THUẬT TOÁN KHAI THÁC DỮ LIỆU TĂNG TRƯỞNG

NGUYỄN XUÂN HUY, ĐOÀN VĂN BAN, NGUYỄN HỮU TRỌNG, HUỲNH VĂN ĐỨC

I. MỞ ĐẦU

Bài toán tìm các luật kết hợp là bài toán cơ bản trong khai thác dữ liệu, gồm hai bước chính như sau: bước một, tìm tất cả các tập thường xuyên theo ngưỡng S_0 cho trước và bước hai, dựa vào các tập thường xuyên, tìm các luật kết hợp. Tất cả khó khăn của việc giải quyết bài toán tập trung ở bước một, một công việc tốn nhiều thời gian là xác định tất cả các tập mục dữ liệu thường xuyên theo một ngưỡng S_0 cho trước.

Sự phát triển của bài toán khai thác dữ liệu được Qiankun Zhao tổng kết trong [1]. Từ thuật toán AIS lần đầu tiên được Agrawal. R giới thiệu năm 1993 trong [2], thuật toán Apriori năm 1996 3] rồi từng bước được cải tiến: thuật toán FP-Tree do Han J, Pei H., Yin Y. đưa ra năm 2000 [4, thuật toán DCI được nhóm của Claudio Lucchese đề nghị năm 2005 [5], thuật toán CHARM được nhóm Mohammed J. Zaki đưa ra năm 2005 [6], thuật toán LCM được nhóm Takeaki Uno đưa ra năm 2006 [7], thuật toán BFS được Vicky Choi đưa ra năm 2006 [8],... chủ yếu xử lí trên tập dữ liệu xác định trước. Ta biết rằng, các tập dữ liệu được bổ sung và tăng trưởng theo thời gian, do vậy các tập thường xuyên và các luật kết hợp đã được tính toán không còn giá trị. Ngoài ra, với một dữ liệu ổn định, khi cần tìm các tập thường xuyên với độ hỗ trợ khác, công việc phải tính lại từ đầu.

Để khắc phục điều này, chúng tôi đề nghị một thuật toán tăng trưởng, với ý tưởng cơ bản như sau:

1) Với một ngữ cảnh khai thác dữ liệu **(T, I, ∂)** với $||\mathbf{T}|| = m$, $||\mathbf{I}|| = n$ ban đầu, thuật toán tính độ hỗ trợ của tất cả các tập mục dữ liệu có trong ∂ rồi lưu trữ trong tập $K = \{(X, Supp(X)) \mid X \in \mathbf{I}$ và X nằm trong ít nhất một giao tác nào đó$\}$. Theo thời gian, số lượng các giao tác tăng dần, thuật toán chỉ tính toán với dữ liệu tăng thêm, không cần tính toán lại từ đầu. Với cách tổ chức này, khi cần tìm các tập thường xuyên thỏa mãn ngưỡng S_0, ta chỉ cần lọc ra những tập mục dữ liệu trong K thỏa $Supp(X) \geq S_0$.

2) Để tính độ hỗ trợ của các tập mục dữ liệu, không cần phải tính cho tất cả các tập mục dữ liệu trong Subset(**I**), mà chỉ cần tính cho các tập mục dữ liệu xuất hiện trong các giao tác.

Bài viết có 5 phần. Sau phần mở đầu, chúng tôi trình bày các khái niệm cơ bản của bài toán khai thác dữ liệu ở phần 2. Phần 3 là phần chính, chúng tôi đưa ra một thuật toán tính độ hỗ trợ của tất cả các tập dữ liệu tồn tại trong ngữ cảnh khai thác. Phần 4 nêu kết quả áp dụng thuật toán. Cuối cùng, chúng tôi tổng kết các kết quả đã đạt được.

II. BÀI TOÁN KHAI THÁC DỮ LIỆU

Định nghĩa 1. *Ngữ cảnh khai thác dữ liệu*

Cho $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ là tập hợp các mục dữ liệu, và $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ là tập hợp các giao tác. Trên tích Descartes $\mathbf{T} \times \mathbf{I}$ ta định nghĩa một quan hệ hai ngôi ∂ như sau:

$\forall (t, i) \in \mathbf{T} \times \mathbf{I}$: $t \partial i \Leftrightarrow$ **Giao tác t có chứa mục dữ liệu** i.

