Tạp chí Tin học và Điều khiển học, T.20, S.1 (2004), 80–90

THUẬT TOÁN KHAI THÁC DỮ LIỆU TĂNG TRƯỞNG TRONG CƠ SỞ DỮ LIỆU CÓ TÍNH THỜI GIAN

NGUYỄN ĐÌNH THUÂN

Đại học Thủy sản Nha Trang

Abstract. Discovering association rules is a well-established problem in the field of data mining. In later years, many researchers have focused on the mining rules from temporal data ([3, 8]). In this paper, we propose an algorithm for incremental mining of association rules, by partitioning a transaction database into several partitions. The algorithm employs a filtering thresholds in each partition and outputs a cumulative filter, which consists of the progressive candidate set of itemsets.

Tóm tắt. Khám phá các luật kết hợp là bài toán cơ bản được đặt ra trong lĩnh vực khai thác dữ liệu. Trong thời gian gần đây, một số tác giả đã nghiên cứu về khai thác các luật với dữ liệu có tính thời gian ([3, 8]). Trong bài báo này, chúng tôi đề xuất thuật toán khai thác tăng trưởng các luật kết hợp bằng cách chia dữ liệu thành các phần, tính các ngưỡng cho mỗi phần dữ liệu và các ngưỡng tích luỹ, để tìm ra các tập dữ liệu thoả mãn.

1. GIỚI THIỆU

Khai thác các luật kết hợp là một trong các thuật toán ứng dụng nhiều trong việc khám phá tri thức trong cơ sở dữ liệu hoặc khai thác dữ liệu. Các luật này được ví dụ như: 90% khách hàng đã mua bơ và bánh mì thì sẽ mua sữa, hoặc 85% sinh viên ngành tin học đạt điểm giỏi môn toán cao cấp và lập trình căn bản thì sẽ đạt điểm giỏi môn cấu trúc dữ liệu.

Thuật toán về khai thác dựa trên các luật kết hợp lần đầu tiên được giới thiệu trong [2] và bao gồm hai bước chính như sau:

+ Tìm các tập đơn vị dữ liệu xảy ra thường xuyên (nghĩa là tìm tất cả các tập đơn vị dữ liệu có số lần xuất hiện lớn hơn một ngưỡng cho trước).

+ Từ các tập đơn vị dữ liệu xảy ra thường xuyên, thiết lập các luật thể hiện mối liên hệ giữa các đơn vị dữ liệu.

Nhận xét:

- Khi dữ liệu được tăng trưởng theo thời gian thì sau một đơn vị thời gian như một tuần, một tháng... thì kho dữ liệu sẽ được lưu trữ tăng lên rất nhiều, muốn đưa ra các luật thì phải kết hợp các dữ liệu mới thu thập được với các dữ liệu đã có trong quá trình khai thác dữ liệu và xoá bỏ đi các dữ liệu không còn tác dụng.

- Bên cạnh đó với các ứng dụng trong khai thác dữ liệu thì vấn đề thời gian tìm kiếm các luật kết hợp giữa các đơn vị dữ liệu là vấn đề quan trọng, bởi vì đối với các tập dữ liệu lớn theo thời gian càng được bổ sung, nếu không có thuật toán hữu hiệu thì chi phí cho thời gian thực hiện thuật toán là rất lớn.

Trong Mục 3 của bài báo sẽ giới thiệu thuật toán cải tiến từ thuật toán tựa Apriori trên cơ sở yếu tố thời gian. CSDL được chia thành các phần theo thời gian (mỗi tuần, tháng...) tuỳ vấn đề giải quyết. Bên cạnh đó ta xác định một chu kỳ (một quí, một năm...) cần duyệt CSDL tìm ra các tập luật. Khi thêm hoặc bớt các phần dữ liệu, thuật toán tìm các luật kết hợp bằng cách dựa vào các phần dữ liệu đã có trước đó và thêm các thông tin mới cập nhật.

