Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015
DOI: 10.15625/vap.2015.000208

THUẬT TOÁN KHAI THÁC TẬP PHỔ BIẾN TỪ CƠ SỞ DỮ LIỆU SỐ LƯỢNG CÓ SỰ PHÂN CẤP CÁC MỤC

Nguyễn Duy Hàm[1], Võ Đình Bảy[2], Nguyễn Thị Hồng Minh[3]
[1]Bộ môn Toán Tin học, Trường Đại học An ninh Nhân dân
[2]Khoa Công nghệ Thông tin, Trường Đai học Công nghệ TP.HCM
[3]Khoa Sau đại học, Đại học Quốc gia Hà Nội
duyham@gmail.com, *bayvodinh@gmail.com*, *minhnth@gmail.com*

TÓM TẮT: *Khai thác tập phổ biến để tìm mối quan hệ giữa các item (mục) trong cơ sở dữ liệu (CSDL) là bài toán quan trọng trong khai thác dữ liệu. Bên cạnh khai thác tập phổ biến từ các CSDL truyền thống, khai thác tập phổ biến trên CSDL trọng số và CSDL số lượng đã nhận được nhiều quan tâm từ các nhóm nghiên cứu. Tuy nhiên, các nghiên cứu này mới chỉ khai thác trên các CSDL mà các mục không có mối quan hệ nào với nhau. Trong bài báo này, chúng tôi đề xuất bài toán khai thác tập phổ biến trên CSDL số lượng có sự phân cấp item, đồng thời đề xuất thuật toán để giải quyết bài toán này và áp dụng kĩ thuật diffset hai cấu trúc MByS, MBiS trong lưu trữ tidset của các itemset. Kết quả thực nghiệm cho thấy thuật toán sử dụng cấu trúc MBiS hiệu quả nhất về mặt thời gian xử lý.*

Từ khóa: *CSDL số lượng, CSDL có sự phân cấp mục, tập phổ biến, itemsets.*

I. GIỚI THIỆU

Khai thác tập phổ biến là bài toán quan trọng trong khai thác dữ liệu nói chung. Từ tập phổ biến người ta có thể khai thác luật kết hợp, gom cụm hay phân lớp, .v.v. Do đó, bài toán khai thác tập phổ biến được nhiều nhóm nghiên cứu trên thế giới quan tâm [1-11]. Khai thác tập phổ biến trọng số hữu ich FWUI (frequent weighted utility itemsets) được đề xuất lần đầu tiên năm 2008 [4]. Sau đó Vo và các đồng sự [12] đề xuất sử dụng hướng tiếp cận khai thác theo CSDL chiều dọc với chỉ một lần đọc dữ liệu. Hàm và các đồng sự [9, 10] đề xuất các cấu trúc mới trong khai thác tập phổ biến trên CSDL số lượng, các đề xuất này đã đạt được một số kết quả nhất định. Tuy nhiên các nghiên cứu này chưa đặt các mục vào trong các mối quan hệ khách quan của nó.

Bài toán khai thác luật kết hợp dựa trên khai thác tập phổ biến trên CSDL có sự phân cấp các mục được đề xuất lần đầu tiên năm 1995 bởi Han và các đồng sự trong [5], các tác giả đưa ra định nghĩa về CSDL có nhiều cấp của các item, và đề xuất bài toán khai thác luật kết hợp trên CSDL dạng này với chỉ một ngưỡng hỗ trợ. Trong [6,7] đưa ra các đề xuất về khai thác tập phổ biến với nhiều ngưỡng hỗ trợ khác nhau. Vo và các đồng sự trong [8] đề xuất hướng tiếp cận CSDL chiều dọc với chỉ một lần đọc CSDL cho hiệu quả tốt về thời gian xử lý. Tuy nhiên, cho đến hiện nay, các nghiên cứu trên CSDL có sự phân cấp các mục mới chỉ đề cập đến CSDL nhị phân, chưa quan tâm đến CSDL số lượng với các mục có trọng số, và trên mỗi giao dịch thể hiện số lượng của các mục.

Trong bài báo này, chúng tôi đề xuất bài toán "*Khai thác tập phổ biến trên CSDL số lượng có sự phân cấp các mục*", đồng thời đưa ra thuật toán để giải quyết bài toán này. Đây là bài toán chưa được đặt ra trước đây.

Phần còn lại của bài báo được cấu trúc như sau: Phần II, trình bày các nghiên cứu liên quan. Một số định nghĩa được trình bày trong phần III. Phần IV đưa ra thuật toán khai thác hiệu quả trên CSDL số lượng có sự phân cấp các mục. Kết quả thực nghiệm được trình bày trong phần V. Phần VI trình bày kết luận và hướng phát triển.

II. KIẾN THỨC CƠ BẢN VÀ CÁC NGHIÊN CỨU LIÊN QUAN

A. Khai thác tập phổ biến trên CSDL số lượng

Khan và các đồng sự [4] đưa ra bài toán khai thác tập phổ biến trọng số hữu ích FWUI (frequent weight utility itemsets) từ CSDL số lượng. Nhóm tác giả đề xuất độ đo trọng số hữu ích của giao dịch *twu* (transaction weight utility) và độ hỗ trợ trọng số hữu ích *wus* (weight utility support). Đồng thời đưa ra một "framework" để khai thác FWUI dựa trên các độ đo đã đề xuất.