∂ được gọi là quan hệ khai thác dữ liệu. Bộ ba **(T, I, ∂)** được gọi là **ngữ cảnh khai thác dữ liệu**.

Mỗi mục dữ liệu $i \in \mathbf{I}$ còn được gọi là một *thuộc tính* của **I**.

Định nghĩa 2. *Ma trận giao tác*

Cho ngữ cảnh khai thác dữ liệu **(T, I, ∂)**, ta định nghĩa ma trận $\mathbf{M} = (m_{ij})_{mxn}$ với $||\mathbf{T}|| = m$, $||\mathbf{I}|| = n$, là một ma trận nhị phân với:

$$m_{ij} = \begin{cases} 1 \text{ khi } (i,j) \in \partial \\ 0 \text{ khi } (i,j) \notin \partial \end{cases}$$

M gọi là ma trận giao tác. Hàng thứ i của ma trận biểu diễn giao tác thứ i, cột thứ j của ma trận biểu diễn mục dữ liệu thứ j.

Mỗi tập hợp con $X \subseteq \mathbf{I}$ gọi là tập mục dữ liệu (itemset), mỗi tập con $S \subseteq \mathbf{T}$ gọi là tập định danh giao tác (tidset). Để thuận tiện trong kí hiệu, ta viết X = ABC thay cho X = {A, B, C}, S = 123 thay cho S = {1, 2, 3}.

Định nghĩa 3. *Kết nối Galois*

Cho ngữ cảnh khai thác dữ liệu **(T, I, ∂)**. Ta định nghĩa hai ánh xạ:

$$tran: SubSet(\mathbf{I}) \rightarrow SubSet(\mathbf{T})$$

$$X \subseteq \mathbf{I}: tran(X) = \{s \in \mathbf{T} \mid \forall x \in X, (s, x) \in \partial\} = \{s \in \mathbf{T} \mid \forall x \in X, s.x = 1\}$$

trong đó, s.x chỉ sự xuất hiện của mục dữ liệu x trong giao tác s nếu cho giá trị 1 và *tran(*X) là tập hợp tất cả các giao tác của **T** chứa tất cả các mục dữ liệu trong X.

$$item : SubSet(\mathbf{T}) \rightarrow SubSet(\mathbf{I})$$

$$S \subseteq \mathbf{T}: item(S) = \{x \in \mathbf{I} \mid \forall s \in S, (s, x) \in \partial\} = \{x \in \mathbf{I} \mid \forall s \in S, s.x = 1\}$$

*item(*S) là tập hợp tất cả các mục dữ liệu của **I** xuất hiện ở tất cả các giao tác trong S.

Cặp ánh xạ (*tran, item*) được gọi là kết nối Galois trên $\mathbf{T} \times \mathbf{I}$.

Định nghĩa 4. *Độ hỗ trợ*

Độ hỗ trợ (support) của một tập mục dữ liệu X, kí hiệu Supp(X) là số các giao tác trong **T** xuất hiện tất cả các mục dữ liệu của X.

$$Supp(X) = ||\{t \in \mathbf{T} \mid item(t) \supseteq X\}||.$$

Định nghĩa 5. *Tập mục dữ liệu thường xuyên*

Cho S_0 là một số nguyên và $X \subseteq \mathbf{I}$. Ta nói X là ***tập mục dữ liệu thường xuyên*** (*frequent itemset*) ***theo ngưỡng S_0*** nếu $Supp(X) \geq S_0$.

Đặt $Fi(\mathbf{I}, S_0) = \{X \subseteq \mathbf{I} \mid Supp(X) \geq S_0\}$.

Để đơn giản trong cách gọi, từ đây khi nói: ***X là tập thường xuyên*** thay vì nói ***X là tập mục dữ liệu thường xuyên*** và được hiểu theo nghĩa: ***X là tập mục dữ liệu thường xuyên theo ngưỡng S_0*** cho trước.