2. CÁCH TIẾP CẬN CỦA CÁC THUẬT TOÁN TỰA-APRIORI

2.1. Các định nghĩa ([3, 8])

Cho $I = \{I_1, I_2...I_m\}$ là tập các đơn vị dữ liệu. Cho D là tập các giao tác, mỗi giao tác T là tập các đơn vị dữ liệu sao cho $T \subseteq I$.

Định nghĩa 1. Ta gọi giao tác T chứa X với X là tập các đơn vị dữ liệu của I nếu $X \subseteq T$.

Định nghĩa 2. Một luật kết hợp là ký hiệu kéo theo có dạng $X \to Y$, trong đó $X \subset I$, $Y \subset I$ và $X \cap Y = \varnothing$.

Định nghĩa 3. Gọi luật $X \to Y$ có độ tin cậy c (confidence) trên tập giao tác D với $0 \leq c \leq 1$, nếu tỉ lệ c các giao tác trong D mà chứa X thì cũng chứa Y, ký hiệu $Conf(X \to Y) = c$.

Định nghĩa 4. Ta gọi luật $X \to Y$ có mức hỗ trợ (support) là s trong tập giao tác D nếu tỷ lệ giao tác trong D chứa $X \cup Y$ là s, ký hiệu $Supp(X \to Y) = s$.

Nhận xét. Các mức hỗ trợ và độ tin cậy chính là các xác suất sau:

$$Supp(X \to Y) = P(X \cup Y): \quad \text{Xác suất của } X \cup Y \text{ trong } D,$$

$$Conf(X \to Y) = P(Y/X): \quad \text{Xác suất có điều kiện.}$$

Định nghĩa 5. Cho trước $\min\_Supp = s_0$ và $\min\_Conf = c_0$. Ta gọi luật $X \to Y$ là xảy ra thường xuyên nếu $Supp(X \to Y) > s_0$ và $Conf(X \to Y) > c_0$.

Ví dụ 1. Xét CSDL D gồm $N = 12$ giao tác $t_1, t_2...t_{12}$ như sau:

Ngày	*T_ ID*	*Các đơn vị dữ liệu*					
D_1	t_1	A			D	E	
	t_2	A					F
	t_3	A	B		D	E	
D_2	t_4	A	B	C		E	
	t_5				D		F
	t_6	A		C	D	E	
D_3	t_7		B		D	E	
	t_8	A			D		F
	t_9		B	C		E	
D_3	t_{10}		B	C		E	F
	t_{11}		B	C	D		F
	t_{12}	A			D		

Ta có: $A \to D$ với độ tin cậy $c = 5/7 = 71.42\%$ và mức hỗ trợ $s = 5/12 = 41.66\%$.

$A \to E$ với độ tin cậy $c = 4/7 = 57.14\%$ và mức hỗ trợ $s = 4/12 = 33.33\%$.

$C \to B$ với độ tin cậy $c = 4/5 = 80\%$ và mức hỗ trợ $s = 4/12 = 33.33\%$.

Hai bước chính của bài toán khai thác dữ liệu dựa trên các luật kết hợp ([2]):

- Tìm tất cả các đơn vị dữ liệu có mức hỗ trợ lớn hơn ngưỡng min_$Supp$ cho trước, nghĩa là số lần xuất hiện lớn hơn một ngưỡng. Tập đơn vị dữ liệu là tập xảy ra thường xuyên nếu có mức hỗ trợ lớn hơn hoặc bằng min_$Supp$.

- Dựa trên các tập thường xuyên này để tìm ra các luật thỏa mãn bài toán. Nếu XY và X là các tập thường xuyên xảy ra, ta tính được độ tin cậy của luật này:

$$Conf(X \to Y) = Support(XY)/support(X)$$

Nếu $Conf(X \to Y) \geq$ min_$conf$ thì $X \to Y$ là một luật cần tìm (luật này chắc chắn lớn hơn hoặc bằng min_$Supp$ vì XY là tập thường xuyên xảy ra). Như vậy, bài toán khai thác các luật kết hợp được đưa về hai bài toán sau:

(1) Tạo ra tất cả tập đơn vị dữ liệu thường xuyên xảy ra (thỏa ngưỡng là min_$Supp$).

