Tạp chí Tin học và Điều khiển học, T.24, S.4 (2008), 307–320

KHAI PHÁ HIỆU QUẢ TẬP MỤC LỢI ÍCH CAO TRONG CƠ SỞ DỮ LIỆU LỚN

VŨ ĐỨC THI[1], NGUYỄN HUY ĐỨC[2]

[1] *Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam*
[2] *Khoa Thông tin - Máy tính, Trường Cao đẳng Sư phạm Trung ương*

Abstract. This paper proposes a new disk-based High Utility Itemsets mining algorithm, which achieves its efficiency by applying three new ideas. First, transactional data is converted into a new database layout called Transactional Array that prevents multiple scanning of the database during the mining phase. Second, for each item, a relatively small independent tree is built summarizing co-occurrences. Finally, a simple and non-recursive mining process reduces the memory requirements as minimum candidacy generation and counting is needed. We have tested our algorithm on several large transactional databases and the results show that our algorithm works efficiently.

Tóm tắt. Bài báo đề xuất thuật toán mới khai phá tập mục lợi ích cao trên tập dữ liệu lớn, thuật toán áp dụng ba ý tưởng mới. Thứ nhất, cơ sở dữ liệu được chuyển đổi sang dạng biểu diễn mới gọi là mảng giao tác để giảm số lần duyệt khi khai phá. Thứ hai, với mỗi mục dữ liệu, xây dựng một cây nhỏ chứa các mục dữ liệu cùng xuất hiện với mục này. Và cuối cùng, khai phá cây không đệ quy đã giảm yêu cầu về bộ nhớ và tính toán. Thuật toán đã được thử nghiệm trên một số cơ sở dữ liệu giao tác lớn và kết quả cho thấy thuật toán thực hiện hiệu quả.

1. MỞ ĐẦU

Mô hình khai phá tập mục lợi ích cao đã được Yao và cộng sự đề xuất (Hong Yao và Hamilton, 2006; H. Yao, Hamilton và Butz, 2004) [7 – 9]. Trong mô hình khai phá tập mục lợi ích cao, giá trị của mục dữ liệu trong giao tác là một số (gọi là giá trị khách quan), ngoài ra còn có bảng lợi ích cho biết lợi ích mang lại khi bán một đơn vị hàng đó (gọi là giá trị chủ quan). Lợi ích của một tập mục là số đo lợi nhuận đóng góp của tập mục đó trong cơ sở dữ liệu. Khai phá tập mục lợi ích cao là khám phá tất cả các tập mục có lợi ích không nhỏ hơn ngưỡng lợi ích tối thiểu quy định bởi người sử dụng.

Trong [7 – 9], Hong Yao và Howard Hamilton đã đề xuất phương pháp khai phá và các chiến lược tỉa dựa trên các tính chất của ràng buộc lợi ích, thể hiện trong hai thuật toán Umining và Umining-H. Các thuật tỉa mà hai thuật toán này áp dụng có khả năng thu gọn phần nào tập ứng viên, tuy vậy có những nhược điểm nên hiệu quả không cao.

Trong [10], Liu và cộng sự (Y. Liu, Liao và Choudhary, 2005) đã đưa ra khái niệm lợi ích của giao tác và lợi ích của tập mục tính theo lợi ích của các giao tác chứa nó, gọi là lợi ích TWU (Transaction-weighted Utilization). Tác giả đã chứng minh lợi ích theo giao tác TWU có tính chất phản đơn điệu như tính chất của tập mục thường xuyên và tập tất cả các tập

mục lợi ích cao chứa trong tập tất cả các tập mục lợi ích TWU cao. Nhờ những tính chất này của lợi ích TWU, Liu đã đề xuất thuật toán gồm hai pha để khai phá tập mục lợi ích cao. Tuy nhiên, thuật toán thực hiện kém hiệu quả khi khai phá các tập dữ liệu dày hoặc mẫu dài vì tốn nhiều thời gian cho việc sinh các tập ứng viên và tính lợi ích TWU của các tập ứng viên đó trong mỗi lần duyệt cơ sở dữ liệu.

Trong bài báo này, chúng tôi đề xuất một thuật toán khai phá tập mục lợi ích cao trong các cơ sở dữ liệu lớn. Thuật toán tổ chức lại dữ liệu và lưu giữ ở bộ nhớ ngoài của máy tính, quá trình khai phá chỉ đưa vào bộ nhớ trong một phần nhỏ của dữ liệu và khai phá thực hiện theo phương pháp không đệ quy theo ý tưởng của thuật toán COFI-tree do Mohammad El-Hajj và Osmar R. Zaiane đề xuất năm 2003 [13, 14]. Toàn bộ dữ liệu lưu ở bộ nhớ ngoài cho phép thuật toán khai phá được trên những tập dữ liệu rất lớn. Sau khi dữ liệu được tổ chức lại và lưu trên đĩa, có thể khai phá tập mục lợi ích cao với các ngưỡng lợi ích khác nhau mà không cần tổ chức lại dữ liệu.

Nội dung tiếp theo của bài báo gồm: Phần 2 nêu một số định nghĩa, thuật ngữ và phát biểu bài toán khai phá tập mục lợi ích cao. Phần 3 tóm tắt nội dung của thuật toán khai phá tập mục thường xuyên nhờ cây COFI-tree. Phần 4 trình bày thuật toán mới khai phá tập mục lợi ích cao trong các cơ sở dữ liệu lớn. Phần 5 đánh giá thuật toán và kết luận dựa trên việc phân tích thuật toán và các thử nghiệm.

2. BÀI TOÁN KHAI PHÁ TẬP MỤC LỢI ÍCH CAO

Phần này nêu một số định nghĩa và thuật ngữ mô tả bài toán khai phá tập mục lợi ích cao theo [7-9] (H. Yao, Hamilton và Butz, 2004; Hong Yao, Hamilton và Geng, 2006).

Cho tập các mục (item) $I = \{i_1, i_2, ..., i_n\}$. Một giao tác (transaction) T là một tập con của $I, T \subseteq I$. Cơ sở dữ liệu là một tập các giao tác $DB = \{T_1, T_2, ..., T_m\}$. Mỗi giao tác được gán một định danh TID. Một tập mục con $X \subseteq T$, gồm k mục phân biệt được gọi là một k-tập mục. Giao tác T gọi là chứa tập mục X nếu $X \subseteq T$.

Định nghĩ 2.1. Ta gọi giá trị của mục i_p trong giao tác T_q là giá trị khách quan (objective value) của mục i_p tại giao tác T_q, ký hiệu là $o(i_p, T_q)$.