Một ngữ cảnh khai thác dữ liệu với các tập thường xuyên theo ngưỡng S_0 gọi là ngữ cảnh khai thác dữ liệu theo ngưỡng S_0, **kí hiệu B = (T, I, ∂, S_0)**.

Với ngữ cảnh khai thác dữ liệu **(T, I, ∂)** và $||\mathbf{I}|| = n$ thì không gian tìm kiếm tất cả các tập thường xuyên là 2^n.

Định nghĩa 6. *Luật kết hợp*

Một *luật kết hợp (Association rule)* trên ngữ cảnh khai thác dữ liệu **(T, I, ∂)** là một biểu thức $X_1 \rightarrow X_2$, với X_1, X_2 là các tập mục dữ liệu ($X_1, X_2 \subseteq \mathbf{I}$) và $X_1 \cap X_2 = \phi$.

Độ hỗ trợ của luật kết hợp $X_1 \rightarrow X_2$, kí hiệu $Supp(X_1 \rightarrow X_2)$, được định nghĩa:

$Supp(X_1 \rightarrow X_2) = Supp(X_1 \cup X_2) = Supp(X_1X_2)$.

Định nghĩa 7. *Luật thường xuyên*

Cho S_0 là một số nguyên và luật kết hợp f: $X_1 \rightarrow X_2$, ta nói f là ***luật thường xuyên*** *(frequent rule)* ***theo ngưỡng*** S_0 nếu $X_1 \cup X_2$ là ***tập mục dữ liệu thường xuyên*** theo *ngưỡng* S_0, nghĩa là: $Supp(X_1 \rightarrow X_2) \geq S_0$.

Định nghĩa 8. *Độ tin cậy*

Độ tin cậy (Confidence) của luật $X_1 \rightarrow X_2$, kí hiệu $Conf(X_1 \rightarrow X_2)$, là tỉ số:

$$p = Conf(X_1 \rightarrow X_2) = Supp(X_1X_2) / Supp(X_1).$$

Định nghĩa 9. *Luật tin cậy theo* C_0

Cho S_0 là một số nguyên, $C_0 \in (0, 1]$ và luật kết hợp f: $X_1 \rightarrow X_2$, ta nói f là ***luật tin cậy*** *(Confident rule)* ***theo ngưỡng*** S_0 và C_0 nếu f là ***luật thường xuyên theo ngưỡng*** S_0 và $Conf(X_1 \rightarrow X_2) \geq C_0$.

III. THUẬT TOÁN TĂNG TRƯỞNG

1. Các bước của thuật toán

Bước 1: Với ngữ cảnh khai thác dữ liệu **(T, I, ∂)**, công việc đầu tiên là tính độ hỗ trợ của các tập mục dữ liệu xuất hiện trong các giao tác của **T** và lưu vào tập:

Thực hiện thuật toán 1. bằng cách gọi hàm **Supp(M)** với M là ma trận giao tác để tính

$$\mathbf{K} = \{(item(t), Sup\_X) | \text{ với } t \in \mathbf{T}, Sup\_X = Supp(item(t))\}$$

Bước 2:

Khi dữ liệu tăng thêm với ngữ cảnh khai thác **(T', I, ∂')**, công việc tiếp theo là tính độ hỗ trợ của các tập mục dữ liệu xuất hiện trong các giao tác của **T'** và lưu vào tập:

Thực hiện thuật toán 1. bằng cách gọi hàm **Supp(M)** với M là ma trận giao tác để tính

$$\mathbf{K'} = \{(item(t), Sup\_X) | \text{ với } t \in \mathbf{T'}, Sup\_X = Supp(item(t))\}$$

Sau khi tính toán **K'**, ta gộp **K'** vào tập **K** đã tính trước bằng cách gọi thủ tục **Union(K', K, K)** theo thuật toán 4.

Bước 3: Tìm các tập thường xuyên theo ngưỡng S_0 cho trước theo thuật toán 5., nghĩa là gọi thủ tục **Frequent(S_0, K,** K_{S_0}**)** để tính

$$K_{S_0} = \{(X, Sup\_X) | (X, Sup\_X) \in K \text{ và } Sup\_X \geq S_0\}.$$

Thuật toán 1. *Tính độ hỗ trợ của mọi tập mục dữ liệu item(t)* $\subseteq I$ *với* $t \in \mathbf{T}$.

