KHAI THÁC TOP-RANK-K TẬP ĐƯỢC ĐÁNH TRỌNG SỐ

Nguyễn Thành Ngô

Trường Đại học Công nghiệp Thực phẩm Tp Hồ Chí Minh

Ngày gửi bài: 08/4/2016 **Ngày chấp nhận đăng: 06/6/2016**

TÓM TẮT

Trong bài báo này, chúng tôi đề xuất giải thuật WIT-TOP-K dựa trên cấu trúc cây WIT-tree để khai thác Top-rank-k của các tập được đánh trọng số dựa trên cơ sở giao dịch có trọng số. Kết quả thực nghiệm cho thấy thuật toán đề xuất thực hiện khá nhanh và chính xác đối với các cơ sở dữ liệu có kích thước vừa phải với mức ngưỡng thích hợp.

Từ khóa: Top-rank-k mining, luật kết hợp, tập phổ biến, khai thác tập có trọng số

AN EFFICIENT ALGORITHM FOR MINING WEIGHTED ITEMSET TOP-RANK-K

ABSTRACT

In this paper, we propose an efficient algorithm, called WIT-TOP-K, based on the WIT-tree data structure to mine the weighted itemset Top-rank-k in weighted transaction database. The experiment results show that the proposed algorithm performs quickly and accurately with respect to the moderate-sized databases with a appropriate thresholds.

Key words: Top-rank-k mining, association rules, frequent itemset, frequent patterns, weighted itemsets

1. GIỚI THIỆU

Khai thác dữ liệu được dùng để mô tả quá trình tìm kiếm, chắc lọc và khai phá tri thức trong cơ sở dữ liệu. Quá trình này bao gồm tập hợp nhiều kỹ thuật được sử dụng trong tiến trình phát hiện tri thức và tìm ra được mối quan hệ giữa các mẫu trong cơ sở dữ liệu (transaction database).

Khai thác luật kết hợp là một phần quan trọng trong quá trình khám phá tri thức trong dữ liệu (KDD) [2]. Khai thác luật kết hợp được sử dụng để xác định mối quan hệ giữa các sản phẩm trong cơ sở dữ liệu giao dịch và điều này dẫn đến việc nó chỉ quan tâm đến việc khách hàng có mua hay không mua sản phẩm nào đó. Thực tế, mỗi một sản phẩm có thể có giá trị khác nhau. Tương tự mỗi *item* trong cơ sở dữ liệu giao dịch cũng có trọng số khác nhau tùy thuộc từng cơ sở dữ liệu cụ thể. Vì vậy, việc khai thác trên loại dữ liệu này mang tính thực tiễn cao.

Năm 1998, Ramkumar, Ranka và Tsur [4] cũng như Cai, Fu, Cheng và Kwong [3] đã đề xuất một mô hình để mô tả các khái niệm về việc khai thác luật kết hợp có trọng số và dựa trên giải thuật Apriori để tìm ra các tập phổ biến được đánh trọng số. Từ đó nhiều kỹ thuật khai thác luật kết hợp có trọng số được đề xuất như: Wang, Yang, Yu [6] và Tao, Murtagh, và Farid [5].

Giải thuật khai thác Top-rank-k bằng *Node-list* tiến hành khai thác Top-rank-k của các tập phổ biến chỉ dựa trên cơ sở dữ liệu nhị phân chưa tiến hành trên cơ sở dữ liệu nhị phân có trọng số, và với việc dựa trên cấu trúc cây FP cho nên khiến cho việc tìm kiếm các tập phổ biến phải quét cơ sở dữ liệu đến 2 lần.