Định nghĩ 2.2. Ta gọi giá trị do nhà kinh doanh gán cho mục i_p trong cơ sở dữ liệu là giá trị chủ quan (subjective value) của mục i_p và ký hiệu là $s(i_p)$.

Lợi ích của một mục trong một giao tác được đánh giá thông qua hàm 2 biến sau:

Định nghĩ 2.3. Ký hiệu x là giá trị khách quan, y là giá trị chủ quan của một mục. Một hàm 2 biến $f(x, y) : R \times R \to R$, đơn điệu tăng theo x và theo y, được gọi là hàm lợi ích.

Thông thường hàm lợi ích được xác định như sau: $f(x, y) = x \times y$.

Bảng 2.1. Cơ sở dữ liệu giao tác

TID	A	B	C	D	E
T1	0	12	2	0	2
T2	0	12	0	2	1
T3	2	0	1	0	1
T4	1	0	0	2	1
T5	0	0	4	0	2
T6	1	2	0	0	0
T7	0	20	0	2	1
T8	3	0	25	6	1
T9	1	2	0	0	0
T10	0	0	16	0	1

Bảng 2.2. Bảng lợi ích

Mục	Lợi nhuận ($/đơn vị)
A	3
B	5
C	1
D	3
E	5

Định nghĩ 2.4. Cho hàm lợi ích $f(x, y)$. Lợi ích của mục i_p tại giao tác T_q, ký hiệu $u(i_p, T_q)$ là giá trị của hàm $f(x, y)$ tại $o(i_p, T_q)$ và $s(i_p)$, tức là $u(i_p, T_q) = f(o(i_p, T_q), s(i_p))$.

Định nghĩ 2.5. Cho tập mục X chứa trong giao tác T_q. Lợi ích của tập mục X tại giao tác T_q, ký hiệu $u(X, T_q)$, là tổng lợi ích của tất cả các mục i_p thuộc X tại giao tác T_q , tức là

$$u(X, T_q) = \sum_{i_p \in X \subseteq T_q} u(i_p, T_q).$$

Ký hiệu db_X là tập các giao tác chứa tập mục X trong cơ sở dữ liệu DB, tức là:

$$db_X = \{T_q | X \subseteq T_q, T_q \in DB\}.$$

Định nghĩ 2.6. Lợi ích của tập mục X trong cơ sở dữ liệu DB, ký hiệu $u(X)$, là tổng lợi ích của tập mục X tại các giao tác thuộc db_X, tức là:

$$u(X) = \sum_{T_q \in db_X} u(X, T_q) = \sum_{T_q \in db_X} \sum_{i_p \in X} u(i_p, T_q).$$

Ví dụ, trong cơ sở dữ liệu Bảng 2.1 và Bảng 2.2, $u(B, T_2) = 12.5 = 60$. Xét $X = \{B, D\}$, $u(X, T_2) = u(B, T_2) + u(D, T_2) = 66$. Ta có $db_X = \{T_2, T_7\}$, $u(X) = u(X, T_2) + u(X, T_7) = 172$.

Định nghĩ 2.7. Cho ngưỡng lợi ích $minutil(> 0)$ và xét tập mục X, X được gọi là *tập mục lợi ích cao* nếu $u(X) \geq minutil$. Trường hợp ngược lại, X được gọi là *tập mục lợi ích thấp.*

Định nghĩ 2.8. Cho cơ sở dữ liệu giao tác DB và ràng buộc lợi ích $minutil$, bài toán khai phá tập mục lợi ích cao là việc tìm tập HU tất cả các tập mục lợi ích cao, tức là tập $HU = \{X | X \subseteq I, u(X) \geq minutil\}$.

Dễ thấy tính chất Apriori của tập mục thường xuyên không còn đúng với ràng buộc lợi ích. Chẳng hạn, trong cơ sở dữ liệu Bảng 2.1, ta có $u(BC) = 62 < 72 = u(BCE)$, trong khi đó $u(BC) = 62 > 0 = u(BCD)$.

Phần tiếp theo sau đây trình bày nội dung cơ bản của thuật toán IM [15] khai phá tập mục thường xuyên nhờ cấu trúc cây COFI-tree.

3. KHAI PHÁ TẬP MỤC THƯỜNG XUYÊN TRÊN CẤU TRÚC CÂY COFI-TREE

Năm 2003, nhóm tác giả Mohammad El-Hajj và Osmar R. Zaiane ở đại học Alberta Edmonton, Canada đề xuất thuật toán IM (Inverted Matrix) [15] khai phá tập mục thường xuyên trong các cơ sở dữ liệu lớn.

Thuật toán IM gồm 2 giai đoạn. Giai đoạn thứ nhất, thuật toán tổ chức lại dữ liệu thành ma trận và lưu ma trận dữ liệu đó lên bộ nhớ ngoài của máy tính. Giai đoạn thứ hai thuật toán khai phá ma trận dữ liệu này nhờ cấu trúc cây COFI-tree (Co-Occurrence Frequent Item tree) cho từng mục dữ liệu [13, 14].

Giai đoạn thứ nhất, thuật toán IM tổ chức dữ liệu trên đĩa thành hai phần: phần chỉ số và mảng giao tác. Phần chỉ số chứa các mục dữ liệu và độ hỗ trợ của nó. Mảng giao tác là tập các dòng mà mỗi dòng tương ứng với một mục dữ liệu trong phần chỉ số. Xây dựng ma trận giao tác cần 2 lần duyệt cơ sở dữ liệu. Duyệt lần thứ nhất, thuật toán đếm độ hỗ trợ của từng mục dữ liệu, sau đó sắp xếp các mục dữ liệu theo thứ tự tăng dần của độ hỗ trợ của chúng và tạo ra phần chỉ số. Duyệt cơ sở dữ liệu lần thứ hai, mỗi giao tác được đọc ra, sắp các mục dữ liệu của nó theo thứ tự tăng dần của độ hỗ trợ và lưu vào mảng giao tác.

Giai đoạn thứ hai, thuật toán khai phá ma trận dữ liệu (mảng giao tác) nhờ cấu trúc cây COFI-tree. Thuật toán duyệt phần chỉ số theo thứ tự tăng dần của độ hỗ trợ, bỏ qua các mục dữ liệu không thỏa mãn ngưỡng độ hỗ trợ, với mỗi mục dữ liệu thường xuyên, thuật toán đọc các giao tác chứa nó trong mảng giao tác và xây dựng cây COFI-tree chứa các giao tác này, sau đó khai phá cây COFI-tree để tìm ra các tập mục thường xuyên. Sau khi khai phá xong, thuật toán loại bỏ cây này và lại làm tương tự cho mục dữ liệu tiếp theo.