Theo đó, *twu* của mỗi giao dịch t_k được xác định theo công thức sau:

$$twu(t_k) = \frac{\sum_{i_j \in S(t_k)} w_j \times x_{ki_j}}{S(tk)} \quad (1)$$

Trong đó, x_{ki_j} là số lượng của mục i_j trong giao dịch t_k, w_j là trọng số của mục i_j, và $S(tk)$ là tổng số lượng các phần tử có mặt trong giao dịch t_k.

Tiếp theo, *wus* của mỗi itemset X được tính theo công thức:

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

Vo và các đồng sự [12] đề xuất hướng tiếp cận theo thuật toán Eclat [2] với chỉ một lần đọc dữ liệu, với việc đề xuất cấu trúc MWIT-tree là một mở rộng của IT-tree [2]. Mỗi nút trên WIT-tree gồm 3 thành phần <*tidset*(X), X, *wus*(X)>

Hàm và các đồng sự [9] đề xuất cấu trúc MByS là một cải tiến của cấu trúc *DBV*[3], MByS bao gồm các đoạn byte khác 0 liên tiếp trong biểu diễn tidset của các itemset dưới dạng bit vecto. Đồng thời nhóm đề xuất cấu trúc MByS-tree trong khai thác FWUI với chỉ một lần đọc dữ liệu.

Hàm và các đồng sự [10] đề xuất cấu trúc MBiS là một cải tiến khác của *DB*V[3], MBiS bao gồm các đoạn bit 1 liên tiếp trong biểu diễn tidset của các itemset dưới dạng bit vecto. Đồng thời, nhóm tác giả đề xuất cấu trúc MBiS-tree trong khai thác FWUI với chỉ 1 lần đọc dữ liệu.

B. Khai thác tập phổ biến trên CSDL có sự phân cấp các mục

Han và các đồng sự [5] đề xuất bài toán khai thác tập phổ biến trên CSDL có sự phân cấp các mục và sử dụng hướng tiếp cận Apriori. Đồng thời đề xuất sử dụng chung một ngưỡng hỗ trợ duy nhất cho tất cả các mục như khai thác trên CSDL truyền thống. Cách tiếp cận này không hiệu quả do tốn thời gian đọc CSDL.

Liu và các đồng sự [6] đề xuất một tiếp cận khác với việc khai thác tập phổ biến với nhiều ngưỡng hỗ trợ khác nhau. Cách tiếp cận này khá thực tế, bởi CSDL có sự phân cấp thì các mục cha ở mỗi mức có một giá trị ảnh hưởng khác nhau.

Tseng và các đồng sự [7] đề xuất hướng tiếp cận sử dụng FP-tree với thuật toán FP-Growth trong khai thác tập phổ biến với nhiều ngưỡng hỗ trợ. Cách tiếp cận này khá tốt với chỉ hai lần đọc CSDL, tuy nhiên quá trình duyệt cây FP-tree lại tốn khá nhiều thời gian.

Vo và các đồng sự [8] sử dụng hướng tiếp cận Eclat với việc đề xuất cấu trúc GIT-tree là một mở rộng của IT-tree với chỉ một lần đọc CSDL. Bước đầu tiên thêm các mục cha vào CSDL, bước thứ hai, đọc CSDL để chuyển CSDL sang chiều dọc. Sau đó sử dụng cấu trúc GIT-tree để khai thác tập phổ biến.

Một số nghiên cứu khác trong thời gian gần đây khai thác trên CSDL có sự phân cấp item theo thời gian [13], hay khai thác mẫu phổ biến phân cấp [14] từ đó sinh luật kết hợp phân cấp với một ngưỡng phổ biến theo hai tiếp cận Aprriori hay FP-Growth. Các nghiên cứu này là các trường hợp riêng của bài toán khai thác tập phổ biến trên CSDL nhị phân có sự phân cấp item.

C. CSDL số lượng có sự phân cấp các mục

CSDL số lượng có sự phân cấp các mục là một bộ $DB = <T, I, W, Tr>$, trong đó: $T = \{t_1, t_2, \ldots, t_m\}$ là tập các giao dịch, $I = \{i_1, i_2, \ldots, i_n\}$ là tập các mục, $W = \{w_1, w_2, \ldots, w_n\}$ là tập các trọng số (lợi ích) tương ứng của mỗi mục trong tập các mục I, và H là tập các cây phân cấp thể hiện mối quan hệ giữa các mục. Mỗi giao dịch t_k có dạng $t_k = \{x_{k1}, x_{k2}, \ldots, x_{kn}\}$, trong đó x_{ki} là số nguyên chỉ số lượng mục thứ i trong giao dịch t_k, $k = 1..\ m$, .