(2) Từ tập các đơn vị dữ liệu xảy ra thường xuyên $Y = \{I_1 I_2 ... I_k\}$ với $k \geq 2$, tìm các luật bằng cách dựa vào các tập con của mỗi tập đơn vị dữ liệu và tính các độ tin cậy của chúng như trên.

Trong hai bài toán trên thì giải pháp cho bài toán (2) rõ ràng hơn, còn bài toán (1) sẽ được giải quyết ở Mục 3.

2.2. Thuật toán Tựa - Apriori ([2, 4])

Cách tiếp cận này dựa trên Heuristic sau: Nếu bất kỳ tập k đơn vị dữ liệu nào là không xảy ra thường xuyên thì tập $(k+1)$ đơn vị dữ liệu bất kỳ chứa chúng cũng sẽ không xảy ra thường xuyên.

Tìm các tập thường xuyên:

(1) Trong lần duyệt đầu tiên, tính mức hỗ trợ của từng đơn vị dữ liệu riêng và xác định đơn vị dữ liệu nào là thường xuyên (tương ứng với min_$Supp$ cho trước).

(2) Trong các lần sau đó, ta sẽ bắt đầu với tập thường xuyên đã xét lần trước, tạo ra các tập đơn vị dữ liệu ứng viên và từ các tập ứng viên này, ta tính được mức hỗ trợ của chúng.

(3) Cuối của phép duyệt ta xác định được tập ứng viên nào là tập thường xuyên thật sự và chúng trở thành nhân trong phép duyệt tiếp theo.

(4) Quá trình trên được tiếp tục cho đến khi không còn tập thường xuyên nào được tìm.

Ký hiệu:

- Ta gọi số đơn vị dữ liệu trong một tập là kích thước của chúng và tập có k phần tử là k đơn vị dữ liệu.

- Gọi L_k là tập hợp các tập thường xuyên k đơn vị dữ liệu. Thông tin về mỗi phần tử của L_k có dạng (X, p) trong đó X là tập con của I và p là số lần xuất hiện của X trong D.

- C_k là tập hợp các tập ứng viên k đơn vị dữ liệu. Thông tin về mỗi phần tử của C_k cũng có dạng (X, p) như trên.

Trong thuật toán này, với sử dụng Heuristic trên, một tập ứng viên k đơn vị dữ liệu $c \in C_k$ sẽ thỏa mãn L_{k-1} chứa toàn bộ các tập con gồm $(k-1)$ đơn vị dữ liệu của chúng. Ngoài ra, tập ứng viên là tập xảy ra thường xuyên nếu thỏa mãn min_$Supp$.

Thuật toán 2.1. Tạo ra các tập thường xuyên

Input. Tập đơn vị dữ liệu I

Tập giao tác D gồm N giao tác

$\min\_Supp$

Output. Tập L gồm các phần tử là k đơn vị dữ liệu xảy ra thường xuyên

Begin

$L_1 = \{$tập thường xuyên 1- đơn vị dữ liệu$\}$;

for $(k = 2; L_{k-1} \neq \varnothing; k++)$ do

begin

C_k = tạo ra tập ứng viên từ (L_{k-1});

for mỗi giao tác $t \in D$ do

begin

C_t = tập con của (C_k) chứa t;

for mỗi $c \in C_t$ do $c.Count++$;

end

$L_k = \{c \in C_k | c.Count \geq N.\min\_Supp\}$;

end

Return $(\cup_k L_k)$;

end

Thuật toán 2.2. Tạo ra tập ứng viên từ (L_{k-1}) : $C_k = L_{k-1} \times L_{k-1}$

Input. Tập các đơn vị xảy ra thường xuyên $(k-1)$ đơn vị dữ liệu L_{k-1}.