Cây COFI-tree cho một mục dữ liệu chứa các mục dữ liệu cùng xuất hiện với nó và có độ hỗ trợ lớn hơn hoặc bằng độ hỗ trợ của nó. Cây có bảng đầu mục chứa các mục dữ liệu theo thứ tự tăng dần của độ hỗ trợ (thứ tự trong phần chỉ số). Mỗi mục trong bảng đầu mục chứa 3 trường: tên mục dữ liệu, độ hỗ trợ địa phương (số lần xuất hiện của mục dữ liệu trong cây COFI-tree) và con trỏ trỏ đến nút đầu tiên biểu diễn mục dữ liệu này trong cây. Một danh sách liên kết được duy trì giữa các nút cùng tên để thuận lợi cho quá trình khai phá. Mỗi nút của cây COFI-tree có 4 trường: tên mục dữ liệu, 2 biến s và p (biến s biểu diễn độ hỗ trợ của nút, biến p cho biết số lần nút đó đã tham gia tạo mẫu), con trỏ (trỏ đến nút tiếp theo cùng tên trên cây). Chi tiết về cây COFI-tree có thể tham khảo trong [13, 14].

Phần tiếp theo sau đây, chúng tôi đề xuất thuật toán mới khai phá tập mục lợi ích cao trong các cơ sở dữ liệu lớn dựa trên ý tưởng của thuật toán IM. Thuật toán sử dụng khái niệm lợi ích TWU do Y.Liu [10] đề xuất để rút gọn tập các ứng viên.

4. THUẬT TOÁN KHAI PHÁ TẬP MỤC LỢI ÍCH CAO TRONG CƠ SỞ DỮ LIỆU LỚN

Trong [10], Y. Liu đưa ra khái niệm lợi ích của giao tác và lợi ích của tập mục tính theo lợi ích các giao tác chứa nó.

Định nghĩ 4.1. (Lợi ích giao tác-Transaction Utility) Tổng lợi ích của tất cả các mục có

mặt trong giao tác T_q gọi là lợi ích của giao tác T_q và ký hiệu là $tu(T_q)$,

$$tu(T_q) = \sum_{i_p \in T_q} u(i_p, T_q).$$

Định nghĩ 4.2. (Transaction Weighted Utility-TWU) Lợi ích theo giao tác TWU của tập mục X, ký hiệu $twu(X)$ là tổng lợi ích của tất cả các giao tác chứa X trong cơ sở dữ liệu, $twu(X) = \sum_{T_q \in DB \wedge X \subseteq T_q} tu(T_q)$.

Ví dụ, trong cơ sở dữ liệu Bảng 2.1 và Bảng 2.2, $tu(T_2) = 12.5 + 2.3 + 1.5 = 71$.

Xét $X = DE, db_X = \{T_2, T_4, T_7, T_8\}, twu(X) = tu(T_2) + tu(T_4) + tu(T_7) + tu(T_8) = 253$.

Nhận xét: Vì $u(X, T_q) \leq tu(T_q)$ nên $u(X) = \sum_{T_q \in DB \wedge X \subseteq T_q} u(X, T_q) \leq \sum_{T_q \in DB \wedge X \subseteq T_q} tu(T_q) = twu(X)$.

Có thể coi $twu(X)$ như là cận trên của $u(X)$. Với ngưỡng lợi ích *minutil*, nếu X là tập mục lợi ích cao thì X cũng là tập mục lợi ích TWU cao vì $twu(X) \geq u(X) > minutil$, ngược lại, nếu X là tập mục lợi ích TWU thấp thì X cũng là tập mục lợi ích thấp.

Theo [10], ràng buộc lợi ích TWU có tính chất phản đơn điệu: Mọi tập mục cha của tập mục lợi ích TWU thấp cũng là tập mục lợi ích TWU thấp. Do vậy, nếu X là tập mục lợi ích TWU thấp, $twu(X) < minutil$, thì tập X và mọi tập cha của X đều là tập mục lợi ích thấp, có thể loại bỏ chúng trong quá trình khai phá tập mục lợi ích cao.

Dựa trên ý tưởng của thuật toán IM khai phá tập mục thường xuyên, chúng tôi đề xuất thuật toán mới khai phá tập mục lợi ích cao trong cơ sở dữ liệu lớn, gọi là thuật toán COUI-Mine2 (Co-Occurrence Utility Items Mine). Thuật toán gồm hai giai đoạn chính. Giai đoạn thứ nhất xây dựng mảng giao tác chứa đầy đủ thông tin để khai phá tập mục lợi ích cao, mảng giao tác này lưu ở bộ nhớ ngoài của máy tính và giai đoạn thứ hai khai phá mảng giao tác để tìm các tập mục lợi ích cao nhờ cấu trúc dữ liệu gọi là cây COUI-tree (Co-Occurrence Utility Items tree).

4.1. Xây dựng mảng giao tác

Thuật toán tổ chức dữ liệu trên đĩa thành hai phần, phần chỉ số và mảng giao tác. Mỗi mục của phần chỉ số chứa thông tin của một mục dữ liệu gồm 5 trường: tên mục dữ liệu, tổng số lượng, lợi nhuận của một đơn vị, độ hỗ trợ và lợi ích TWU của mục dữ liệu đó. Trong phần chỉ số, các mục dữ liệu được sắp xếp tăng dần theo độ hỗ trợ của chúng. Mảng giao tác là tập các dòng mà mỗi dòng tương ứng với một mục dữ liệu trong phần chỉ số. Mỗi phần tử trong mảng giao tác chứa 4 trường: số lượng mục dữ liệu trong giao tác, lợi ích của giao tác, địa chỉ (dòng, cột) trỏ đến phần tử biểu diễn mục dữ liệu tiếp theo trong cùng một giao tác, nếu đã đến mục dữ liệu cuối cùng của giao tác thì địa chỉ này là rỗng.

Thuật toán cần duyệt cơ sở dữ liệu hai lần. Lần duyệt thứ nhất, thuật toán tính lợi ích các giao tác, số lần xuất hiện (độ hỗ trợ), tổng số lượng, lợi ích theo giao tác TWU của từng mục dữ liệu. Sau đó thuật toán sắp xếp các mục dữ liệu theo thứ tự tăng dần của độ hỗ trợ của chúng và tạo ra phần chỉ số. Lợi ích của các giao tác được lưu lại để sử dụng cho việc đưa các giao tác của cơ sở dữ liệu vào mảng giao tác trong lần duyệt tiếp theo.