Ví dụ 1: cho CSDL số lượng $DB = < T, I, W, Tr >$ như sau:

Tập các mục $I = \{A, B, C, D, E, F\}$

Tập các trọng số $W = \{0.3, 0.2, 0.5, 0.6, 0.9, 0.1\}$ như trong bảng 2

Tập các giao dịch T được cho trong bảng 1 dưới đây:

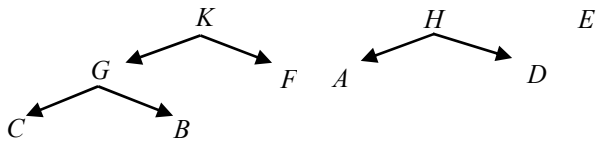
Bảng 1. Các giao dịch

Giao dịch	A	B	C	D	E	F
t_1	1	1	0	2	1	0
t_2	0	1	3	0	1	0
t_3	2	1	0	2	2	2
t_4	3	1	1	0	1	0
t_5	1	2	2	1	3	1
t_6	0	1	1	1	0	1

Bảng 2. Bảng trọng số

Mục	Trọng số
A	0.3
B	0.2
C	0.5
D	0.6
E	0.9
F	0.1

Và tập các cây phân cấp *Tr*



Hình 1. Tập các cây phân cấp *Tr*

Trong đó các kí hiệu *A*, *B*, *C*, *D*, *E*, *F* là đại diện cho tập các mặt hàng theo bảng sau:

Bảng 3. Tên mặt hàng của các mục

Mục	Tên mặt hàng
A	Desktop
B	Ink-jet Printer
C	Laser Printer

D	Notebook
E	Scanner
F	Dot-matrix Printer
G	Non-impact
H	PC
K	Printer

Theo bảng 1, và bảng 2, CSDL *DB* có 6 giao dịch $\{t_1, t_2, t_3, t_4, t_5, t_6\}$ và 6 mục $\{A, B, C, D, E, F\}$, trọng số của các mục tương ứng là {0.3, 0.2, 0.5, 0.6, 0.9, 0.1}. Giao dịch t_1 = {1, 1, 0, 2, 1, 0} có nghĩa là trong giao dịch t_1 có một mục *A* (Desktop), một mục *B* (Ink-jet Printer), hai mục *D* (Notebook), một mục *E* (Scanner), và không có mục *C* (Laser Printer) và mục *F* (Dot-matrix Printer) nào.

Tập $J = \{G, K, H\}$ là tập các mục nút cha của cây phân cấp, là các mục không xuất hiện trong các giao dịch của CSDL *DB*. Tuy nhiên chúng có vai trò nhất định, thể hiện mối quan hệ của các mục trong CSDL. Do đó, khi khai thác tập phổ biến trên CSDL phân cấp đòi hỏi phải khai thác cả tập các mục trên cây phân cấp bao gồm ($I \cup J$).

Liu và các đồng sự [11] đưa ra hai định nghĩa để khai thác tập phổ biến từ CSDL có sự phân cấp các mục như sau:

Định nghĩa 1: Một giao dịch $t = <tid, X>$ với $X \in (I \cup J)$, $X = (Y \cup Z)$ là tập các mục có trong giao dịch (Y) và các mục cha của Y trên cây phân cấp (Z).

Định nghĩa 2: Tập X là tập phổ biến thì $suport(X) > minsup$, đồng thời trong X không tồn tại một cặp mục nào có quan hệ cha con, như vậy X là phổ biến khi:

$$\begin{cases} Support(X) \geq minsup \\ \forall X_i, X_j \in X, X_i \neq Parent(X_j) \end{cases}$$

Khai thác FWUI trên CSDL số lượng có sự phân cấp có những đặc trưng riêng khác với trên CSDL nhị phân có sự phân cấp, bởi các mục trong CSDL có kèm theo số lượng và trọng số. Do đó, để khai thác tập phổ biến trên CSDL có sự phân cấp các mục bao gồm cả các mục nút cha, chúng tôi đề xuất các định nghĩa sau:

Định nghĩa 3: Nút cha trên cây phân cấp thuộc các giao dịch chứa nút con của nó.

Với mỗi mục nút cha X trên cây phân cấp và $t_k \in T$:

$X \in t_k$ nếu và chỉ nếu $Y \in t_k$ và Y là con ở nút lá của X trên cây phân cấp.

Khai thác tập phổ biến trên CSDL sỗ lượng có sự phân cấp, cần xác định trọng số của các mục nút cha trên cây phân cấp, đồng thời phải xác định số lượng của các mục cha trong mỗi giao dịch mà nó có mặt. Do các mục cha này sẽ được thêm vào các giao dịch trước khi khai thác theo định nghĩa 3.

Để đảm bảo các mục nút cha sau khi thêm vào các giao dịch trong CSDL không quá khác biệt với các mục con ở nút lá, đồng thời các mục nút cha vẫn thể hiện được vai trò của nó, trong bài báo này chúng tôi xác định trọng số mục nút cha và số lượng của chúng trong mỗi giao dịch theo định nghĩa 4 và 5.

Định nghĩa 4: Trọng số của mục nút cha trên cây phân cấp bằng trọng số lớn nhất của trọng số các nút con của nó ở nút lá:

$$weight(A) = max(weight(A_1), weight(A_1), .. weight(A_k))$$

Trong đó A là mục nút cha trên cây phân cấp, $A_1, A_2, .., A_k$ là các nút lá của A

Ví dụ 2: $weight(K) = max(weight(C), weight(B), weight(F)) = max(0.5, 0.2, 0.1) = 0.5$

Theo định nghĩa 4, các mục nút cha ở mức càng thấp thì trọng số càng cao, điều này phản ánh được độ "quan trọng" của các mục ở mức khái quát, nghĩa là các mục ở mức khái quát càng cao thì trọng số càng lớn.