Function Supp(M)

Input: M_{mxn} là ma trận giao tác của ngữ cảnh khai thác dữ liệu **(T, I, ∂)**;

Output: K = {(X, Sup_X) | X = *item*(t) với t ∈ T và Sup_X = Supp(X)};

Method:

```
   begin
K := ϕ;
For each t = (i1, i2, …, in) ∈ M do begin   //t là một hàng của ma trận M
      X := item(t);
      Inserted:= False;
      K1 := ϕ;                  //Cấu trúc của K1 giống như cấu trúc của K
      If (K = ϕ) and (Not X ⊆ Y) and (Not Inserted) then
begin
          Insert((X, 1), K1);
          Inserted := True;
      end
      For each (Y, Sup_Y) ∈ K do begin
          If Z = Y ∩ X ≠ ϕ then
              Insert((Z, Sup_Y + 1), K1);
          If (Not X ⊆ Y) and (Not Inserted) then begin
              Insert((X, 1), K1);
              Inserted := True;
          end
      end
      Merge(K1, K);
end
Return(K);
  end
```

Thuật toán 2. *Thêm một tập mục dữ liệu vào tập lưu trữ.*

Procedure Insert((X, Sup_X), K);

Input: (X, Sup_X) với X ⊆ I và Sup_X = Supp(X);

K = {(X, Sup_X) | X = *item*(t) với t ∈ T và *Sup_X = Supp(X)*};

Output: K = {(X, Sup_X) | X = *item*(t) với t ∈ T và *Sup_X = Supp(X)*};

Method:

```
   begin
If (∃ (Y, Sup_Y) ∈ K) and (X = Y) then begin
      Sup_Z := Max(Sup_X, Sup_Y);
      K := (K \ {(Y, Sup_Y)}) ∪ {(X, Sup_Z)};
end
```

```
Else
        K := K ∪ {(X, Sup_X)};
    end
```

Bổ đề 1. *Thuật toán 2 chèn thêm một tập mục dữ liệu vào tập lưu trữ là đúng đắn.*

Chứng minh: Khi thêm một tập mục dữ liệu (X, Sup_X) vào tập lưu trữ **K**, ta lần lượt so sánh (X, Sup_X) với tất cả các bộ (Y, Sup_Y) trong **K**. Khi so sánh, một trong hai trường hợp sau sẽ xảy ra:

Trường hợp 1: $\exists$(Y, Sup_Y) $\in$ **K** thỏa X = Y, nghĩa là X đã xuất hiện Sup_Y lần trước đó, bây giờ chèn thêm bộ (X, Sup_X) với X xuất hiện Sup_X lần, tức X xuất hiện Max(Sup_X, Sup_Y) lần nên thuật toán thay Sup_Y bởi Max(Sup_X, Sup_Y) là đúng.

Trường hợp 2: $\forall$(Y, Sup_Y) $\in$ **K** đều có X $\neq$ Y, tức X chưa xuất hiện lần nào, do đó chèn thêm (X, Sup_X) vào **K** là đúng.

Như vậy, thuật toán 2 là đúng.

Thuật toán 3. *Nối tập đã tính độ hỗ trợ K_1 vào tập K.*

Prcedure ***Merge(K_1, K);***

Input: $K_1 = \{(X, Sup\_X) \mid X \in I$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

$K = \{(X, Sup\_X) \mid X \in I$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

Output: $K = \{(X, Sup\_X) \mid X \in I$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

Method:

```
begin
For each (X, Sup_X)∈ K1 do
    Insert((X, Sup_X), K);
end
```

Bổ đề 2. *Thuật toán 3 nối tập vừa tính độ hỗ trợ K_1 vào K là đúng.*

Chứng minh:

Trong thuật toán 1, quá trình tính toán cần tính giao của tập mục dữ liệu trong từng giao tác với tất cả các tập mục dữ liệu đã xét. Trong quá trình tính, một tập mục dữ liệu có thể xuất hiện nhiều lần, do đó, khi một tập mục dữ liệu xuất hiện ta phải duyệt lại từ đầu. Nếu tập đã có, ta chỉ thay độ hỗ trợ mới nếu độ hỗ trợ trước nhỏ hơn; nếu tập mục dữ liệu chưa xuất hiện, ta phải thêm tập mục dữ liệu này vào **K**.