Trong bài báo này chúng tôi trình bày về thuật toán khai thác Top-rank-k dựa trên thuật toán WIT-FWI. Thuật toán WIT-FWI dựa trên phần giao giữa các *Tidset* để tính nhanh các trọng số hỗ trợ và để sinh ra các ứng viên từ các lớp tương đương cho trước. Các tập phổ biến có trọng số hỗ trợ thỏa điều kiện trọng số hỗ trợ tối thiểu sẽ được giữ lại, các tập không thỏa

sẽ bị xóa bỏ khỏi cây. Thuật toán WIT-TOP-K cũng được dựa trên giải thuật WIT-FWI để tính nhanh các các trọng số hỗ trợ, các tập được đánh trọng số có kích thước là 1 mà trọng số hỗ trợ của nó thỏa điều kiện ngưỡng Top-rank-k cho trước sẽ được đưa vào bảng gọi là Tab_k. Từ các tập được tìm thấy trong bảng Tab_k, sẽ phát sinh ra các ứng viên mới dựa theo các lớp tương đương, và sử dụng định lý 2.1 để loại bỏ các tập không thỏa điều kiện ngưỡng k.

2. MỘT SỐ ĐỊNH NGHĨA VÀ KHÁI NIỆM CƠ BẢN

Định nghĩa 2.1. Cơ sở dữ liệu giao dịch của tập được đánh trọng số *(D)* bao gồm: một tập hợp các giao dịch $T = \{t_1,t_2,....,t_m\}$; một tập các *item* $I = \{i_1,i_2,...,i_n\}$ và một tập hợp các trọng số $W = \{w_1,w_2,...,w_n\}$ tương ứng với mỗi một *item* có trong *I.*

Bảng 2.1: Trọng số của các item

Item	Weight
A	3
B	5
C	1
D	3
E	5

Định nghĩa 2.2. Trọng số giao dịch của một giao dịch t_k được gọi là *tw* và được nghĩa như sau:

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

Dựa vào dữ liệu trong Bảng 2.1 và định nghĩa 2.1 ta tính được trọng số giao dịch của giao dịch T_1 như sau :

Ta có $T_1 = \{C,E\}$ tương ứng với $W_C = 1$, $W_E = 5$ suy ra ta có $tw(t_1)$ được tính như sau:

$$tw(t_1) = \frac{W_C + W_E}{2} = \frac{1+5}{2} = 3$$

Tương tự với các giao dịch khác ta có:

Bảng 2.2: Trọng số giao dịch của các giao dịch có trong D

Transations	tw
1	4
2	3.6
3	4
4	3.5
5	3.4
6	3
Sum	21.6

Định nghĩa 2.3. Trọng số hỗ trợ được ký hiệu là *ws* của một tập phổ biến được định nghĩa như sau:

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

Trong đó *T* là danh sách của tất cả các giao dịch có trong CSDL *(D)*

Từ định nghĩa 2.2 và Bảng 2.1 và 2.2, ta bắt đầu tìm kiếm tất cả trọng số hỗ trợ tương ứng với từng tập phổ biến. Ví dụ ta xét *item* A để tính *ws(A),* ta thấy *item A* xuất hiện trong các giao dịch T_1, T_3, T_4, T_5 vì vậy ta có *ws(A)* là:

$$ws(A) = \frac{tw(t_3) + tw(t_4) + tw(t_6) + tw(t_8) + tw(t_9)}{Sum} = \frac{3 + 3.67 + 4 + 3 + 4}{35.33} \approx 0.5$$

Để khai thác các tập được đánh trọng số, chúng ta phải xác định được tất cả các trọng số hỗ trợ tương ứng với từng tập phổ biến thỏa yêu cầu của ngưỡng trọng số hỗ trợ tối thiểu *(minws)* do người dùng đặt ra.

$$FWI = \{X \subseteq I | ws(X) \geq minws\}$$

Định lý 2.1. *"Mọi tập con khác rỗng của tập phổ biến cũng là tập phổ biến và mọi tập chứa tập không phổ biến đều là tập không phổ biến"* có nghĩa là nếu cho $X \subset Y$ khi đó $ws(X) \geq ws(Y)$.