Định nghĩa 5: Số lượng của mục nút cha trên cây phân cấp ở trong giao dịch nào thì bằng số lượng lớn nhất của số lượng các nút con của nó ở trong giao dịch đó.

$$quantitative(A) \in t_k = max(quantitative(A_1), quantitative(A_1), .. quantitative(A_k))$$

Trong đó: $A_1, A_2, .., A_k \in t_k$ và $A_1, A_2, .., A_k$ là con của A trên cây phân cấp.

Ví dụ 3: $quantitative(K) \in t_5 = max(quantitative(B), quantitative(C), quantitative\ (F))$ $(B, C, F \in t_5) = max(2, 2, 1) = 2$

Việc xác định số lượng các mục nút cha khi thêm vào các giao dịch có chứa các nút con của nó theo định nghĩa 5 đảm bảo được vai trò của nó khi thêm vào CSDL, đồng thời số lượng của mục nút cha cũng không quá chênh lệnh so với số lượng của các nút con của nó.

Định nghĩa 6: Itemset $X \in (I \cup J)$ với I là tập mục trong CSDL ban đầu (tập nút lá trên cây phân cấp) và J là tập các mục nút cha trên cây phân cấp được gọi là phổ biến theo ngưỡng *minwus* nếu $wus(X) \geq minwus$, với *minwus* do người dùng xác định trước.

III. THUẬT TOÁN KHAI THÁC TẬP PHỔ BIẾN TỪ CSDL SỐ LƯỢNG CÓ SỰ PHÂN CẤP CÁC MỤC

A. Cấu trúc HIT-tree

Chúng tôi đề xuất một cấu trúc dữ liệu có tên HIT-tree, đây là một mở rộng của IT-tree [2] dùng để khai thác tập phổ biến trên CSDL số lượng có sự phân cấp mục theo tiếp cận khai thác từ CSDL theo chiều dọc với một lần đọc CSDL. Mỗi nút trên HIT-Tree gồm 3 thành phần:

- itemset X – tập các mục trong CSDL
- tidset X – tập các giao dịch chứa X
- $wus(X)$ – độ hỗ trợ trọng số hữu ích của X

HIT-tree gồm nhiều mức, mỗi mức gồm nhiều lớp tương đương, mỗi lớp tương đương gồm nhiều nút. Các cặp itemset trên hai nút trong cùng một lớp tương đương kết hợp với nhau để tạo ra nút mới ở mức tiếp theo. Do đó, các itemset trên các nút cùng một lớp tương đương có cùng số lượng mục và chỉ khác nhau phần tử cuối cùng. Itemset X được tạo ra từ hợp hai itemset của hai nút cùng một lớp tương đương phải thỏa mãn hai điều kiện để được thêm vào HIT-tree:

- $\forall x' \in X \neg x'' \in X$: $parent(x') = x''$ (Không có cặp mục nào có mối quan hệ cha con trong X)
- $wus(X) \geq minwus$

Sau khi xây dựng xong, các itemset trên các nút của HIT-tree chính là tập FWUI cần tìm theo ngưỡng *minwus*.

B. Thuật toán

Thuật toán MINE_FWUI

Input: CSDL số lượng có sự phân cấp các mục *DB* và ngưỡng *minwus*
Output: HIT-tree chứa các tập phổ biến trọng số hữu ích.

```
1   MINE_FWUI()
2     ADD_PARENT();//thêm nút cha cùng số lượng vào các giao dịch, đồng thời tính trọng số cho nút cha
3     CALCULATE_ TWU();// tính twu của các giao dịch trong CSDL mới
4      F = { i ∈ (I ∪ J), wus(i) ≥ minwus}; //tập 1-itemset thỏa mãn ngưỡng phổ biến minwus
5      HIT-tree = ∅;
6      CREATE_HIT-tree(F)
7            P = ∅;
8            for all i ∈ F      // xét từng phần tử trong F
9                 for j ∈ F with j > i  //j phía sau i
10                    X = F_i ∪ F_j;     // X là itemset mới tạo thành từ F_i và F_j
11                    if ∀x' ∈ X ¬x''∈ X: parent(x') = x''// không tồn tại 1 cặp mục nào là cha con trong X
12                       T = tidset(F_i) ∩ tidset(F_j) //T là tidset của X
13                       if wus(X) ≥ minwus
14                            P = P ∪ <X, T, wus(X)>  // kết nạp nút mới vào lớp P
15                            HIT-tree = HIT-tree ∪ <X, T, wus(X)>  // kết nạp nút mới vào HIT-tree
16                    CREATE_HIT-tree(P)// gọi đệ quy với lớp P
```

Hình 2. Thuật toán khai thác FWUI từ CSDL trọng số có sự phân cấp các mục

Ví dụ 4: Thuật toán MINE_FWUI trong hình 2 với CSDL *DB* trong ví dụ 1 và *minwus* = 0.6 như sau:

Dòng 2, thủ tục *ADD_PARENT*() cho kết quả như bảng 4 và 5. Thêm các mục nút cha, số lượng mục nút cha vào CSDL được thực hiện theo định nghĩa 3 và 5, thêm trọng số các mục nút cha theo định nghĩa 4, ta có kết quả như sau:

Bảng 4. Các giao dịch

Giao dịch	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*K*
t_1	1	1	0	2	1	0	1	2	1
t_2	0	1	3	0	1	0	3	0	3
t_3	2	1	0	2	2	2	1	2	2
t_4	3	1	1	0	1	0	1	3	1
t_5	1	2	2	1	3	1	2	1	2
t_6	0	1	1	1	0	1	1	1	1

Bảng 5. Bảng trọng số

Mục	Trọng số
A	0.3
B	0.2
C	0.5
D	0.6
E	0.9
F	0.1
G	0.5
H	0.6
K	0.5

Dòng 3, thủ tục *CALCULATE_TWU*() cho kết quả như bảng 6:

Bảng 6. twu của các giao dịch

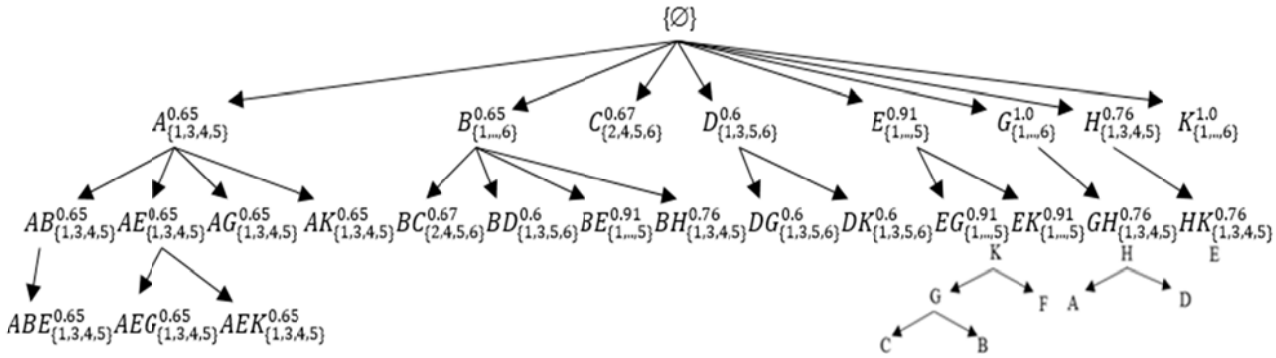
Giao dịch	*twu*
t_1	$(0.3 + 0.2 + 2 \times 0.6 + 0.9 + 0.5 + 2 \times 0.6 + 0.5)/7 =$ 0.68
t_2	$(0.2 + 0.5 \times 3 + 0.9 + 0.5 \times 3 + 0.5 \times 3)/5 =$ 1.12
t_3	$(0.3 \times 2 + 0.2 + 0.6 \times 2 + 0.9 \times 2 + 0.1 \times 2 + 0.5 + 0.6 \times 2 + 0.5 \times 2)/8 =$ 0.84
t_4	$(0.3 \times 3 + 0.2 + 0.5 + 0.9 + 0.5 + 0.6 \times 3 + 0.5)/5 =$ 0.76
t_5	$(0.3 + 0.2 \times 2 + 0.5 \times 2 + 0.6 + 0.9 \times 3 + 0.1 + 0.5 \times 2 + 0.6 \times 0.5 \times 2)/9 =$ 0.84
t_6	$(0.2 + 0.5 + 0.6 + 0.1 + 0.5 + 0.6 + 0.5)/7 =$ 0.43
sum	4.67

Dòng 4, tập *F* (1-itemset phổ biến) gồm {*A*, *B*, *C*, *D*, *E*, *G*, *H*, *K*} như bảng 7:

Bảng 7. Tập 1-itemset phổ biến

Mục	*wus*	F
A	$(0.68 + 0.84 + 0.76 + 0.84)/4.76 =$ 0.65	*A*
B	$(0.68 + 1.12 + 0.84 + 0.76 + 0.84 + 0.43)/4.76 =$ 1.0	*B*
C	$(1.12 + 0.76 + 0.84 + 0.43)/4.76 =$ 0.67	*C*
D	$(0.68 + 0.84 + 0.84 + 0.43)/4.76 =$ 0.6	*D*
E	$(0.68 + 1.12 + 0.84 + 0.76 + 0.84)/4.76 =$ 0.91	*E*
F	$(0.84 + 0.84 + 0.43)/4.76 =$ 0.45	
G	$(0.68 + 1.12 + 0.84 + 0.76 + 0.84 + 0.43)/4.76 =$ 1.0	*G*
H	$(0.68 + 0.84 + 0.76 + 0.84 + 0.43)/4.76 =$ 0.76	*H*
K	$(0.68 + 1.12 + 0.84 + 0.76 + 0.84 + 0.43)/4.76 =$ 1.0	*K*

Từ dòng 6 đến dòng 16 thủ tục đệ quy *CREATE_HIT-tree*() xây dựng cây HIT-tree bao gồm các nút là FWUI.



Hình 3. Cây HIT-tree với CSDL *DB* và *minwus* = 0.6

Xét nút A trên cây HIT-tree:

A kết hợp với *B*: *tidset*(*AB*) = *tidset*(*A*) ∩ *tidset*(*B*) = {1, 3, 4, 5} ∩ {1, 2, 3, 4, 5, 6} = {1, 3, 4, 5}, *wus*(*AB*) = 0.68 > *minwus* → kết nạp *AB* vào HIT-tree.

