dịch mới được chèn có vượt quá mức an toàn f hay không, nếu đúng thì thuật toán sẽ quét lại cơ sở dữ liệu gốc để xử lý các tập itemset đã đánh dấu.

3.3. Ví dụ minh họa

Trong phần này, một ví dụ được đưa ra để minh họa thuật toán Pre-FWIs. Giả sử rằng cơ sở dữ liệu D ban đầu bao gồm 9 giao dịch trong Bảng 1. Đối với S_l = 30 % và S_u = 40 %, cây IT-tree Tr được xây dựng từ D được thể hiện trong Hình 3. Trên thực tế, thông tin tidset của các tập itemset được khai thác từ cơ sở dữ liệu gốc không được giữ lại vì thuật toán này xử lý khi quá trình cập nhật thêm giao dịch mới.



Hình 3. Cây IT-tree Tr được xây dựng từ cơ sở dữ liệu (Bảng 1) với $sumtw$ = 4.12; S_l = 0.3; S_u = 0.4

Giả sử hai giao dịch mới trong Bảng 4 được chèn vào sau khi cây Tr được xây dựng và dạng dọc của hai giao dịch mới.

Bảng 4. Trọng số của các giao dịch trong WD

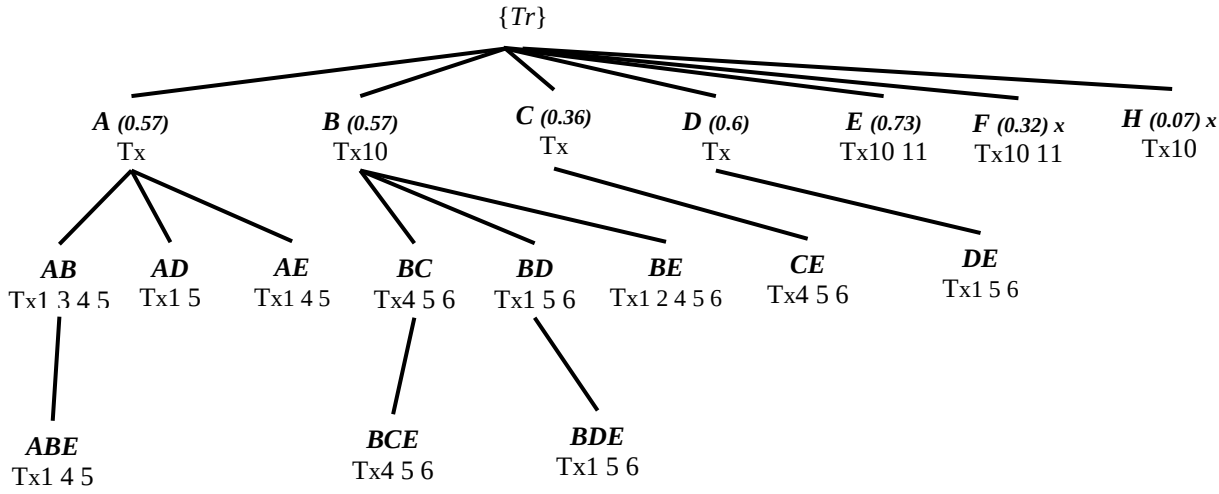
Transaction Database (T)	
ID	Item
10	{B, E, F, H}
11	{E, F}
$Sumtw_t$	0.575

Định dạng dọc	
Item	Transaction
B	10
E	10 11
F	10 11

Đối với dữ liệu trên, thuật toán đề xuất Pre-FWIs tiến hành như sau. Biến $sumtw_c$ ban đầu có giá trị 0, giá trị f được tính theo công tức trong thuật toán:

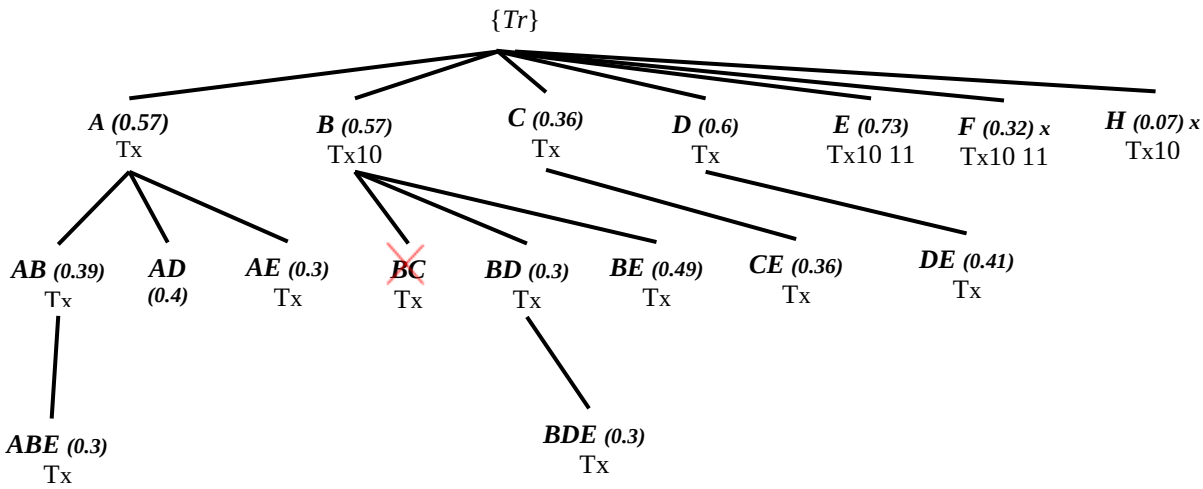
$$f = \left\lceil \frac{(S_u - S_l)sumtw_d}{1 - S_u} \right\rceil = \frac{(0.4 - 0.3)4.12}{1 - 0.4} = 0.69$$

Vì $sumtw_t + sumtw_c = 0.575 < f$, việc quét lại cơ sở dữ liệu gốc là không cần thiết. Sau đó cập nhật độ hỗ trợ và tập tidset của các 1- itemset trong *Tr* dựa trên tidset của các 1-itemset trong hai giao dịch mới và kết quả được thể hiện trong Hình 4.



Hình 4. Cây kết quả trước khi đệ quy

Và sau khi thủ tục đệ quy của thuật toán kết thúc thì kết quả cho ra của thuật toán là cây *Tr* lưu trữ các tập phổ biến (với ngưỡng từ S_u trở lên) và tập cận phổ biến (với ngưỡng S_l) cho toàn bộ cơ sở dữ liệu cập nhật, cũng như các tập hợp itemset được đánh dấu khi cần quét lại. Cây *Tr* sau đó có thể được sử dụng để xử lý các giao dịch mới được chèn tiếp theo sau đó.



Hình 5. Cây kết quả khi kết thúc thuật toán Pre-FWIs

Kết quả cây *Tr* sau quá trình cập nhật của thuật toán Pre-FWIs như Hình 5, mẫu phổ biến khai thác được là các itemset có độ hỗ trợ lớn hơn S_u: Cụ thể là 07 tập phổ biến như sau: {{*A*: 0.57}, {*B*: 0.57}, {*D*: 0.6}, {*E*: 0.73}, {*AD*: 0.4}, {*BE*: 0.49}, {*DE*: 0.41}}.