Output. Tập ứng viên k đơn vị dữ liệu

Begin

1. Kết nối L_{k-1} với L_{k-1} :

insert into C_k;

select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$

from $L_{k-1}p,\ L_{k-1}q$

where $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},\ p.item_{k-1} < q.item_{k-1}$;

2. Thu gọn bằng cách xoá $c \in C_k$ sao cho có tập con $(k-1)$ của $c \notin L_{k-1}$:

for với mỗi $c \in C_k$ do

for với mỗi tập con s gồm $(k-1)$ phần tử của c do

if $(s \notin L_{k-1})$ then

delete c from C_k;s

Return C_k;

end

Ví dụ 2. Cho L_3 = {{ABC}, {ABD}, {ACD}, {ACE}, {BCD}}

+ Bước kết nối:

{ABC} kết nối với {ABD} ta được {ABCD},

{ACD} kết nối với {ACE} ta được {ACDE}.

Như vậy, sau khi nối kết, ta có $C_4 = \{\{ABCD\}, \{ACDE\}\}$.

+ Bước thu gọn:

Mỗi phần tử của C_4 sau bước kết nối sẽ được kiểm tra điều kiện là L_3 có chứa tất cả các tập con của nó gồm 3 phần tử hay không?

{ABCD} thỏa mãn vì L_3 chứa tất cả các tập con gồm 3 phần tử của nó là {ABC}, {ABD}, {ACD}, {BCD},

{ACDE} không thỏa vì {ADE}$\notin L_3$. Kết quả C_4={{ABCD}}

Thuật toán 2.3. Tìm ra tất cả các luật kết hợp

Input: Tập L gồm các phần tử là k đơn vị dữ liệu xảy ra thường xuyên, $\min\_Conf$.

Output: Tập các luật kết hợp thỏa $\min\_Conf$

```
begin
    Result = ∅;
    for (với mỗi tập xảy ra thường xuyên l ∈ L) do
      begin
        for (mỗi tập con a ⊂ l sao cho a ≠ ∅) do
        if (Mức hỗ trợ(l)/Mức hỗ trợ(a) ≥ min_Conf) then
            Result = Result ∪{a → (l − a)};
      end
    return Ressult;
end
```

Ví dụ 3. Trong ví dụ 1, với $\min\_Conf = c_0 = 70\%$ và $\min\_Supp = s_0 = 40\%$ ta có tập L gồm các tập đơn vị dữ liệu xảy ra thường xuyên như sau:

$$L = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{AD\}, \{BE\}, \{CE\}, \{DE\}\},$$

có các luật kết hợp như sau:

$A \to D$ với $c = 71.42\%$ và s=41.66%,

$D \to A$ với $c = 71.42\%$ và s=41.66%,

$B \to E$ với $c = 83.33\%$ và s=41.66%,

$E \to B$ với $c = 71.42\%$ và s=41.66%.

3. THUẬT TOÁN KHAI THÁC TĂNG TRƯỞNG CÁC LUẬT KẾT HỢP

Trên cơ sở thuật toán tựa-Apriori, một số tác giả đã phát triển thuật toán này, chẳng hạn như FP-Tree ([5, 6]), FUP ([4]). Đối với thuật toán FP-Tree sẽ rất hữu hiệu đối với các CSDL nhỏ, vì thuật toán này đưa toàn bộ dữ liệu vào bộ nhớ chính để xử lý, do đó không phù hợp với quá trình tăng trưởng của dữ liệu. Với thuật toán FUP, là mở rộng gần nhất của thuật toán Apriori, FUP sẽ cập nhật các luật kết hợp khi có dữ liệu được thêm vào

bằng cách duyệt CSDL ban đầu.

Như vậy, mặc dù có những cải tiến nhưng trong các thuật toán ([4, 5, 6]) còn hai vấn đề tồn tại sau:

(i) Có khả năng xảy ra các tập ứng viên rất lớn,

(ii) Cần thiết phải duyệt CSDL quá nhiều lần.

Dưới đây là thuật toán khai thác các luật kết hợp, ý tưởng của thuật toán này như sau:

- Nếu các đơn vị dữ liệu là xảy ra thường xuyên trên tập D thì nó phải xảy ra thường xuyên trên ít nhất một tập con của D.
- Các tập dữ liệu xảy ra thường xuyên sẽ được thêm vào do thỏa $\min\_Supp$ từ các phần dữ liệu trước đó hoặc thỏa $\min\_Supp$ tại phần dữ liệu đang xét.
- Thuật toán được bắt đầu với việc tìm tập thường xuyên 2-đơn vị dữ liệu vì mỗi luật có ít nhất hai đơn vị dữ liệu.