Lần duyệt thứ hai, mỗi giao tác được đọc ra, sắp xếp các mục dữ liệu của nó theo thứ tự tăng dần của độ hỗ trợ và đưa danh sách các mục dữ liệu đã sắp đó vào mảng giao tác như sau: trong phần chỉ số, xác định vị trí của mục dữ liệu thứ nhất, tại dòng tương ứng trong mảng giao tác tìm ô (phần tử) còn trống đầu tiên, lưu vào ô này số lượng mục dữ liệu, lợi ích của giao tác và địa chỉ (dòng và cột) của ô trong mảng biểu diễn cho mục dữ liệu thứ hai. Tại ô biểu diễn cho mục dữ liệu thứ hai lại chứa địa chỉ của ô biểu diễn cho mục dữ liệu thứ ba,... Thuật toán làm như vậy cho đến mục dữ liệu cuối cùng của danh sách. Tại ô biểu diễn cho mục dữ liệu cuối cùng, địa chỉ cho mục dữ liệu tiếp theo là rỗng.

Mảng giao tác được tạo ra và lưu ở bộ nhớ ngoài của máy tính. Quá trình khai phá tập mục lợi ích cao ở giai đoạn sau thực hiện trên mảng giao tác này.

Bảng 4.1. Lợi ích các giao tác của cơ sở dữ liệu Bảng 2.1 và Bảng 2.2

TID	A	B	C	D	E	tu
T1	0	12	2	0	2	72
T2	0	12	0	2	1	71
T3	2	0	1	0	1	12
T4	1	0	0	2	1	14
T5	0	0	4	0	2	14
T6	1	2	0	0	0	13
T7	0	20	0	2	1	111
T8	3	0	25	6	1	57
T9	1	2	0	0	0	13
T10	0	0	16	0	1	21
Tổng	8	48	48	12	10	398

Bảng 4.2

Mục dữ liệu	Số lượng	Độ hỗ trợ	*twu*
A	8	5	109
B	48	5	280
C	48	5	176
D	12	4	253
E	10	8	372

Bảng 4.3

Vị trí	Mục dữ liệu	Số lượng	Lợi nhuận/ đơn vị	Độ hỗ trợ	*twu*
1	D	12	3	4	253
2	A	8	3	5	109
3	B	48	5	5	280
4	C	48	1	5	176
5	E	10	5	8	372

Bảng 4.2: Số lượng, *twu*, độ hỗ trợ của các mục.

Bảng 4.3: Phần chỉ số của mảng giao tác.

Ta minh họa xây dựng mảng giao tác qua xét cơ sở dữ liệu Bảng 2.1 và Bảng 2.2, ngưỡng lợi ích bằng 30% của tổng lợi ích, $minutil = 30\% \times 398 = 119,4$.

Duyệt cơ sở dữ liệu lần thứ nhất, tính được lợi ích của các giao tác (Bảng 4.1), tổng số lượng, số lần xuất hiện, lợi ích TWU của từng mục dữ liệu (Bảng 4.2). Sắp xếp các mục theo thứ tự tăng dần của độ hỗ trợ và xây dựng phần chỉ số (Bảng 4.3).

Duyệt cơ sở dữ liệu lần thứ hai, với mỗi giao tác, sắp các mục dữ liệu theo thứ tự tăng dần của độ hỗ trợ và lưu vào mảng giao tác. Bảng 4.4 minh họa các giao tác đã sắp xếp lại các mục dữ liệu. Với giao tác thứ nhất, $T_1 = (B : 12, C : 2, E : 2)$, trong phần chỉ số mục B tìm thấy ở vị trí số 3, mục C ở vị trí 4, mục E ở vị trí 5, do đó 3 ô biểu diễn cho 3 mục này nằm tương ứng ở dòng 3, dòng 4 và dòng 5 trong mảng giao tác. Tìm ô đầu tiên còn trống trên dòng 3 nhận được ô địa chỉ $[3, 1]$, ô này sẽ lưu các thông tin của mục B trong giao tác

và địa chỉ của ô biểu diễn cho mục C tiếp theo. Ô đầu tiên còn trống trên dòng 4 có địa chỉ $[4,1]$, do đó tại ô địa chỉ $[3,1]$ sẽ lưu số lượng của mục B là 12, lợi ích của giao tác $tu = 72$ và địa chỉ $[4,1]$ của ô biểu diễn cho mục C. Tương tự, tại ô địa chỉ $[4,1]$ lưu số lượng của mục C là 2, lợi ích giao tác $tu = 72$ và địa chỉ $[5,1]$ của ô biểu diễn cho mục E. Tại ô địa chỉ $[5,1]$ lưu số lượng của mục E là 2 và lợi ích giao tác $tu = 72$. Vì E là mục cuối của giao tác nên phần địa chỉ của ô tiếp theo là rỗng. Các giao tác khác của cơ sở dữ liệu được thực hiện tương tự. Hình 4.2 là mảng giao tác biểu diễn cơ sở dữ liệu Bảng 2.1 và Bảng 2.2.

Bảng 4.4. Các giao tác đã sắp xếp lại các mục dữ liệu và lợi ích của nó

TID	D	A	B	C	E	tu
T1	0	0	12	2	2	72
T2	2	0	12	0	1	71
T3	0	2	0	1	1	12
T4	2	1	0	0	1	14
T5	0	0	0	4	2	14
T6	0	1	2	0	0	13
T7	2	0	20	0	1	111
T8	6	3	0	25	1	57
T9	0	1	2	0	0	13
T10	0	0	0	16	1	21

Vị trí	Chỉ số	Mảng giao tác								
		1	2	3	4	5	6	7	8	9
1	**D**, 12, 3 4, 253	2, 71 [3,2]	2, 14 [2,2]	2, 111 [3,4]	6, 57 [2,4]					
2	**A**, 8, 3 5, 109	2, 12 [4,2]	1,14 [5,4]	1,13 [3,3]	3, 57 [4,4]	1, 13 [3,5]				
3	**B**, 48, 5 5, 280	12, 72 [4,1]	12, 71 [5,2]	2, 13 [Ø,Ø]	20,111 [5,6]	2, 13 [Ø,Ø]				
4	**C**, 48, 1 5, 176	2, 72 [5,1]	1, 12 [5,3]	4, 14 [5,5]	25, 57 [5,7]	16, 21 [5,8]				
5	**E**, 10, 5 8, 372	2, 72 [Ø,Ø]	1, 71 [Ø,Ø]	1, 12 [Ø,Ø]	1,14 [Ø,Ø]	2, 14 [Ø,Ø]	1, 111 [Ø,Ø]	1, 57 [Ø,Ø]	1, 21 [Ø,Ø]	

Hình 4.2. Mảng giao tác biểu diễn cơ sở dữ liệu Bảng 2.1 và Bảng 2.2

Ta thấy rằng các giao tác của cơ sở dữ liệu Bảng 2.1 và Bảng 2.2 đã được lưu vào mảng giao tác. Khai phá tập mục lợi ích cao với các ngưỡng lợi ích khác nhau đều thực hiện được trên mảng này mà không cần phải xây dựng lại.