Do đó, thuật toán 3 nối từng phần tử (X, Sup_X) của $\mathbf{K_1}$ vào **K** là đúng.

Định lí 1. *Thuật toán 1 tìm hết tất cả các tập mục dữ liệu xuất hiện trong ngữ cảnh khai thác dữ liệu cùng với độ hỗ trợ của nó.*

Chứng minh:

i) Giả sử $Z = \{z_1, \ldots, z_j\} \subseteq \mathbf{I}$ với $j \geq 1$ xuất hiện trong một giao tác nào đó của (**T**, **I**, ∂), nghĩa là $\exists t \in \mathbf{T}$ sao cho $Z \subseteq item(t)$, ta chứng minh $\exists$(Y, Sup_Y) $\in$ **K** (*Output của Supp(M)*) để $Z \subseteq Y$ và ngược lại.

Thật vậy, vòng lặp ***For*** thứ nhất của thuật toán vét hết các phần tử của **T**, nên sẽ tồn tại t ∈ **T** thỏa $Z \subseteq X = item(t)$. Trong vòng lặp ***For*** thứ hai của thuật toán, có hai khả năng xảy ra:

Khả năng thứ nhất: $\exists(Y, Sup\_Y) \in \mathbf{K}$ thỏa $Z \subseteq Z_1 = X \cap Y$, thuật toán thực hiện lệnh ***If*** thứ nhất, chèn Z_1 chứa Z vào $\mathbf{K}_1$ sau đó nối vào **K.**

Khả năng thứ hai: $\forall(Y, Sup\_Y) \in K$: $Z \not\subset X \cap Y$, vì $Z \subseteq X$ nên $X \not\subset Y$, thuật toán thực hiện lệnh ***If*** thứ hai, chèn X chứa Z vào $\mathbf{K}_1$, sau đó nối $\mathbf{K}_1$ vào **K**.

Cả hai khả năng đều có $(Y, Sup\_Y) \in \mathbf{K}$ để $Z \subseteq Y$.

Ngược lại, với $(Y, Sup\_Y) \in \mathbf{K}$, theo thuật toán thì phải có ít nhất một t ∈ **T** sao cho $Y \subseteq item(t)$.

ii) Với $(X, Sup\_X) \in \mathbf{K}$, ta chứng minh $Sup\_X = Supp(X)$.

Với $(X, Sup\_X) \in \mathbf{K}$, có các khả năng xảy ra:

Khả năng thứ nhất: ∃t ∈ **T** thỏa $item(t) = X$. Với vòng lặp ***For*** thứ hai, nếu X chưa là tập con của bất kỳ item(t') nào trước đó, tức X xuất hiện lần đầu tiên trong giao tác t, thuật toán lưu (X, 1) vào $\mathbf{K}_1$ là đúng. Ngược lại, cứ mỗi lần xuất hiện một $(Y, Sup\_Y) \in \mathbf{K}$ thỏa $X \subseteq Y$, nghĩa là X đã xuất hiện Sup_Y lần trước đó, bây giờ xuất hiện một lần nữa, ta chèn $(X, Sup\_Y + 1)$ vào $\mathbf{K}_1$ là đúng.

Khả năng thứ hai: ∀t ∈ **T** đều có $item(t) \neq X$, phải tồn tại một số hữu hạn các giao tác $t_{l_1}, \ldots, t_{l_j} \in \mathbf{T}$, $j \geq 1$ sao cho $X = item(t_{l_1}) \cap \ldots \cap item(t_{lj})$. Theo thuật toán, mỗi lần có một t_{l_i} để $X \subseteq item(t_{l_i})$ thì tăng Sup_X lên 1 đơn vị, nghĩa là số lần xuất hiện của X tăng lên 1.