A kết hợp với *C*: *tidset*(*A*, *C*) = *tidset*(*A*) ∩ *tidset*(*C*) = {1, 3, 4, 5} ∩ {2, 4, 5, 6} = {4, 5}, *wus*(*AC*) = 0.34 < *minwus* → không kết nạp *AC* vào HIT-tree.

Tương tự kết nạp {*AE*, *AG*, *AK*} vào HIT-tree.

A kết hợp với *H*, không xét do *H* là cha của *A* trên cây phân cấp.

Tương tự với các nút *B*, *C*, *D*, *E*, *G*, *H*, *K* ta có cây HIT-tree như hình 3 có các nút là FWUI.

C. Một số kĩ thuật cải tiến tốc độ tính toán

Zaki và các đồng sự [11] đề xuất kĩ thuật diffset thay thế tidset nhằm rút gọn bộ nhớ và tăng tốc độ tính toán. Hàm và các đồng sự đề xuất cấu trúc MByS [9], MBiS [10] với mục tiêu tối ưu bộ nhớ trên tidset và tăng hiệu quả xử lý. Trong bài báo này chúng tôi sử dụng cả ba giải pháp này trong thuật toán MINE_FWUI và so sánh chúng với nhau trong khai thác itemset trên CSDL số lượng có sự phân cấp các mục.

IV. KẾT QUẢ THỰC NGHIỆM

A. Môi trường thực nghiệm

Hệ thống thử nghiệm được cài đặt bằng C# 2014 trên nền Microsoft Windows 8 Pro 64 bit, .Net Framework 4.5, thực hiện trên CPU Intel® Haswell Core™ i5 - 1.4 GHz, Ram 4Gb. Hệ thống phần mềm được sử dụng: Visual Studio 2013 Ultimate.

B. CSDL thực nghiệm

CSDL thực nghiệm gồm ba CSDL: SALE-FACT_1997, SALE-FACT-1997_1998, SALE-FACT_SYNC rút ra từ CSDL Mirosoft Foodmart2000 của Microsoft SQL2000 (trong đó, SALE-FACT-1997_1998 là bản kết hợp của SALE-FACT-1997 và SALE-FACT-1998; SALE-FACT-SYNC là bản kết hợp của SALE-FACT-1997, SALE-FACT_1998 và SALE-FACT-dec_1998). Cụ thể các CSDL phân cấp mục được mô tả như trong bảng 8 và 9.

Bảng 8. Mô tả CSDL thực nghiệm

Tên CSDL	Số lượng giao dịch
SALE-FACT_1997	20.522
SALE-Fact_1997_1998	54.537
SALE-FACT_SYN	58.308

Bảng 9. Cấu trúc cây phân cấp

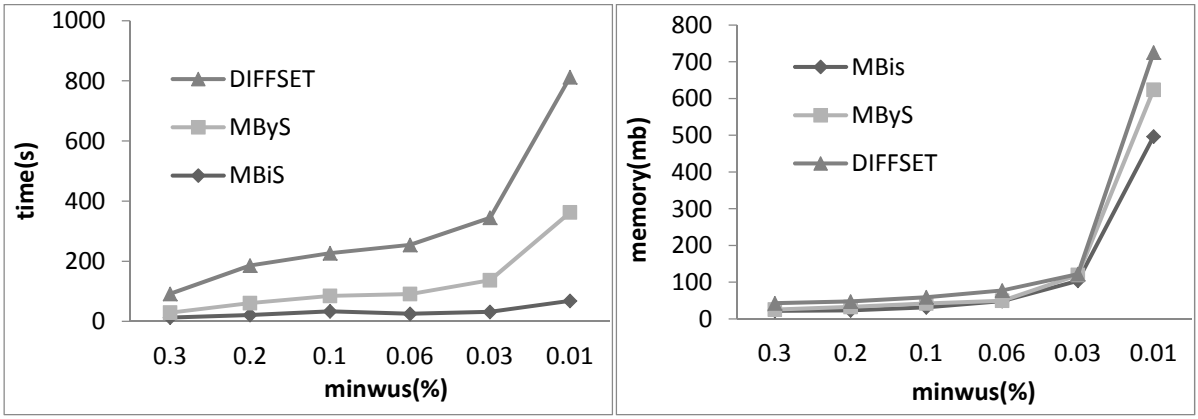
Mức	Tên mức	Số lượng nút
1	Product_family	3
2	Product_department	24
3	Product_category	48
4	Product_subcategory	56
5	Product_class	110
6	Product	1560

Từ bảng 9 ta thấy, có ba cây phân cấp (số lượng nút ở mức 1 là 3), độ cao của các cây phân cấp là 6 (có 6 mức).