Thuật toán xây dựng mảng giao tác có thể mô tả như sau.

Thuật toán 1. Xây dựng mảng giao tác

Input: Cơ sở dữ liệu giao tác DB.

Output: Mảng giao tác biểu diễn cơ sở dữ liệu DB lưu ở bộ nhớ ngoài để khai phá lợi ích cao.

Method:

1. for each $T \in DB$ // duyệt cơ sở dữ liệu lần thứ nhất.
2. begin

3. - Tính lợi ích giao tác $tu(T)$;
4. - Tính độ hỗ trợ, tổng số lượng, lợi ích TWU của từng mục dữ liệu;
5. end;
6. Sắp xếp các mục dữ liệu theo thứ tự tăng dần của độ hỗ trợ của chúng;
7. Dựa trên danh sách các mục dữ liệu đã sắp, xây dựng phần chỉ số của mảng giao tác;
8. for each $T \in DB$ // duyệt cơ sở dữ liệu lần thứ hai.
9. begin
10. Sắp các mục dữ liệu trong T theo thứ tự của phần chỉ số, nhận được danh sách $TList = (A_1 : s_1, A_2 : s_2, ..., A_k : s_k)$; // s_i là số lượng của mục A_i trong giao tác T.
11. Xác định địa chỉ $[d_1, c_1]$ lưu thông tin mục A_1 trong mảng giao tác;
12. for $i := 1$ to $k - 1$ do // xét từng mục dữ liệu trong danh sách $TList$.
13. begin
14. - Xác định địa chỉ $[d_{i+1}, c_{i+1}]$ lưu thông tin mục A_{i+1} trong mảng;
15. - Lưu tại $[d_i, c_i]$: số lượng s_i, lợi ích giao tác $tu(T)$, địa chỉ $[d_{i+1}, c_{i+1}]$;
16. end;
17. Lưu tại $[d_k, c_k]$: số lượng s_k, lợi ích giao tác $tu(T)$, địa chỉ rỗng $[\emptyset, \emptyset]$;
18. end;

4.2. Khai phá mảng giao tác

Xét lần lượt các mục dữ liệu từ trên xuống trong phần chỉ số của mảng giao tác. Với mỗi mục dữ liệu, nếu lợi ích theo giao tác TWU của nó lớn hơn ngưỡng *minutil* thì đọc ra các giao tác chứa nó trong mảng giao tác, từ các giao tác này, xây dựng một cấu trúc dữ liệu gọi là cây COUI-tree, khai phá cây COUI-tree để tìm các mẫu (tập mục) lợi ích cao, sau khi khai phá xong, loại bỏ cây này và xét mục dữ liệu tiếp theo.

Cây COUI-tree của mục dữ liệu nào thì mục đó là nhãn của nút gốc của cây. Cây có bảng đầu mục, mỗi mục trong Bảng đầu mục chứa 3 trường: tên mục dữ liệu, biến twu lưu lợi ích theo giao tác của mục dữ liệu và con trỏ trỏ đến nút đầu tiên trên cây có nhãn là mục dữ liệu này. Bảng đầu mục chứa các mục dữ liệu tham gia xây dựng cây và được sắp thứ tự theo thứ tự trong phần chỉ số của mảng giao tác. Mỗi nút của cây COUI-tree gồm 4 trường: tên mục dữ liệu, biến twu lưu lợi ích của giao tác chứa mục dữ liệu, mảng lưu số lượng các mục dữ liệu tương ứng trên đường đi từ nút đó đến nút gốc của cây và con trỏ (trỏ đến nút tiếp theo cùng nhãn trên cây hoặc là null nếu không có). Các giao tác đọc ra từ mảng giao tác được chèn vào cây COUI-tree như sau. Giả sử các mục của giao tác là $[x|L]$ với x là mục dữ liệu đầu và L là phần còn lại của giao tác. Ta kiểm tra xem mục x có là nhãn của nút con nào của nút gốc không? Nếu có thì điều chỉnh phù hợp nút con này, ngược lại, thêm cho nút gốc một nút con mới với nhãn là x. Tiếp theo, coi nút hiện thời là nút gốc và lặp lại tương tự với các mục dữ liệu tiếp theo trong giao tác. Khi thêm một nút mới nhãn x vào cây cần duy trì danh sách liên kết giữa nút này với mục dữ liệu x tương ứng trong Bảng đầu mục.

Ta minh họa thuật toán qua xét mảng giao tác Hình 4.2. Khai phá tập mục lợi ích cao trên mảng giao tác Hình 4.2 cần xây dựng các cây COUI-tree cho các mục D, B và C, gọi các cây này là D-COUI-tree, B-COUI-tree và C-COUI-tree. Không xây dựng cây A-COUI-tree cho mục A vì $twu(A) = 109 < minutil$, mọi tập mục chứa A không thể là tập mục lợi ích

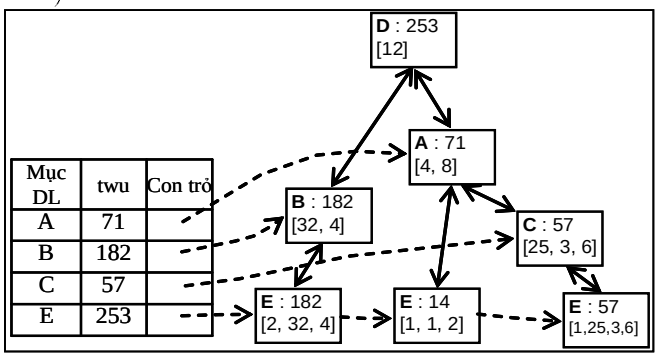
cao, không xây dựng cây E-COUI-tree vì cây này chỉ có gốc nhãn là E. Cây D-COUI-tree chứa các mục dữ liệu cùng xuất hiện với mục D, cây B-COUI-tree chứa các mục dữ liệu cùng xuất hiện với mục B nhưng không chứa mục D và A, cây C-COUI-tree chứa các mục dữ liệu cùng xuất hiện với mục C nhưng không chứa mục D, A và B.