Chứng minh. Theo tính các chất của kết nối Galois[8], nếu $X \subset Y$ kéo theo $t(X) \supseteq t(Y)$, vì vậy ta có:

$$\Rightarrow \sum_{t_k \in t(X)} tw(t_k) \geq \sum_{t_k \in t(Y)} tw(t_k) \Rightarrow \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)} \geq \frac{\sum_{t_k \in t(Y)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)} \Rightarrow w(X) \geq w(Y)$$

Để khai thác các luật kết hợp có trọng số, đầu tiên chúng ta phải tìm tất cả các tập được đánh trọng số thỏa điều kiện ngưỡng trọng số tối thiểu *minws*. Việc khai thác các tập được đánh trọng số được xem là quá trình quan trọng nhất trong việc khai thác các luật kết hợp có trọng số. Ramkumar và các đồng sự (1998) [4] đã trình bày giải thuật khai thác các tập được đánh trọng số dựa trên mô hình thuật toán Apriori.

Nhược điểm chính của các giải thuật dựa trên thuật toán Apriori là việc phải quét cơ sở dữ liệu nhiều lần để tìm ra các tập phổ biến, dẫn đến việc sẽ phát sinh chi phí lớn.

Giải thuật khai thác các tập được đánh trọng số dựa trên cấu trúc WIT-tree (Weighted *Item*set-Tidset), cấu trúc này là phần mở rộng dựa trên cấu trúc cây IT-tree và mỗi một node trên WIT-tree bao gồm 3 thành phần:

1) *X:* đại diện cho một tập phổ biến
2) *t(x): Tidset* tập hợp các giao dịch có chứa *X*
3) *ws:* trọng số hỗ trợ của *X*

Mỗi một nút được ký hiệu như là một bộ ba $\langle X, t(X), ws \rangle$.

Giá trị của mỗi một nút được tính dựa theo công thức trọng số hỗ trợ (định nghĩa 2.2). Việc tính toán các trọng số hỗ trợ dựa trên các *Tidset*. Các liên kết kết nối các nút ở mức thứ *k* (gọi là *X*) với các các mức thứ *k+1* (gọi là *Y*).

Nút gốc *"root"* của cây WIT-tree chứa tất cả các nút có kích thước là 1 gọi là *1-itemset.* Tất cả các nút ở mức 1 sẽ trở thành lớp tương đương với tiền tố là {} (hay [∅]). Mỗi một nút trong mức 1 sẽ trở thành một lớp tương đương mới, và sử dụng các *item* này như là một tiền tố. Với mỗi một nút có chung tiền tố, nó sẽ kết hợp với các nút phía sau nó để tạo ra các lớp tương đương mới. Quá trình này sẽ được thực hiện đệ qui để tìm ra các lớp tương đương ở các mức cao hơn.

Định nghĩa 2.4. Rank của một tập được đánh trọng số

Cho một cơ sở dữ liệu giao dịch *(D)* và một tập $A(\subseteq I)$. Rank của A, R_A được định nghĩa như sau $R_A = |\{ws_X | X \subseteq I \text{ và } ws_X \geq ws_A\}|$ trong đó $|Y|$ là số phần tử có trong Y

Định nghĩa 2.5. Top-rank-k của tập được đánh trọng số

Cho một cơ sở dữ liệu giao dịch (D), một ngưỡng k và một mẫu A ($A \subseteq I$). A được gọi là thuộc Top-rank-k mẫu phổ biến khi và chỉ khi R_A không lớn hơn k, nghĩa là $R_A \leq k$.

Để tìm được Top-rank-k của một cơ sở dữ liệu giao dịch có trọng số ta phải tìm tất cả các tập được đánh trọng số mà *Rank* của nó không được lớn k.