Giả sử CSDL được chia thành n phần (mỗi phần ứng với một đơn vị thời gian lưu trữ dữ liệu) $D = \cup_{k=1}^{n} D_k$. Gọi $|D_k|$ là số giao tác trong D_k. Gọi $N_{Dk}(X)$ là số giao tác trong D_k chứa tập đơn vị dữ liệu X.

Thuật toán 3.1. Tạo ra tập thường xuyên gồm 2-đơn vị dữ liệu

Input: Tập đơn vị dữ liệu I

Tập các giao tác $D = \cup_{k=1}^{n} D_k$,

$\min\_Supp$.

Output: Tập L_2 gồm các phần tử là 2-đơn vị dữ liệu xảy ra thường xuyên

```
begin
  Ret = ∅;
  For k = 1 to n do begin
    For với mỗi tập 2-đơn vị dữ liệu X2 ∈ Dk
      if (X2 ∉Ret) then
        If (NDk(X2) ≥ min_Supp * |Dk|) then
          X2.Count = NDk(X2);
          X2.start = k;
          Ret = Ret ∪ X2;
        end if
      if (X2 ∈ Ret) then
        X2.Count = X2.Count + NDk(X2);
        If (X2.Count) < min_Supp × Σ_{i=Start}^{k} |Di| then
        Ret = Ret − X2;
          end if
      end for
    end for
  Return Ret;
```

end

Thuật toán 3.2. Tạo ra tất cả các tập thường xuyên trong D

Input: Tập đơn vị dữ liệu I

Tập các giao tác $D = \cup_{k=1}^{n} D_k$

$\min\_Supp, \min\_Conf$

Output: Tập L gồm các phần tử là k-đơn vị dữ liệu xảy ra thường xuyên

Begin

$L = \varnothing$;

Tạo ra tập thường xuyên gồm 2-đơn vị dữ liệu;

$m = 2$;

while $(C_m \neq \varnothing)$ do //Thực hiện phép kết nối với tất cả đơn vị dữ liệu

$C_m = C_{m-1} * C_{m-1}$;

$m = m + 1$;

end

For $k = 1$ to n do

For với mỗi tập đơn vị dữ liệu $X \in C_m$ do

$X.Count = X.Count + N_{Dk}(X)Z$;

end for

end for

For với mỗi tập đơn vị dữ liệu $X \in C_m$ do

If $(X.Count) \geq \min\_Supp * \sum_{i=1}^{n} |D_i|$ then

$L = L \cup X$;

Return L;

end

Ghi chú: Trong quá trình lưu trữ dữ liệu theo thời gian, chúng ta sẽ giảm đi một phần dữ liệu trước và tăng một phần mới, thao tác cũng tương tự bằng cách cập nhật lại Start, Count thích hợp.

Ví dụ 4. Trong ví dụ 1 ta xét với $\min\_Supp = 0,4$ và $\min\_Conf = 0,7$.
Trong lần xét thứ nhất ta xét D_1, D_2, D_3 :

D_1		
X_2	Start	Count
AD	1	2
AE	1	2
DE	1	2

$D_1 \cup D_2$		
X_2	Start	Count
AC	2	2
AD	1	3
AE	1	4
CE	2	2
DE	1	3

$D_1 \cup D_2 \cup D_2$		
X_2	Start	Count
AD	1	4
AE	1	4
BE	3	2
CE	2	3
DE	1	4

Ứng với D_1 ta có $[\min\_Supp * |D_k|] = [0.4 * 3] = 2$,

Ứng với $D_1 \cup D_2$ ta có $[\min\_Supp * |D_k|] = [0.4 * 6] = 3$,

Ứng với $D_1 \cup D_2 \cup D_3$ ta có $[\min\_Supp * |D_k|] = [0.4 * 9] = 4$.

Tập các tập đơn vị dữ liệu ứng viên trong $D_1 \cup D_2 \cup D_3$ là:

$\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{AD\}, \{AE\}, \{BE\}, \{CE\}, \{DE\}, \{ADE\}$.