Sau đây minh họa quá trình xây dựng và khai phá cây D-COUI-tree.

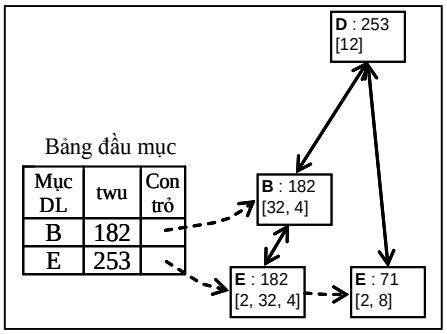
Xây dựng cây D-COUI-tree

Phần chỉ số của mảng giao tác cho biết độ hỗ trợ của mục D bằng 4, tức là có 4 giao tác chứa D. Bắt đầu từ các ô ở dòng 1 của mảng, theo dây chuyền của các ô trong mảng, lấy được 4 giao tác chứa D: đầu tiên, ô $[1,1]$ xác định mục D với số lượng là 2, ô này trỏ đến địa chỉ $[3,2]$, địa chỉ $[3,2]$ xác định mục B với số lượng là 12, địa chỉ $[3,2]$ chứa địa chỉ mục tiếp theo là $[5,2]$, địa chỉ $[5,2]$ xác định mục E với số lượng là 1 và địa chỉ mục tiếp theo là rỗng, tức là đã hết giao tác. Cuối cùng ta nhận được giao tác $T_1 = (D:2, B:12; E:1)$ và lợi ích của giao tác là $tu(T_1) = 71$. Tương tự, bắt đầu từ các ô $[1,2], [1,3]$ và $[1,4]$ xác định được 3 giao tác $T_2 = (D:2, A:1, E:1)$ với $tu(T_2) = 14$, $T_3 = (D:2, B:20, E:1)$ với $tu(T_3) = 111$ và $T_4 = (D:6, A:3, C:25, E:1)$ với $tu(T_4) = 57$. Mỗi giao tác được đọc ra và chèn vào cây D-COUI-tree, lưu ý là phải điều chỉnh biến *twu* của bảng đầu mục của cây cho phù hợp. Hình 4.3 biểu diễn cây D-COUI-tree.

Khai phá cây D-COUI-tree. Khai phá cây D-COUI-tree tìm được các tập mục lợi ích cao có chứa mục D. Quá trình khai phá như mô tả trong thuật toán D-COUI-tree trình bày trong [4]. Hai mục A và C có $twu < minutil$, do đó bước tỉa cây sẽ tỉa 2 mục này (Hình 4.4 là cây D-COUI-tree sau khi tỉa).



Hình 4.3. Cây D-COUI-tree



Hình 4.4. Cây D-COUI-tree sau khi tỉa mục A và C

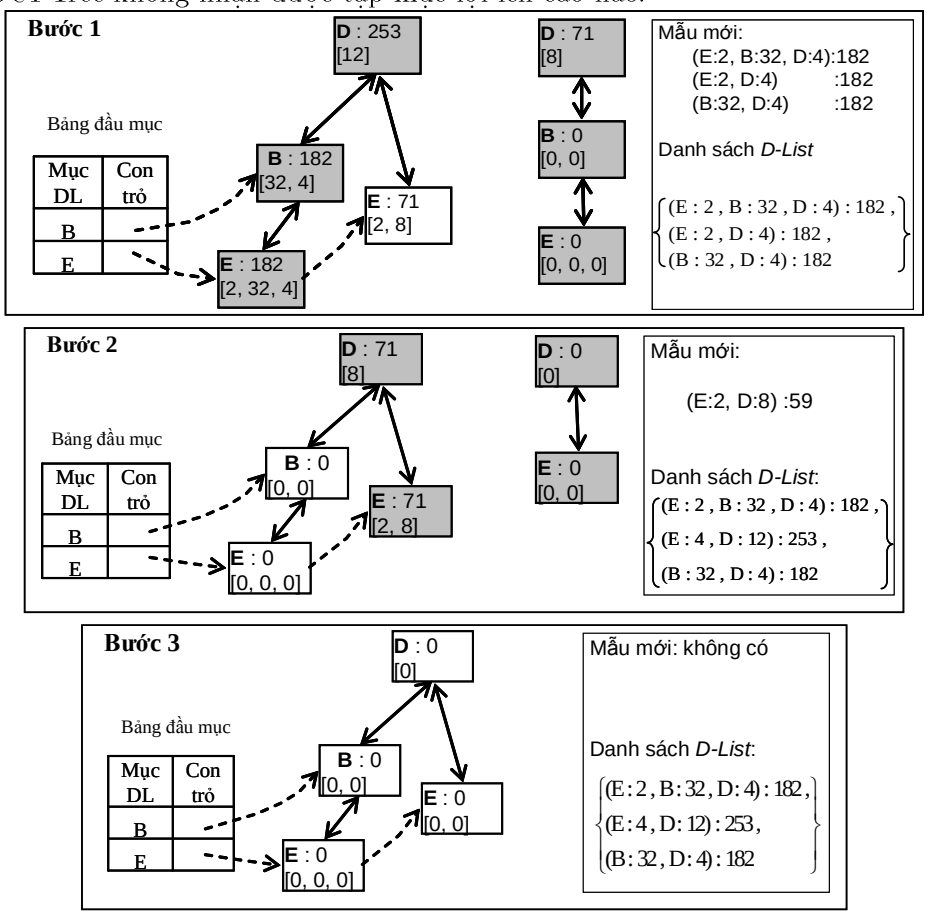
Xét mục E: Từ con trỏ của mục E trên Bảng đầu mục tìm được 2 nút trên cây có nhãn E. Đường đi từ nút E thứ nhất lên nút gốc xác định mẫu $(E:2, B:32, D:4):182$. Kết nạp mẫu này cùng các mẫu con có chứa D của nó vào danh sách $D-List$ chứa các ứng viên của mục D, $D-List = \{(E:2, B:32, D:4):182; (E:2, D:4):182; (B:32, D:4):182\}$. Điều chỉnh biến *twu* và mảng số lượng của các nút nhãn E, B và D trên đường đi: biến *twu* giảm đi 182, mảng số lượng được trừ đi tương ứng (bước 1).