Vậy theo định nghĩa, $Sup\_X = Supp(X)$.

iii) Việc kết nối $\mathbf{K}_1$ vào **K** là đúng, theo Bổ đề 2. ◆

Kết luận: Thuật toán 1 tìm hết tất cả các tập mục dữ liệu xuất hiện trong ngữ cảnh khai thác dữ liệu cùng với độ hỗ trợ của nó.

Thuật toán 4. *Nối tập đã tính độ hỗ trợ K_1, K_2 thành tập K*

K_1 đã tính trước, K_2 vừa tính tăng trưởng

Procedure *Union(K_1, K_2, K);*

Input: $K_1 = \{(X, Sup\_X) \mid X \in \mathbf{I}$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

$K_2 = \{(X, Sup\_X) \mid X \in \mathbf{I}$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

Output: $K = \{(X, Sup\_X) \mid X \in \mathbf{I}$ và $tran(X) \neq \phi, Sup\_X = Supp(X)\}$;

Method:

```
   begin
K := ϕ;
For each (X, Sup_X) ∈ K1 do begin
      Inserted := False;
      For each (Y, Sup_Y) ∈ K2 do begin
          If Z = Y ∩ X ≠ ϕ then
               Insert((Z, Sup_Y + Sup_Y), K);
```

```
            If X = Z Then
                Inserted :=True;
            If Not Y = Z then
                Insert((Y, Sup_Y), K);
    end
end
 If Not Inserted Then
            Insert(X, Sup_X), K);
   end
```

Định lí 2. *Thuật toán 4 nối tập đã tính độ hỗ trợ K_1 và K_2 thành tập K là đúng đắn.*

Chứng minh:

Khi lấy từng tập mục dữ liệu (X, Sup_X) $\in$ $\mathbf{K}_1$ giao với từng tập mục dữ liệu (Y, Sup_Y) $\in$ $\mathbf{K}_2$. Ba khả năng sau có thể xảy ra:

Khả năng thứ nhất: $Z = X \cap Y \neq \phi$, nghĩa là Z xuất hiện Sup_X lần trong $\mathbf{K}_1$ và Sup_Y lần trong $\mathbf{K}_2$, do đó chèn (Z, Sup_X + Sup_Y) vào **K** là đúng.

Khả năng thứ hai: $Y \neq Z$, nghĩa là Y là một tập mục dữ liệu chưa chèn vào **K**, do đó việc chèn thêm (Y, Sup_Y) vào **K** là đúng.

Khả năng thứ ba: $X \neq Z$, nghĩa là X là một tập mục dữ liệu chưa chèn vào **K**, do đó việc chèn thêm (X, Sup_X) vào **K** là đúng. Như vậy, thuật toán 4 là đúng.

Thuật toán 5. *Tìm tất cả các tập thường xuyên theo ngưỡng S_0*

Procedure Frequent(S_0, K, K_1);

Input: $\mathbf{K} = \{(X, Sup\_X) \mid X = item(t)$ với $t \in \mathbf{T}$ và $Sup\_X = Supp(X)\}$;

Output: $K_1 = \{(X, Sup\_X) \mid (X, Sup\_X) \in \mathbf{K}$ và $Sup\_X \geq S_0\}$;

Method:

```
  begin
K1 := ϕ;
For each (X, Sup_X)∈ K do begin
      If Sup_X ≥ S0 then
          K1 := K1 ∪ {(X, Sup_X)};
end
  end
```

Giải thích thuật toán 5

Duyệt qua từng phần tử (X, Sup_X) $\in$ K, nếu $Sup\_X \geq S_0$, tức X là tập mục dữ liệu thỏa ngưỡng S_0, ta lưu (X, Sup_X) vào K_1.