3. GIẢI THUẬT KHAI THÁC TOP-RANK-K DỰA TRÊN CẤU TRÚC WIT-TREE

Đầu vào: cơ sở dữ liệu D, ngưỡng k

Đầu ra: Tab_k, với các mục cố định, mỗi một mục sẽ chứa các *item* cùng cấp

Mã giả:

WIT-TOP-K()

1.I_{order} = tất cả các tập có kích thước là 1 cùng với trọng số hỗ trợ

2.Sắp xếp các nút trong I_{order} theo thứ tự giảm dần của ws

3.Tab_k = {I thuộc I_{order} sao cho $R_i \leq k$} và gọi tập này là tập L_r

4. Gọi hàm **TOP-K-EXTEND** với tham số là L_r

TOP-K-EXTEND *(L_r)*

5.Cho mỗi node l_i trong L_r

6. Tạo một tập L_i bằng cách nối l_i với tất cả l_j theo sau trong L_r bằng cách

7. Tập $X = l_i.itemset \cup l_j.itemset$ và $Y = t(l_i) \cap t(l_j)$

8. $ws(X)$ = **COMPUTE-WS***(Y)*

9. Nếu $ws(X) \geq$ trọng số tối thiểu của k^{th} trong Tab_k

10. Chèn một node $\langle X, t(X), ws \rangle$ vào trong L_i

11. L_i sẽ được chèn vào trong Tab_k

12. Nếu số lượng node trong $L_i \geq 2$ khi đó

13. Gọi đệ qui hàm **TOP-K-EXTEND** với biến L_i

Mô tả giải thuật:

Đầu tiên cho tập I_{order} chứa tất các các tập được đánh trọng có kích thước là 1 cùng với trọng số hỗ trợ của chúng (dòng 1). Các nút chứa trong tập I_{order} sẽ được sắp xếp theo thứ tự giảm dần của trọng số hỗ trợ (dòng 2). Khởi tạo Tab_k và chèn tất cả các tập mà *Rank* của nó thỏa điều kiện ngưỡng k cho trước (dòng 3) và gọi tập này là L_r. Khởi tạo hàm TOP-K-EXTEND với biến là tập L_r để khai thác các tập con dựa trên các tập cha có trong Tab_k. Cho nút l_i có trong L_r với các nút phía sau nó để tạo các tập nút mới gọi là L_i (dòng 5 và 6). Để tạo ra L_i: đầu tiên, cho $X = l_i.itemset \cup l_j.itemset$ *và tính toán* $Y = t(X) = t(l_i) \cap t(l_j)$ (dòng 7). Và tính toán trọng số hỗ $ws(X)$ dựa trên hàm COMPUTE-WS*(Y)* (định nghĩa 2.2) (dòng 8). Nếu

ws(X) lớn hơn hoặc bằng trọng số hỗ trợ của *k*th (mức k thấp nhất trong Tab_k) thì $(l_i.itemset, l_i.ws)$ sẽ được chèn vào L_i (dòng 10) và L_i sẽ được chèn tiếp vào Tab_k (dòng 11). Nếu số lượng các nút trong L_i lớn hơn 1 (dòng 12), thì tiếp tục gọi đệ qui hàm TOP-K-EXTEND với biến là L_i để sinh ra các ứng viên khác (dòng 13). Nếu các tập con sinh ra mà trọng số của nó nhỏ hơn trọng số của tập cha và không thỏa điều kiện ngưỡng k, thì toàn bộ nút đó sẽ bị xóa bỏ, và không tiến hành khởi tạo hàm TOP-K-EXTEND để sinh ứng viên nữa.

Ta tiến hành sử dụng Bảng 1, 2, 3, để tiến hành tìm kiếm Top-rank-k các tập được đánh trọng với ngưỡng $k = 4$.