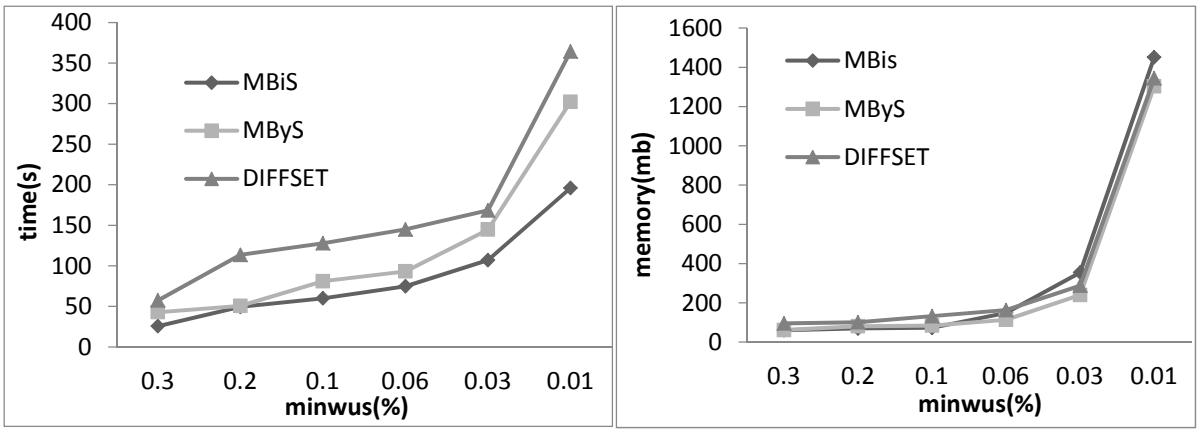
C. Kết quả thử nghiệm

Để kết quả so sánh có độ chính xác cao, với mỗi ngưỡng phổ biến *minwus* chúng tôi tiến hành chạy chương trình 5 lần với mỗi phương pháp, sau đó lấy trung bình cộng của 5 lần chạy.

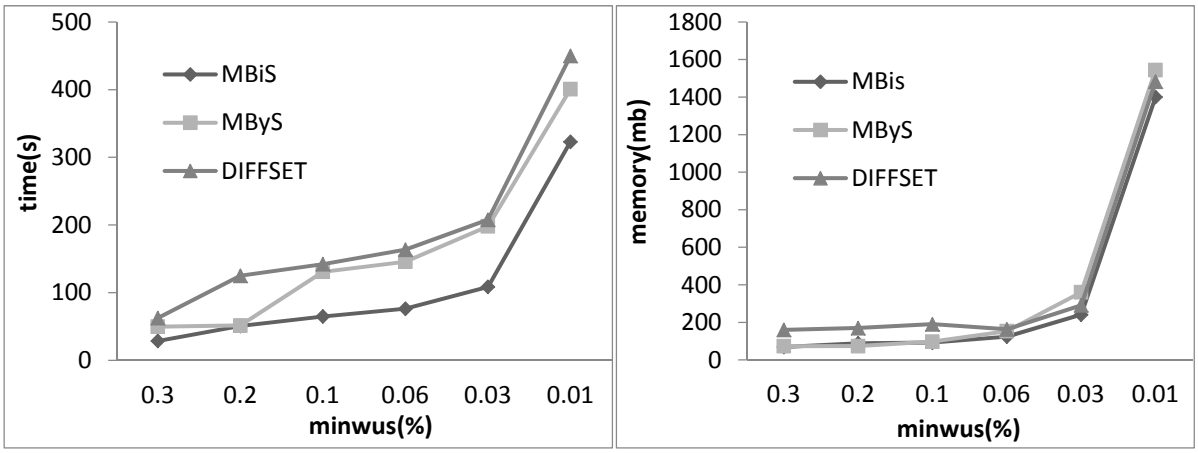
Kết quả thử nghiệm trên các CSDL cho trong bảng 8 lần lượt được thể hiện qua các biểu đồ sau:



Hình 4. Kết quả so sánh về thời gian (hình bên trái) và bộ nhớ (hình bên phải) trên CSDL SALE_FACT-1997



Hình 5. Kết quả so sánh về thời gian (hình bên trái) và bộ nhớ (hình bên phải) trên CSDL SALE_FACT-1997_1998



Hình 6. Kết quả so sánh về thời gian (hình bên trái) và bộ nhớ (hình bên phải) trên CSDL SALE_FACT-SYNC

Các kết quả thực nghiệm từ hình 4 đến 6 trên cho thấy về mặt thời gian thuật toán sử dụng cấu trúc MBiS đạt được hiệu quả cao nhất sau đó là MBySS và cuối cùng là DIFFSET. Ví dụ, CSDL SALE-FACT_1997 với ngưỡng *minwus* = 0.01, MBiS có thời gian xử lý là 68,301s, trong khi MBySS là 294,022s và DIFFSET là 449.854s. Quan sát các hình bên phải (so sánh bộ nhớ sử dụng), ta thấy sự chênh lệch về bộ nhớ giữa các phương pháp là không đáng kể. Quan sát các hình bên trái (so sánh thời gian chạy), ta thấy với CSDL càng lớn thì DIFFSET càng có hiệu quả hơn, tiệm cận dần với phương pháp sử dụng cấu trúc MBySS.

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đề xuất bài toán khai thác tập phổ biến trên CSDL số lượng có sự phân cấp mục, và phương thức tính trọng số và số lượng cho các mục trên cây phân cấp khi thêm vào CSDL bằng các định nghĩa 4 và 5. Đồng thời đề xuất thuật toán MINE_FWUI cùng với cấu HIT-tree để giải quyết bài toán này với chỉ một lần đọc CSDL.