Các tập đơn vị dữ liệu thường xuyên trong $D_1 \cup D_2 \cup D_3$ là:

$\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{AD\}, \{AE\}, \{BE\}, \{DE\}$ ($\{CE\}$ và $\{ADE\}$ không thỏa $\min\_Supp$)

$D - D_1$		
X_2	Start	Count
AD	2	2
AE	2	2
BE	3	2
CE	2	3
DE	2	2

Sau khi giảm D_1

$D \cup D_4$		
X_2	Start	Count
BC	4	2
BF	4	2
BE	3	3
CE	2	4
CF	4	2

Sau khi tăng D_4

Tập các tập đơn vị dữ liệu ứng viên trong $D_2 \cup D_3 \cup D_4$ là:

$\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{BC\}, \{BE\}, \{BF\}, \{CE\}, \{CF\}, \{CBE\}, \{CBF\}$.

Các tập đơn vị dữ liệu thường xuyên trong $D_2 \cup D_3 \cup D_4$ là:

$\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{BC\}, \{BE\}, \{CE\}$.

4. CHỨNG MINH TÍNH ĐÚNG ĐẮN CỦA THUẬT TOÁN KHAI THÁC TĂNG TRƯỞNG CÁC LUẬT KẾT HỢP

Định nghĩa 6. Ta gọi tỷ lệ xảy ra trên khoảng của tập đơn vị dữ liệu X trên các phần dữ liệu $D_i, D_{i+1}...D_j$, ký hiệu $t_{i,j}(X)$ là tỷ số giữa số giao tác chứa X và tổng giao tác trong chúng

$$t_{i,j}(X) = \frac{\sum_{k=1}^{j} N_{Dk}(X)}{\sum_{k=1}^{j} |D_k|}.$$

Bổ đề 1. *Tập đơn vị dữ liệu X là tập thường xuyên xảy ra ($X \in L$) sau khi xử lý qua phần dữ liệu D_j nếu và chỉ nếu $\exists i \leq j$ sao cho mọi số nguyên $k \in [i, j]$, ta có $t_{i,k}(X) \geq \min\_Supp$.*

Chứng minh:

1. Giả sử $\exists i \leq j$ sao cho mọi số nguyên $k \in [i, j]$, ta có $t_{i,k}(X) \geq \min\_Supp$. Có 2 trường hợp:

   (i) X không được đưa vào L trước khi xử lý phần D_i, bởi vì $t_{i,i}(X) \geq \min\_Supp$ nên X được đưa vào L sau khi xử lý D_i.

   (ii) X đã được đưa vào L trước khi xử lý D_i. Trong cả 2 trường hợp trên, áp dụng giả thiết $t_{i,k}(X) \geq \min\_Supp$ với $k \in [i, j]$, ta có X vẫn thuộc L khi xử lý từ D_i đến D_j.

2. Ngược lại: Giả sử $X \in L$ sau khi xử lý qua phần dữ liệu D_j. Như vậy, X sẽ được đưa vào L bởi một trong hai trường hợp sau:

   (i) Đưa L ngay khi xử lý D_j.

   (ii) Hoặc X được đưa vào L do xử lý trước phần dữ liệu D_j.

   Trong trường hợp (i), ta suy ra bằng cách đặt $j = i$, khi đó $t_{i,i}(X) \geq \min\_Supp$. Trong trường hợp (ii), $X \in L$ là do xử lý trước phần dữ liệu D_j, ta sẽ xét lần ngược từ D_j trở về trước, nghĩa là xét $D_{j-1}, D_{j-2}, ...$ cho đến khi phần dữ liệu D_i nào mà X lần đầu tiên là tập ứng viên (có thể X nhiều lần là tập ứng viên và ta chỉ xét lần đầu) như vậy, ta có cả hai trường hợp $t_{i,j}(X) \geq \min\_Supp$. ■

Bổ đề 2. *Tập đơn vị dữ liệu X là tập thường xuyên xảy ra ($X \in L$) sau khi xử lý qua phần dữ liệu D_j nếu và chỉ nếu $\exists i \leq j$ sao cho $t_{i,j}(X) \geq \min\_Supp$.*