Bước 1: Ta tiến hành tính toán trọng số hỗ trợ của các tập có kích thước là 1 và khởi tạo tập I_{order}. Ta có ws(A) = 0.69, ws(B) = 1, ws(C) = 0.63, ws(D) = 0.67, ws(E) = 0.86. Ta có tập

$I_{order} = \{\langle A, 1345, 0.69\rangle, \langle B, 123456, 1\rangle, \langle C, 2456, 0.63\rangle, \langle D, 1356, 0.67\rangle, \langle E, 12345, 0.86\rangle\}$

Bước 2: Sau đó ta tiến hành sắp xếp I_{order} theo thứ tự giảm dần của trọng số hỗ trợ, chúng ta có

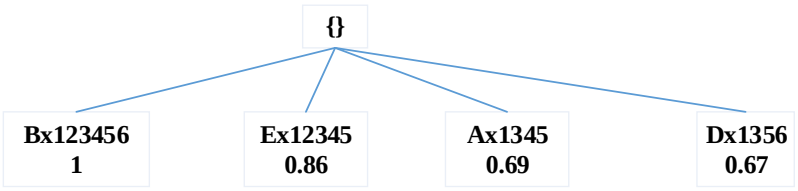
$I_{order} = \{\langle B, 123456, 1\rangle\langle E, 12345, 0.86\rangle\langle A, 1345, 0.69\rangle\langle D, 1356, 0.67\rangle\langle C, 2456, 0.63\rangle\}$.

Bước 3: Khởi tạo Tab_k, và chèn tất cả các tập thỏa điều kiện ngưỡng k = 4 (tập L_r) vào trong Tab_k. Ta có các tập sau:

Bảng 3.1: Tab_k sau khi chèn các tập có kích thước là 1 thỏa k = 4

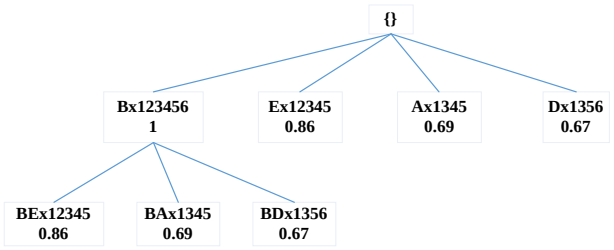
Rank	FWI	Ws
1	{B}	1
2	{E}	0.86
3	{A}	0.69
4	{D}	0.67

Bước 4: Tiến hành khởi tạo lớp tương đương {}, với các tập có kích thước là 1 chứa trong tập L_r



Hình 3.1. WIT-tree với các tập có kích thước là 1 trong Tab_k

Bước 5: Tiến hành khởi tạo ra lớp tương đương mới dựa trên lớp tương đương cũ. Ví dụ ta có B sẽ nối với E, ta có BE với *t(BE)* = 12345 và *ws(BE)* = 0.86, vì *ws(BE)* lớn hơn trọng số tối thiểu của bậc k thứ 4, nên ta thêm *BC* vào trong tập L_B. Ta tiếp tục tiến hành ghép *BA, với t(BA) = 1345 và ws(BA) = 0.69* cũng thỏa điều kiện ta chèn tiếp vào tập L_B. Tiếp tục ghép các tập với nhau ta có tập $L_B = \{\langle BE, 12345, 0.86\rangle, \langle BA, 1345, 0.69\rangle, \langle BD, 1356, 0.67\rangle\}$. Hình 3.2



Hình 3.2. Tập L_B được khởi tạo

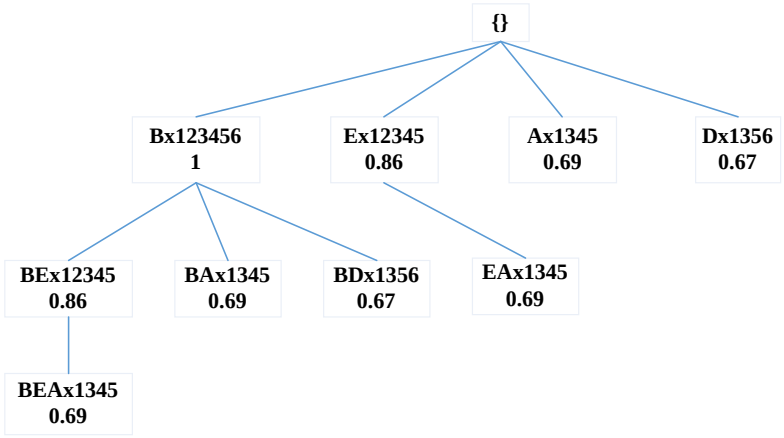
Bước 6: Tiến hành chèn L_B vào trong Tab_k, những mục nào có cùng trọng số thì sẽ được xếp chung với nhau.