IV. KẾT QUẢ THỰC NGHIỆM

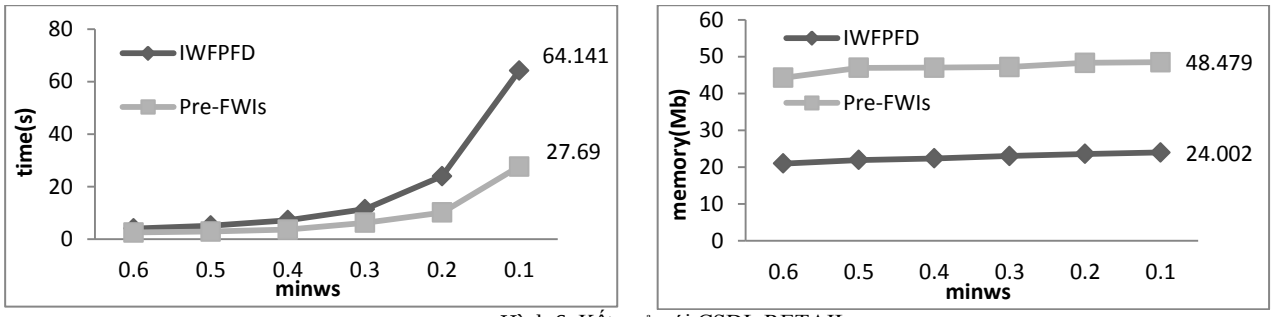
Tất cả các thuật toán trong thử nghiệm đều được thực hiện bằng Java. Cấu hình PC bao gồm CPU Intel Core i7-8550U 1.8 GHz, RAM 16GB và HĐH Windows 10. Bốn cơ sở dữ liệu từ http://fimi.cs.helsinki.fi/data được mô tả trong Bảng 5.

Bảng 5. Mô tả CSDL thực nghiệm

STT	CSDL	Số lượng mục	Số lượng giao dịch	Ghi chú
1.	RETAIL	16.470	88.162	Thêm trọng số
2.	BMS-POS	1.657	515.597	Thêm trọng số
3.	CONNECT	468	340.183	Thêm trọng số
4.	ACCIDENTS	130	67.557	Thêm trọng số

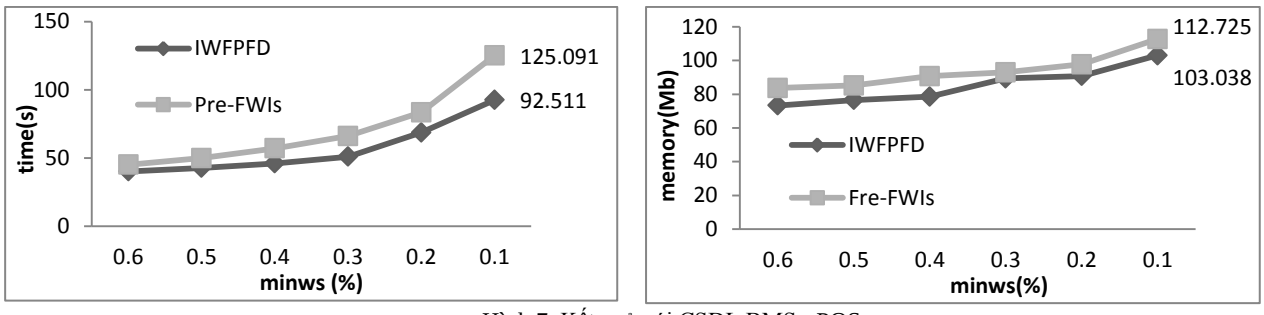
Các thử nghiệm được thực hiện để so sánh thời gian thực hiện khai thác tất cả các tập phổ biến bằng thuật toán thuật toán IWFPFD [8] và thuật toán Pre-FWIs được đề xuất. Với mỗi CSDL trên, chúng tôi tiến hành thực nghiệm bằng cách lấy cận trên bằng đúng *minws* và cận dưới S_l = 0.7**minws*, từ đó tính hàm f theo công thức 4 để xác định có duyệt lại CSDL hay không. Với mỗi CSDL thực nghiệm chúng tôi lấy ra 1000 giao dịch chia thành năm nhóm (mỗi nhóm 200 giao dịch), mỗi lần thực nghiệp chúng tôi thêm một nhóm vào CSDL để chạy lấy số liệu và chia trung bình thời gian chạy, như vậy mỗi CSDL và một ngưỡng *minws* chúng tôi chạy năm lần mỗi lần thêm 200 giao dịch.

Các kết quả thực nghiệm được mô tả qua các biểu đồ dưới đây.