Bài báo thực nghiệm thuật toán đề xuất với các cấu trúc hiện có đối với khai thác dữ liệu theo chiều dọc như Diffset, MBySS, MBiS trong lưu trữ tidset và so sánh hiệu quả của chúng về mặt thời gian chạy và bộ nhớ sử dụng. Kết quả thực nghiệm cho thấy cấu trúc MBiS có kết quả tốt nhất về mặt thời gian, kế đến là MBySS và cuối cùng là kĩ thuật Diffset.

Tiếp tục phát triển các kết quả đã đạt được, thời gian tới nhóm sẽ tiếp tục nghiên cứu mở rộng các bài toán trên CSDL số lượng có sự phân cấp mục, như khai thác tập phổ biến với nhiều ngưỡng hỗ trợ, khai thác tập phổ biến đóng, v.v.. Đồng thời, nghiên cứu các thuật toán hiệu quả hơn để giải quyết bài toán này như loại bỏ quá trình thêm mục nút cha vào CSDL, cũng như đề xuất các cấu trúc hiệu quả hơn trong khai thác tập phổ biến trên CSDL loại này.

VI. TÀI LIỆU THAM KHẢO

[1] Agrawal, R., Srikant, R.: "Fast algorithms for mining association rules". *Proc. of the 20th VLDB Conf. Santiago*, Chile, pp. 487–499, 1994.

[2] Zaki, M. J.: "Scalable algorithms for association mining". *IEEE Transactions on Knowledge and Data Engineering*, 12(3), pp. 372-390, 2000.

[3] Vo, B., Hong, le., Le, B.: "DBV-Miner: *A* Dynamic Bit-Vector approach for fast mining frequent closed itemsets". *Expert Systems with Applications* 39, pp. 7196–7206, 2012.

[4] Khan, M. S., Muyeba, M., Coenen, F.: "A weighted utility framework for mining association rules". *Proc. of conf. IEEE European Modeling Symposium*, pp. 87 – 92, 2008.

[5] Han, J., Fu, Y.: "Discovery of multiple-level association rules from large databases". *Proc.* of *Conf. on Very Large Data Bases*, Zurich, Switzerland, pp.420–431, 1995.

[6] Liu, B., Hsu, W., Ma, Y.: "Mining association rules with multiple minimum *supports*". *Proc.1999 Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, USA, pp.337–341, 1999.

[7] Tseng, M, C., Lin, W, Y.: "Efficient mining of generalized association rules with non-uniform minimum support". *Data & Knowledge Engineering* 66(1), pp.41-64, 2007.

[8] Vo, B., Le, B.: "Fast Algorithm for Mining Generalized Association Rules". *International Journal of Database Theory and Application* 2(3), pp.1-12, 2009.

[9] Nguyễn Duy Hàm, Võ Đình Bảy, Minh, Nguyễn Thị Hồng Minh: "Một phương pháp khai thác nhanh FWUI trên CSDL số lượng". *Một số vấn đề chọn lọc về CNTT và TT lần thứ 17*. pp.280-285, 2014.

[10] Ham, N, D., Vo, B., Minh, N, T, H., Hong , T, P.: "MBiS: an efficient method for mining frequent weighted utility itemsets from quantitative databases". *Journal of Computer Science and Cybernetics*, 31(1), pp.17-30, 2015.

[11] Zaki, M. J., Gouda, K.: "Fast Vertical Mining Using Diffsets". *KDD '03 Proc. of the ninth ACM SIGKDD international conf. on Knowledge discovery and data mining*, Washington, DC, USA, pp.326-335, 2003.

[12] Vo, B., & Le, B., Jason J. Jung, "A Tree-based Approach for Mining Frequent Weighted Utility Itemsets", *Computational Collective Intelligence Technologies and Applications*, *Lecture Notes in Computer Science Volume* 7653, pp. 114-123, 2012.

[13] Lan, G, C., Hong, T,P., Wu, P, S., "Mining hierarchical temporal association rules in a publication database", *Proc of IEEE Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pp.503 – 508, 2013.

[14] Ali, S, Z., Rathore, Y., "A effective and efficient algorithm for cross level frequent pattern mining", *Conf on Advances in Engineering and Technology Research (ICAETR)*, pp.1-6, 2014.

MINING FREQUENT WEIGHTED UTILITY ITEMSETS FROM QUANLITY DATABASE WITH HIERARCHY OF ITEMS

Nguyen Duy Ham, Vo Dinh Bay, Nguyen Thi Hong Minh

ABSTRACT *- Mining frequent itemsets* (*FIs*) *to find relationships among items plays an important role in data mining*. *Besides, mining FIs from traditional databases*, *mining FIs from weighted transactions databases and quantitative databases has received a lot of attention in recent years*. *However*, *there research only mining from database which no relation between the items from database*. *This paper*, *we propose the problem for mining FIs from quantitative databases with hierachy of items and propose an algorithm for sloving this problem based on diffset strategy*, *and MByS*, *MBiS structure in storing the tidset of itemset*. *The experimental results show that the method used MBiS structure to give the best effectively on runtime*.