Bảng 3.2: Tab_k sau khi chèn các tập có trong L_B thỏa minws

Rank	FWI	WS
1	{B}	1
2	{E}{BE}	0.86
3	{A}{BA}	0.69
4	{D}{BD}	0.67

Bước 7: Sau khi tạo ra tập L_B, bởi vì số lượng các nút trong L_B nhiều hơn 1, nên ta tiến hành gọi đệ qui hàm TOP-K-EXTEND để tiến hành tạo ra các nút con của tập L_B. Ta ghép *BE* với *BA* ta có *t(BEA) = 1345*, *ws(BEA)= 0.69*, ta thêm tập *BEA* vào trong tập $L_{BE} = \{\langle BEA, 1345, 0.69\rangle\}$ vì *ws(BEA)* không nhỏ hơn bậc k thứ 4 trong Tab_k. Ghép *BE* với *BD* ta có *t(BED) = 135*, *ws(BEA)= 0.53*, ta xóa bỏ tập này vì *ws(BEA)* nhỏ hơn bậc k thấp nhất trong Tab_k

Bước 8: Ta tiếp tục thực hiện quá trình để tìm ra tập phổ biến được đánh trọng số thỏa điều kiện ngưỡng k = 4. Giải thuật sẽ dừng lại khi không thể tạo ra được tập phổ biến nào nữa. Ta có Tab_k với ngưỡng k = 4 (Bảng 3.3)



Hình 3.3. Cây WIT-tree hoàn chỉnh mới mức k = 4

Bảng 3.3: Tab_k với mức k = 4

Rank	FWI	WS
1	{B}	1
2	{E}{BE}	0.86
3	{A}{BA}{EA}{BEA}	0.69
4	{D}{BD}	0.67

4. KẾT QUẢ THỰC NGHIỆM

4.1. Môi trường thực nghiệm

Thực nghiệm về hiệu quả của thuật toán khai thác Top-rank-k dựa trên WIT-FWI được tiến hành trên máy tính PC chạy Windows 10 professional 64bit, RAM 8GB, Intel(R) Core(TM) i5-5287U CPU @2.90GHz. Visual Csharp 2012.

4.2. Cơ sở dữ liệu thực nghiệm

Cơ sở dữ liệu được lấy từ trang web: http://fimi.cs.helsinki.fi/data/. Các bộ dữ liệu này đã được sửa đổi bằng cách tạo ra một bảng để lưu trữ các giá trị trọng số của từng *item* (giá trị trong khoảng từ 1 đến 10) cho mỗi một cơ sở dữ liệu.

Bảng 4.1: Cơ sở dữ liệu thực nghiệm có chỉnh sửa

CSDL	#Trans	#Item	Tình trạng
BMS-POS	515597	1656	Đã được sửa đổi
Connect	67557	130	Đã được sửa đổi
Chess	3196	76	Đã được sửa đổi
Mushroom	8124	120	Đã được sửa đổi