Đường đi từ nút E thứ hai lên nút gốc xác định mẫu $(E:2, D:8):71$. Kết nạp mẫu này vào $D-List$. Trong $D-List$ đã có mẫu $(E:2, D:4):182$ nên điều chỉnh mẫu này thành $(E:4, D:12):253$ (bước 2).

Xét mục B tiếp theo, nút nhãn B trên cây có $twu = 0$, do vậy không sinh ra mẫu nào nữa (bước 3). Hình 4.5 minh họa các bước khai phá cây D-COUI-Tree.

Kết thúc khai phá cây D-COUI-Tree, nhận được danh sách $D-List$. Duyệt danh sách $D-List$, với mỗi tập ứng viên $X \in D-List$ tính lợi ích thực sự của nó: $u(EBD) = 182$, $u(BD) = 172$, $u(ED) = 56$. So sánh với $minutil = 119{,}4$ có hai tập mục lợi ích cao tìm được là EBD và BD, tập $HU = \{EBD(182), BD(172)\}$.

Thuật toán xóa cây D-COUI-Tree và danh sách $D-List$, tiếp tục xây dựng cây B-COUI-Tree và C-COUI-Tree. Khai phá cây B-COUI-Tree nhận được $EB(240)$ và $B(240)$. Khai phá cây C-COUI-Tree không nhận được tập mục lợi ích cao nào.



Hình 4.5. Các bước khai phá cây D-COUI-Tree

Hoàn thành khai phá mảng giao tác, thuật toán tìm ra tập các tập mục lợi ích cao $HU = \{EBD(182), BD(172), EB(240), B(240)\}$.

Sau đây là thuật toán tạo và khai phá các cây COUI-tree.

Thuật toán 2. Khai phá mảng giao tác

Input: Mảng giao tác, hàm lợi ích, ngưỡng lợi ích *minutil*.
Output: Tập HU chứa tất cả các tập mục lợi ích cao của cơ sở dữ liệu DB.
Method:

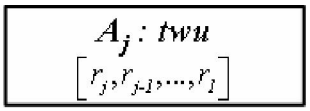
1. Xét lần lượt từ trên xuống của phần chỉ số, lấy mục A có $twu(A) \geq minutil$;
2. repeat
3. If $twu(A) < minutil$ then goto 15; // bỏ qua mục dữ liệu này.
4. Tính lợi ích của mục A; // từ số lượng và lợi nhuận lưu trong phần chỉ số.
5. If $twu(A) > minutil$ then $HU = HU \cup \{A\}$;
6. Đọc độ hỗ trợ s và xác định vị trí dòng d chứa mục A trong mảng giao tác;
7. Tạo nút gốc R có nhãn A, gán twu bằng 0, số lượng bằng 0;
8. for $i := 1$ to s do // xét lần lượt s ô trên dòng d của mảng giao tác.
9. begin
10. Theo dây chuyền từ ô $[d, i]$, xác định giao tác $T = (A_1 : s_1, A_2 : s_2, ..., A_k : s_k)$ và lợi ích $tu(T)$; // A_1 là mục dữ liệu A.
11. Gọi hàm $insert\_tree(T, R)$ để chèn T vào (A)-COUI-tree;
12. end;
13. Gọi hàm MineCOUI-tree (A);// hàm khai phá cây (A)-COUI-tree.
14. Xóa cây (A)-COUI-tree;
15. Lấy mục dữ liệu A tiếp theo trong phần chỉ số;
16. Until (A là mục dữ liệu cuối cùng trong phần chỉ số của mảng giao tác);
17. Tính lợi ích của mục A; // tính lợi ích mục dữ liệu cuối cùng trong phần chỉ số.
18. if $u(A) > minutil$ then $HU = HU \cup \{A\}$;
19. Return HU;

Giải thích thực hiện của hàm $insert\_tree(T, R)$: giả sử giao tác T là $[x|L]$, ở đó x là mục dữ liệu đầu và L là phần còn lại của giao tác. Nếu R có nút con N nhãn x thì điều chỉnh các trường của nút N, ngược lại, tạo nút N mới là nút con của nút R và gán nhãn của nút N là x, đặt giá trị phù hợp cho các trường của nút N. Điều chỉnh bảng đầu mục của cây: bổ sung mục x vào bảng đầu mục (nếu chưa có), tăng biến twu tương ứng với mục x lên $tu(T)$, nếu tạo mới nút N thì bổ sung đường liên kết của các nút cùng nhãn x đến nút N này. Nếu L khác rỗng thì gọi đệ quy hàm $insert\_tree(L, N)$.

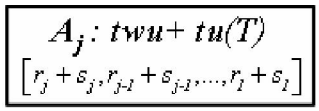
Xét giao tác là $T = (A_1 : s_1, A_2 : s_2, ..., A_k : s_k)$ với lợi ích $tu(T)$ được đọc ra từ mảng giao tác, ở đó cặp $A_j : s_j$ biểu diễn tên mục dữ liệu và số lượng của mục đó trong giao tác. Giả sử ta đã cất $j-1$ mục đầu lên cây, bây giờ xét mục A_j tại nút N. Nút N có nhãn A_j, đường đi từ nút N lên nút gốc là $A_j, A_{j-1}, ..., A_1$, ở đó A_1 là mục A. Nút N có trường twu và mảng số lượng các mục tương ứng với các mục trên đường đi từ nút N lên nút gốc.

Nếu N là nút đã có của cây với mảng số lượng là $[r_j, r_{j-1}, ..., r_1]$, thay đổi tại nút N như sau: $N.twu := N.twu + tu(T)$, cộng tương ứng dãy số $s_j, s_{j-1}, ..., s_1$ vào mảng $[r_j, r_{j-1}, ..., r_1]$

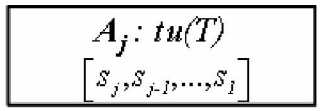
và nhận được mảng $[r_j + s_j, r_{j-1} + s_{j-1}, ..., r_1 + s_1]$ (Hình 4.6 a và 4.6b).



Hình 4.6a. Nút N



Hình 4.6b. Nút N sau khi điều chỉnh



Hình 4.7. Nút N mới tạo.

Nếu N là nút mới được tạo ra, ta thiết đặt tại nút N như sau: $N.twu := tu(T)$, mảng số lượng các mục là $[s_j, s_{j-1}, ..., s_1]$ (Hình 4.7).