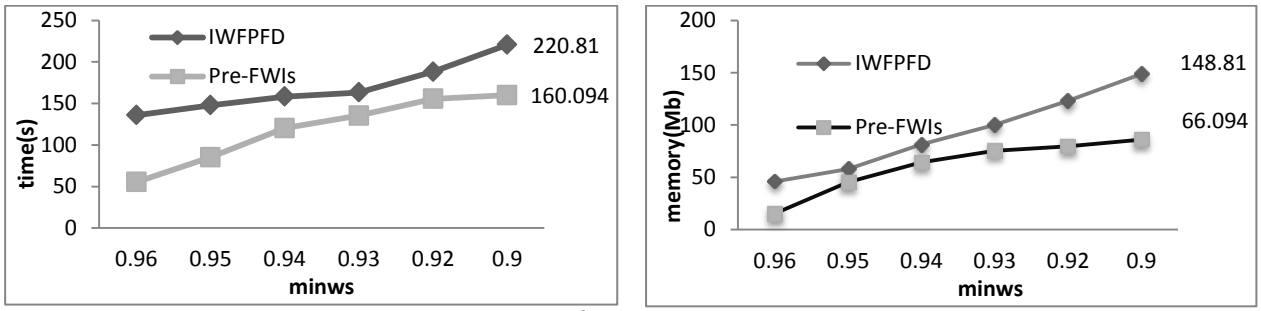
Hình 6. Kết quả với CSDL RETAIL

Kết quả thực nghiệm trong Hình 6 cho thấy thuật toán IWFPFD theo tiếp cận FP-Growth trong [7] có kết quả tốt hơn cả thời gian lẫn bộ nhớ. Ở CSDL này Pre-FWIs chậm hơn hẳn, thời gian và bộ nhớ đều hơn gấp đôi so với IWFPFD.



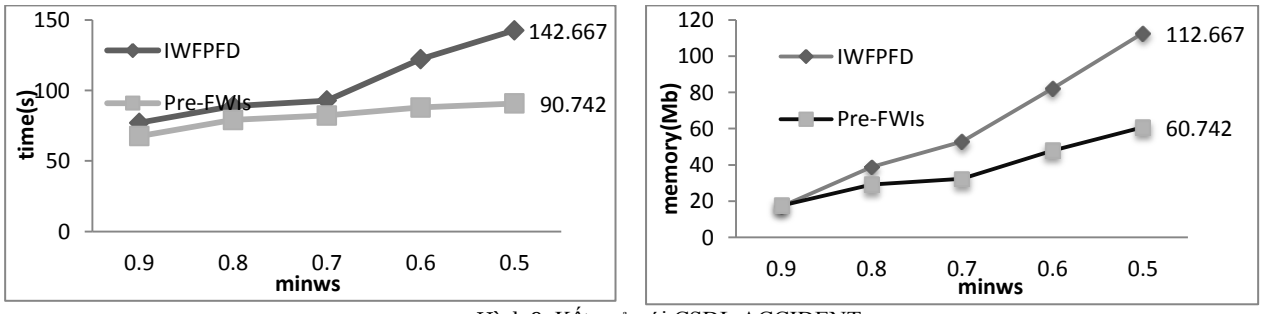
Hình 7. Kết quả với CSDL BMS - POS

Kết quả thực nghiệm trong Hình 7 cho thấy kết quả khả quan hơn của Pre-FWIs so với IWFPFD. Mặc dù cả thời gian lẫn bộ nhớ đều cao hơn IWFPFD nhưng cơ bản sự chênh lệch không quá nhiều như so với kết quả trên CSDL RETAIL.



Hình 8. Kết quả với CSDL CONNECT

Trong Hình 8, kết quả thực nghiệm trên CSDL CONNECT cho thấy Pre-FWIs có kết quả tốt hơn hẳn so với IWFPFD, nhất là về thời gian thì Pre-FWIs luôn thấp hơn một nửa so với IWFPFD.



Hình 9. Kết quả với CSDL ACCIDENT

Với CSDL ACCIDENT, kết quả thực nghiệm trong Hình 9 đã cho thấy sự vượt trội của Pre-FWIs so với IWFPFD. Thậm chí với ngưỡng *minws* =0,5, thời gian khai thác của IWFPFD nhiều gấp 1,85 lần so với Pre-FWIs.