4.3. Thực nghiệm về số lượng tập

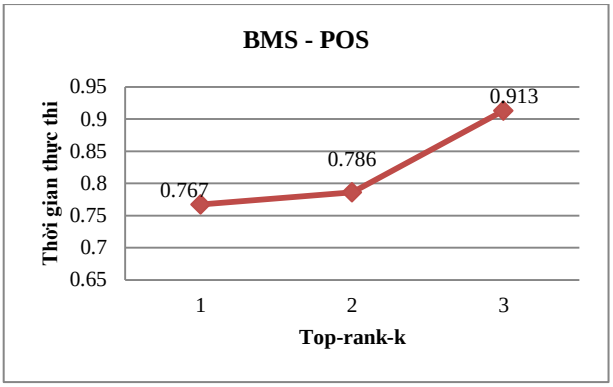
Kết quả thực nghiệm được tiến hành khai thác trên các cơ sở dữ liệu chuẩn ở Bảng 4.1. Bảng 4.2 trình bày số lượng các tập được đánh trọng số tìm thấy ứng với từng ngưỡng *k*, ngưỡng *k* càng cao thì số lượng các tập được tìm thấy càng cao và không tuân theo tỉ lệ nào do số lượng các tập có cùng *ws* trên từng cơ sở dữ liệu là khác nhau.

Bảng 4.2: Số lượng tập được tìm thấy khi khai thác

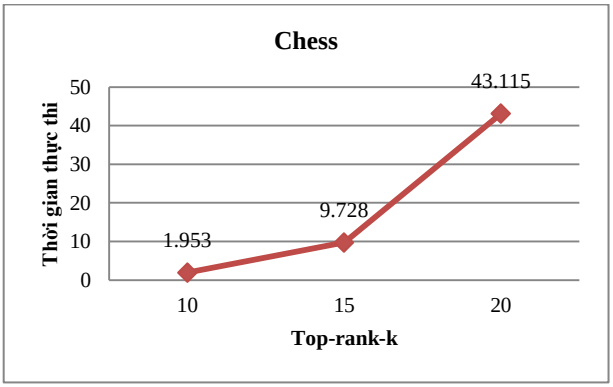
CSDL	Top-rank-k	#WI
BMS-POS	1	1
	3	3
	5	5
Connect	10	10
	15	15
	20	20
Chess	10	10
	20	20
	30	30
Mushroom	30	57
	40	105
	50	121

4.4. Thực nghiệm về thời gian thực thi

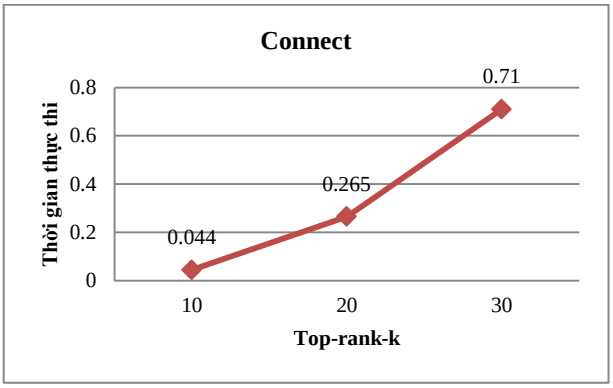
Thời gian thực thi để tìm kiếm các tập được đánh trọng số thay đổi tùy theo từng ngưỡng *k*. Dễ nhận thấy đối với các mức *k* càng cao thì thời gian thực thi càng dài.



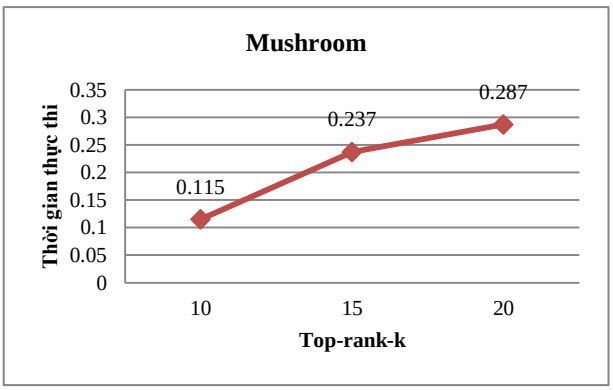
Hình 4.1. Biểu đồ thời gian khi khai thác Top-rank-k trên BMS-POS