Function: MineCOFI-tree (A); // hàm thực hiện khai phá cây (A)-COFI-tree.
Method:
1. $(A) - List := \emptyset$; // khởi tạo danh sách các mẫu ứng viên chứa mục A là rỗng.
2. Tỉa cây: duyệt bảng đầu mục của (A)-COFI-tree, tỉa các mục dữ liệu có $twu < minutil$; // tỉa trên bảng đầu mục và các nút tương ứng trên cây.
3. for each (mục dữ liệu B của bảng đầu mục của cây (A)-COFI-tree)// từ dưới lên.
4. for each (nút N trên cây (A)-COFI-tree có nhãn B) //tìm theo con trỏ
5. begin
6. - Đọc biến twu và mảng số lượng các mục của nút N;
7. - Xác định mẫu X từ đường đi từ nút N lên nút gốc của cây;
8. - Kết nạp X và các mẫu con của X có chứa mục A vào $(A) - List$;
9. - Điều chỉnh biến twu và mảng số lượng các mục của các nút trên đường đi từ nút N lên nút gốc của cây;
10. end; // hoàn thành khai phá cây (A)-COFI-tree).
11. for each $Y \in (A) - List$ // duyệt các mẫu ứng viên của danh sách $(A) - List$.
12. begin
13. - Tính lợi ích $u(Y)$ của mẫu Y;
14. - if $u(Y) > minutil$ then $HU := HU \cup \{Y\}$;
15. end;
16. Return HU;

5. ĐÁNH GIÁ THUẬT TOÁN VÀ KẾT LUẬN

Thuật toán được thử nghiệm trên một số cơ sở dữ liệu giao tác được tạo bằng phương pháp tạo số ngẫu nhiên [16]. Từ kết quả thử nghiệm và phân tích thuật toán, có thể nhận xét thuật toán có những ưu điểm sau:

- Thuật toán được phát triển từ thuật toán COUI-tree trong [4] bằng cách thay cây UP-tree trong thuật toán COUI-tree bởi ma trận giao tác, lưu tại bộ nhớ ngoài. Cách phát triển này làm cho thuật toán có thể khai phá trên những tập dữ liệu rất lớn.

- Xây dựng mảng giao tác chỉ cần duyệt cơ sở dữ liệu hai lần. Khai phá mảng giao tác nhờ một cấu trúc dữ liệu nhỏ là cây COUI-tree theo phương pháp không đệ quy, do đó giảm thời gian tính toán và sử dụng ít bộ nhớ.

- Sau khi chuyển đổi cơ sở dữ liệu sang mảng giao tác, có thể khai phá tập mục lợi ích cao với các ngưỡng lợi ích khác nhau mà không cần xây dựng lại mảng, tạo ra một lần đầu

và sử dụng được nhiều lần.

- Các cây COUI-tree thực chất là kết quả chiếu của mảng giao tác (cũng là của cơ sở dữ liệu) cho từng mục dữ liệu. Cách làm này đã chia bài toán lớn thành nhiều bài toán nhỏ đơn giản hơn.

- Thuật toán đã sử dụng khái niệm lợi ích TWU của tập mục một cách hiệu quả, nhờ đó giảm đáng kể việc sinh ra các tập mục ứng viên dư thừa.

Với những ưu điểm trên và qua kết quả thử nghiệm cho thấy thuật toán COUI-tree là thuật toán hiệu quả để khai phá các tập mục lợi ích cao trong các cơ sở dữ liệu lớn.

TÀI LIỆU THAM KHẢO

1. Vũ Đức Thi, *Cơ sở dữ liệu - Kiến thức và thực hành*, Nhà xuất bản Thống kê, năm 1997.
2. Nguyễn Thanh Tùng, Khai phá tập mục lợi ích cao trong cơ sở dữ liệu, *Tạp chí Tin học và Điều khiển học* **23** (4) (2007) 364–373.
3. Nguyễn Huy Đức, Khai phá luật kết hợp trong cơ sở dữ liệu lớn, *Kỷ yếu Hội thảo khoa học Quốc gia lần thứ nhất về nghiên cứu cơ bản và ứng dụng CNTT*, Hà Nội, 10/2003.
4. Vũ Đức Thi, Nguyễn Huy Đức, Thuật toán hiệu quả khai phá tập mục lợi ích cao trên cấu trúc dữ liệu cây, *Tạp chí Tin học và Điều khiển học* **24** (3) (2008) 204–216.
5. R. Agrawal and R. Srikant, Fast algorithms for mining association rules, *Proceedings of 20th International Conference on Very Large Databases*, Santiago, Chile, 1994.
6. B. Goethals and M. Zaki, Advances in frequent itemset mining implementations: Introduction to fimi03, *Workshop on Frequent Itemset Mining Implementations (FIMI03) in Conjunction with IEEE-ICDM*, 2003.
7. H. Yao, H. J. Hamilton, and C. J. Butz, A foundational approach to mining itemset utilities from databases, *Proceedings of the 4th SIAM International Conference on Data Mining*, Florida, USA, 2004.
8. H. Yao, H. J. Hamilton, Mining itemsets utilities from transaction databases, *Data and Knowledge Engeneering* **59** (3) (2006).
9. H. Yao, H. J. Hamilton, and L. Geng, A unified framework for utility based measures for mining itemsets, *UBDM06 Philadelphia*, Pennsylvania, USA, August 2006.
10. Y. Liu, W. K. Liao, A. Choudhary, A fast high utility itemsets mining algorithm, *Proc. 1st Intl. Conf. on Utility-Based Data Mining*, Chicago, IL, Aug. 2005 (90–99).
11. G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets, *Proc. IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, Melbourne, FL, Nov. 2003.
12. J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, *Proc. 2000 ACM-SIGMOD Intl. Conf. on Management of Data*, Dallas, TX, May 2000 (1–12).
13. M. El-Hajj, Osmar R. Zaiane, Non recursive generation of frequent k-itemsets from frequent pattern tree representations, *Procceding of 5th International Conference on Data Warehousing and Knowledge Discovery*, DaWak, September 2003.

14. M. El-Hajj and Osmar R. Zaiane, COFI-tree mining: A new approach to pattern growth with reduced candidacy generation,*Procceding 2003 Intl Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD)*, Chicago, Illinois, USA, August 2003.

15. M. El-Hajj and Osmar R. Zaiane, Inverted matrix: Effcient discovery of frequent items in large datasets in the context of interactive mining, *Proc. 2003 Intl Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD)*, Chicago, Illinois, USA, August, 2003 (109–118).

16. IBM Synthetic data:
http://www.almaden.ibm.com/software/quest/Resources/index.shtml , 2004.

Nhận bài ngày 15 - 12 - 2008