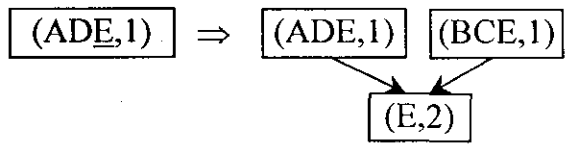
IV. VÍ DỤ ÁP DỤNG

A) Tìm K = Supp(M) với M là ma trận giao tác được cho trong bảng sau:

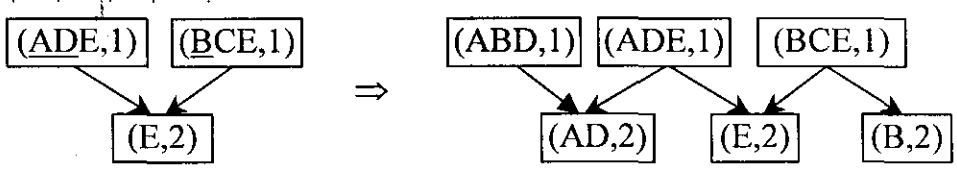
	A	B	C	D	E
①	1	0	0	1	1
②	0	1	1	0	1
③	1	1	0	1	0
④	1	0	1	0	1
⑤	1	0	1	1	1
⑥	0	1	1	0	0