Hình 4.2. Biểu đồ thời gian khi khai thác Top-rank-k trên Chess



Hình 4.3. Biểu đồ thời gian khi khai thác Top-rank-k trên Connect



Hình 4.4. Biểu đồ thời gian khi khai thác Top-rank-k trên Mushroom

Từ những kết quả thử nghiệm ở trên, ta thấy được thời gian xử lý WIT-TOP-K là tốn khá nhiều thời gian khi xử lý các cơ sở dữ liệu có số sản phẩm quá lớn hoặc mức ngưỡng *k* lớn. Tuy nhiên hệ thống xử lý khá nhanh và ổn đối với các cơ sở dữ liệu có kích thước vừa và nhỏ đối với mức ngưỡng thích hợp với từng cơ sở dữ liệu.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Bài báo đã nghiên cứu thuật toán khai thác Top-rank-k tập được đánh trọng số trên cơ sở dữ liệu giao dịch có trọng số dựa trên cấu trúc WIT-tree.

Một số đóng góp cụ thể như sau:

a) Nghiên cứu lý thuyết về các kỹ thuật khai thác các tập phổ biến như các phương pháp Apriori, FP-tree, IT-tree

b) Nghiên cứu lý thuyết về cơ sở dữ liệu giao dịch có trọng số, các định nghĩa và lý thuyết liên quan.

c) Nghiên cứu về thuật toán khai thác Top-rank-k tập phổ biến bằng cách sử dụng Node-lists.

d) Đề xuất thuật toán khai thác Top-rank-k tập được đánh trọng số dựa trên cấu trúc cây WIT-tree.

e) Cài đặt thực nghiệm để khảo sát kết quả của thuật toán đề xuất: tiến hành khai thác Top-rank-k trên các cơ sở dữ liệu chuẩn như BMS-POS, Chess, Connect, Mushroom.

5.2. Hướng phát triển

Thuật toán được xây dựng vẫn còn những hạn chế, thời gian xử lý để tìm ra các Top-rank-k tập được đánh trọng số còn khá chậm khi thực hiện trên các cơ sở dữ liệu lớn hoặc các ngưỡng Top-rank-k lớn.

Do đó, hướng phát triển tiếp theo của phương pháp này là áp dụng các lý thuyết về Diffset,… để tiến hành tính nhanh các trọng số hỗ trợ và phát triển một số chiến thuật để tỉa bớt các nhánh dư thừa trên cây để thu hẹp không gian tìm kiếm. Dựa vào đó để khai thác nhanh các tập được đánh trọng số giúp cho việc khai thác Top-rank-k được xử lý nhanh hơn.

TÀI LIỆU THAM KHẢO

[1]. Agrawal at al. (1993). Mining Association Rule between sets of items in large databases. ACM SIGMOD Record 22 (2) 207-216

[2]. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In: VLDB'94 (pp. 487-499)

[3]. Cai, C. H., Fu, A. W., Cheng, C. H., & Kwong, W. W. (1998). Mining association rules with weighted items. In: Proceedingss of international database engineering and applications symposium (IDEAS 98) (pp. 68-77).

[4]. Ramkumar, G. D., Ranka, S., & Tsur, S. (1998). Weighted association rules: Model and algorithm. In: SIGKDD'98 (pp. 661-666).

[5]. Tao, F., Murtagh, F., & Farid, M. (2003). Weighted association rule mining using weighted support and signficance framework. In: SIGKDD'03 (pp. 661-666)

[6]. Wang, W., Yang, J., & Yu, P. S. (2000). Efficient mining of weighted association rules. In: SIGKDD 2000 (pp. 270-274)

[7]. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In: SIGMOD (pp. 1-12)

[8]. Zaki et al. (1997). New algorithms for fast discovery of association rules. In: KDD97 (pp. 283-286)

[9]. Vo, B., Coenen, F., Le, B (2013). A new method for mining frequent weighted itemsets based on WIT-trees. Expert systems with applications 40(4), 1256-1264.

[10]. Deng, Z.H. (2014). Fast mining Top-rank-k frequent patterns by using Node-lists. Expert Systems with Applications 41(4/2), 1763-1768.