Các kết quả thực nghiệm trên cho thấy với CSDL thưa (như RETAIL) Pre-FWIs chạy chậm hơn hẳn và cũng sử dụng nhiều bộ nhớ hơn so với IWFPFD. Sang CSDL BMS-POS tương đối dày thì hai thuật toán này cho kết quả tương đương nhau, IWFPFD dù vẫn có kết quả tốt hơn cả thời gian lẫn bộ nhớ nhưng sự chệnh lệch không nhiều. Với các CSDL dày như CONNECT và ACCIDENT thì khác hẳn, Pre-FWIs nhanh hơn hẳn so với IWFPFD nhất là đối với CSLD ACCIDENT.

Từ các kết quả thực nghiệm trên có thể thấy Pre-FWIs cho hiệu quả rất tốt về mặt thời gian và bộ nhớ đối với các CSDL dày và ít có hiệu quả, hạn chế đối với các loại CSDL thưa.

V. KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

Bài báo này đề xuất một cách tiếp cận mới dựa trên cấu trúc cây IT-tree để khai thác tập phổ biến trên CSDL định lượng tăng trưởng với việc sử dụng khái niệm "tập cận phổ biến" và sử dụng cấu trúc đoạn bit vector IWS [8] trong biểu diễn tidset của các itemset nhằm tăng tốc thuật toán. Các kết quả thử nghiệm cho thấy tính hiệu quả về mặt thời gian và bộ nhớ của phương pháp đề xuất trên các CSDL thực nghiệm. Trong thời gian tới chúng tôi sẽ tiếp tục theo đuổi chủ đề này trên các loại CSDL định lượng khác, cũng như hướng tới các giải pháp song song cho bài toán này trên các CSDL lớn.

TÀI LIỆU THAM KHẢO

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases", In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp.487-499, 1994.

[2] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules", In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97), AAAI Press, pp.283-286, 1997.

[3] F. Tao, F. Murtagh, and M. Farid, "Weighted Association Rule Mining Using Weighted Support and Significance Framework", KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. pp.661-666, 2003.

[4] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation", In SIGMOD'00 Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1-12, 2000.

[5] T. P. Hong, C. Y. Wang, Y. H. Tao, "A new incremental data mining algorithm using pre-large itemsets", Intelligent Data Analysis, 5(2), pp.111-129, 2001.

[6] T. P. Le, T. P. Hong, B. Vo, B. Le, "An efficient incremental mining approach based on IT-tree", In RIVF'12 Proceedings of the 2012 IEEE International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future, pp.57-61, 2012.

[7] C. F. Ahmed, S. K. Tanbeer, B. Jeong, Y. Lee, H. Choi, "Single-pass incremental and interactive mining for weighted frequent patterns", Expert Systems with Applications, Volume 39, Issue 9, Pages 7976-7994, ISSN 0957-4174, 2012.

[8] H. Nguyen, B. Vo, M. Nguyen, "An efficient algorithm for mining frequent weighted itemsets using interval word segments", Appl Intell 45, pp.1008-1020, 2016.

[9] U. Yun, G. Lee, "Incremental mining of weighted maximal frequent itemsets from dynamic databases", Expert Systems with Applications, Volume 54, pp.304-327, ISSN 0957-4174, 2016.

[10] H. Nam, U. Yun, E. Yoon, J. Lin, "Efficient approach for incremental weighted erasable pattern mining with list structure", Expert Systems with Applications, Volume 143, ISSN 0957-4174, 2020.

AN EFFICIENT ALGORITHM FOR MINING FREQUENT WEIGHTED ITEMSETS FROM INCREMENTAL DATABASES

Vuong Dinh Bac, Nguyen Duy Ham

ABSTRACT: *Mining pattern from transactional databases is one of the most important problems in data mining. In particular, the mining pattern on the incremental database is a problem that appears many applications in practice, because the database is updated and added with new transactions. Recently, many research related to this problem on both of binary databases and quantitative databases. However, the solutions proposed on the current incremental database still take a lot of time to mine. In this paper we propose an efficiency algorithm on a incremental quantitative database based on ITW-Tree. The algorithm we propose has a better performance than the methods already available on the experiment databases.*

Keywords: *pattern, experiment database, quantitative database, incremental quantitative database.*