item(t_1) = ADE

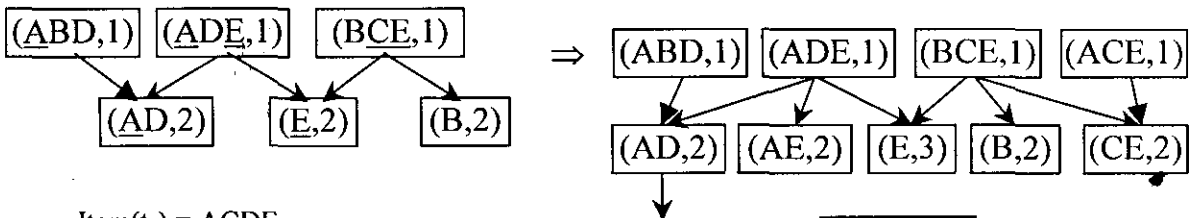
item(t_2) = BCE



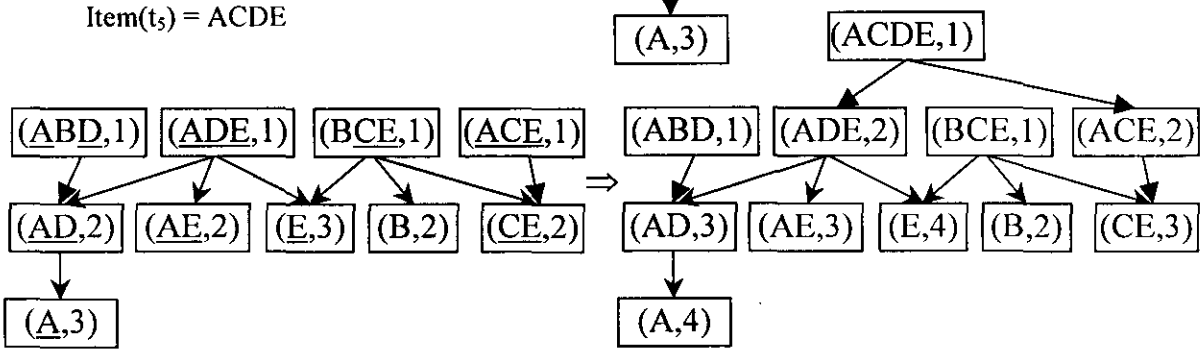
item(t_3) = ABD



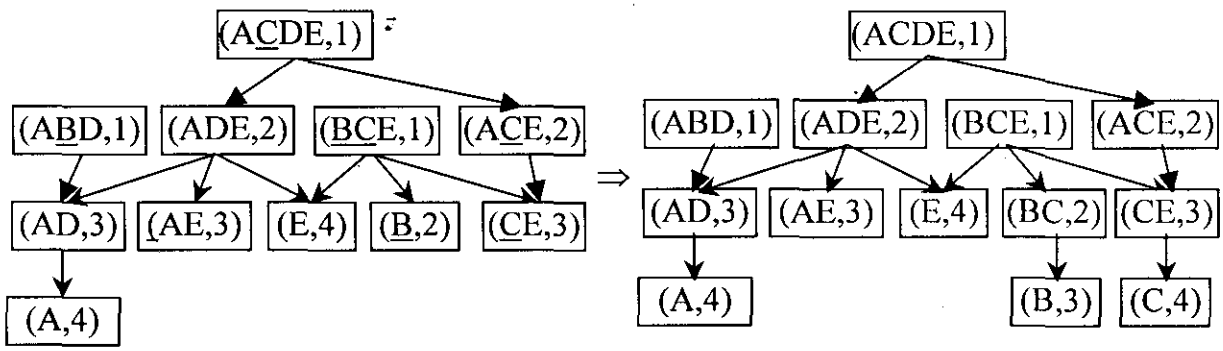
item(t_4) = ACE



Item(t_5) = ACDE



Item(t_6) = BC



K = {(ACDE,1), (ABD,1), (ADE,2), (BCE,1), (ACE,2), (AD,3), (AE,3), (E,4) (BC,2), (CE,3), (A,4), (B,3), (C,4)}

Với ví dụ này, ta nhận thấy chỉ cần tính toán 13 tập mục dữ liệu xuất hiện trong các giao tác thay vì tính 31 tập mục dữ liệu con của **I**.

Chúng tôi đã cài đặt thuật toán này trên máy PC, với ma trận khai thác dữ liệu gồm 1000 hàng và 50 cột, Max{Supp({i}) = 448, Min{Supp({i}) = 53, Average{Supp({i}) = 253, Max{Sum(t_j)} = 25, Min{Sum(t_j)} = 5, Average{Sum(t_j)} = 12,691. Kết quả: 487 763 tập mục dữ liệu cần tính thay vì 2^{50}-1= 1 125 899 906 842 622 tập.

V. KẾT LUẬN

Trong bài báo này, chúng tôi đã đưa ra thuật toán tăng trưởng, giải quyết được vấn đề tính và lưu trữ độ hỗ trợ của dữ liệu tăng trưởng theo thời gian mà không phải tính toán lại trong quá trình khai thác dữ liệu. Thuật toán rất đơn giản về ngữ nghĩa và dễ dàng trong việc cài đặt.

TÀI LIỆU THAM KHẢO

1. Qiankun Zhao, Sourav S. Bhowmick - Association Rule Mining: A Survey Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003116, 2003.
2. R. Agrawal, T. Imielinski, A. Swami - Mining Associations between Sets of Items in Massive Databases, In: Proc.of the 1993 ACM-SIGMOD Int'l Conf. on Management of Data, pp. 207-216.
3. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. L. Verkamo - Fast Discovery of Association Rules, In: Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 307-328.
4. J. Han, H. Pei, and Y. Yin - Mining Frequent Patterns without Candidate Generation, In: Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX), ACM Press, New York, NY, USA, 2000.
5. Claudio Lucchese, Salvatore Orlando, Raffaele Perego - Fast and Memory Efficient Algorithm to Mine Frequent Closed Itemsets, IEEE Transaction On Knowledge and Data Engineering **18** (1) (2006) 21-36.
6. Mohammed J. Zaki and Ching-Jui Hsiao Charm - Efficient algorithm for mining closed itemsets and their lattice structure, IEEE Transactions on knowledge and data engineering **17** (4) (2005) http://www.cs.rpi.edu/_zaki
7. Takeaki Uno, Masashi Kiyomi, Hiroki Arimura *Lcm Ver.2* - Efficient mining algorithms for frequent/closed/maximal itemsets, IEEE ICDM'04 Workshop FIMI'04 (International Conference on Data Mining, Frequent Itemset Mining Implementations), 2004. http://research.nii.ac.jp/~uno/papers/0411lcm2.pdf.
8. Vicky Choi - Faster algorithms for constructing a concept (Galois) lattice, *arXIV:cs.DM/0602069*, V. 2 1 Jun 2006.
9. Francisco Guil, Alfonso Bosch, Roque Marín - Tset: an algorithm for mining Frequent temporal patterns. Http://www.lsi.us.es/~aguilar/ecml2004/FP7.PDF.
10. David Hand, Heikki Mannila and Padhraic Smyth - Principles of Data Mining, The MIT Press © 2001.