Chứng minh:

1. Giả sử $\exists i \leq j$ sao cho $t_{i,j}(X) \geq \min\_Supp$. Ta gọi y là giá trị lớn nhất trong các giá trị x sao cho $t_{i,x}(X) < \min\_Supp$. Có 2 trường hợp xảy ra:

   + Nếu không tồn tại giá trị y, theo Bổ đề 1 ta có $X \in L$ sau khi xử lý phần dữ liệu D_j.
   + Nếu tồn tại y, khi đó $t_{y+1,j}(X) \geq \min\_Supp$ do $t_{i,y}(X) < \min\_Supp$ và $t_{i,j}(X) \geq \min\_Supp$, theo Bổ đề 1, ta cũng có $X \in L$ sau khi xử lý phần dữ liệu D_j.

2. Ngược lại: Hiển nhiên. ■

Định lý 1. *Tập đơn vị dữ liệu X là tập xảy ra thường xuyên nếu và chỉ nếu X là xảy ra thường xuyên sau phép xử lý bởi thuật toán trên.*

Chứng minh:

1. Giả sử X là tập đơn vị dữ liệu xảy ra thường xuyên được xử lý bởi thuật toán 3. Khi đó theo Thuật toán 3.2, ta có:

$$(X.Count) \geq \min\_Supp * \sum_{i=1}^{n} |D_i|,$$

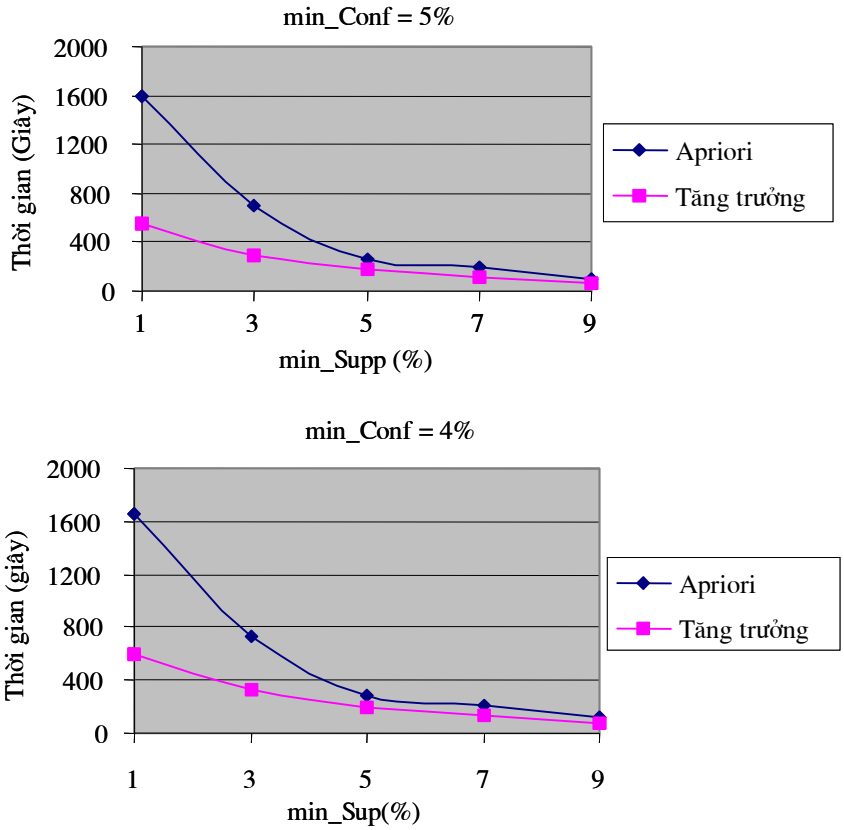
   như vậy X thỏa mãn điều kiện của tập xảy ra thường xuyên.

2. Ngược lại: Gọi n là số phần của CSDL: $D = \cup_{i=1}^{n} D_i$. Nếu X là tập đơn vị dữ liệu xảy ra thường xuyên thì $t_{1,n}(X) \geq \min\_Supp$, áp dụng Bổ đề 2 với $i = 1$ và $j = n$. ■

5. MỘT SỐ KẾT QUẢ THỰC NGHIỆM

Trong phần này chúng tôi nêu một số kết quả được kiểm chứng từ thực nghiệm so sánh thời gian thực hiện hai thuật toán trên, với tập dữ liệu là khoảng hơn 700.000 bản ghi về các kết quả điểm học tập với hơn 400 môn học của 15.000 sinh viên trong 5 năm học của Trường Đại học Thuỷ sản. Các kết quả này được thực hiện trên PC với CPU

tốc độ 570MHz, 128MB RAM, IDE Disk 18GB, các chương trình được viết bằng ngôn ngữ BORLAND C Ver4.



Với các điểm thi của các môn học chúng tôi xét theo các kết quả khác nhau tương ứng với các mức hỗ trợ khác nhau để tìm sự liên hệ giữa các kết quả học tập của các môn học trong một ngành học hoặc sự liên hệ giữa các môn học là môn học cơ bản, cơ sở hoặc chuyên ngành. Dữ liệu được phân hoạch theo mỗi học kỳ. Các kết qủa thu được rất nhiều, trong khuôn khổ bài báo này chúng tôi chỉ xin nêu các kết quả so sánh thời gian thực hiện hai thuật toán trên.

6. KẾT LUẬN

Trong bài báo này chúng tôi đã giới thiệu thuật toán khai thác với dữ liệu tăng trưởng theo thời gian, nhằm giải quyết vấn đề tìm ra các luật kết hợp khi dữ liệu tăng trưởng đáng kể. Với cách chia dữ liệu riêng phần và sử dụng các thông tin từ phần trước để tính tích luỹ cho phần sau đã làm giảm chi phí đáng kể cho việc tìm các luật.

Hướng phát triển của vấn đề là giải quyết trong trường hợp các luật kết hợp không đầy đủ hoặc là tìm ra các luật cho các đơn vị dữ liệu xuất hiện không cùng thời gian hay xuất hiện theo chu kỳ.

Lời cám ơn. Xin chân thành cám ơn PGS.TSKH Nguyễn Xuân Huy, GS.TSKH Nguyễn Đình Ngọc đã có những định hướng cho việc xây dựng thuật toán và PGS.TS Ngô Quốc Tạo đã có những góp ý quan trọng cho bài báo hoàn thành.

TÀI LIỆU THAM KHẢO

[1] R. Agarwal, C. Aggarwal, and V. Prasad, A tree projection algorithm for generation of frequentItemsets,*Journal of Parallel and Distributed Computing (Special Issue on High Performance DataMining)* (2000).

[2] R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in largedatabases, *Proc. of ACMSIGMOD* (1993).

[3] J. Ale, G. Rossi, An approach to discovering temporal association rules, *ACM Symposium on Applied Computing* (2000).

[4] D. Cheung, J. Han, V. Ng, and C. Wong, Maintenance of discovered association rules in largedatabases: An incremental updating technique, *Proc. of 1996 Int' l Conf. on Data Engineering* (1996).

[5] J. Han, J. Pei, Mining frequent patterns by pattern-growth: Methodology and Implications, *ACM SIGKDD Explorations (Special Issue on Scaleble Data Mining Algorithms)* (2001).

[6] J. Han, J.Pei, B.Mortazavi-Asl, Q. Chen,U. Dayal, and M. C. Hsu, FreeSpan: Frequentpattern-projected sequential pattern mining, *Proc. of 2000 Int. Conf. on Knowledge Discovery and Data Mining* (2000).

[7] J. L. Lin, M. Dunham, Mining association rules: Anti-skew algorithms,*Proc. of 1998 Int' l Conf. on Data Engineering* (1998).

[8] A. Savasere, E. Omiecinski, S. Navathe, An ecient algorithm for mining association rules in large databases, *Proc. of the 2th International Conference on Very Large Data Bases* (1996).

Nhận bài ngày 01 - 7 - 2003
Nhận lại sau sửa ngày 02 - 02 -2004