CÁCH TIẾP CẬN KỸ THUẬT KẾT HỢP LUẬT KHÔNG GIAN VÀ THỜI GIAN ỨNG DỤNG CHO BÀI TOÁN DỰ BÁO TRÊN BỘ DỮ LIỆU LỚN

Nguyễn Văn Thiện[1], Phạm Văn Hải[2*]

[1-2] Viện Công nghệ thông tin & Truyền thông, Trường Đại học Bách khoa Hà Nội,

thienkstn93@gmail.com, *haipv@soict.hust.edu.vn*

[2*] Coresponding Author: *haipv@soict.hust.edu.vn*

TÓM TẮT *- Bài báo này trình bày hướng tiếp cận cho việc giải quyết vấn đề hiệu năng cho việc khai phá bộ dữ liệu có đặc tính không gian – thời gian, qua đó tìm ra những quy luật kết hợp phổ biến sinh ra từ bộ dữ liệu. Trong các kỹ thuật sinh luật truyền thống dựa trên dữ liệu, khai phá dữ liệu từ các giao dịch được thực hiện độc lập nhau. Khi sử dụng thuật toán khai phá thông thường như Apriori hay Extend-Apriori thì chi phí tính toán tập các phần tử phổ biến, trong đó việc sinh tập các ứng viên, chi phí thời gian thực hiện lớn do quét cơ sở dữ liệu nhiều lần. Bên cạnh đó, việc sinh luật không gian – thời gian phải dựa trên sự phụ thuộc lẫn nhau giữa các giao dịch, nhằm thể hiện được mức độ liên quan của các phần tử trong một khoảng không – thời gian nào đó. Chúng tôi sử dụng một cửa sổ trượt giúp chuyển các giao dịch độc lập vào trong cùng một giao dịch mới được gọi là liên giao dịch. Sau đó tiến hành áp dụng một kỹ thuật khai phá mới mà chúng tôi đề xuất cho việc khai phá. Nhằm thể hiện kết quả thực nghiệm của thuật toán đề xuất chúng tôi chạy trên bộ dữ liệu lớn về thời tiết, đây là loại dữ liệu mang tính chất không gian và thời gian, từ bộ dữ liệu này chúng tôi tìm ra một cách hiệu quả các quy luật phổ biến ứng dụng cho các lĩnh vực dự báo thời tiết và biến đổi khí hậu, giảm đáng kể chi phí thời so sánh với thuật toán Apriori.*

Từ khóa *- Liên giao dịch, cây phần tử, tập phổ biến, tập phổ biến tối đa.*

I. GIỚI THIỆU

Trong việc tìm kiếm các luật kết hợp cho các bộ dữ liệu mang tính chất không gian và thời gian, nghĩa là ngoài những trường đặc tính đặc trưng cho loại dữ liệu, chúng còn gắn chặt với các thuộc tính kèm theo như chúng được thu thập ở đâu và khi nào. Vì thế với các bản ghi dữ liệu độc lập khi thu thập, chúng cần có cơ chế tạo được sự phụ thuộc lẫn nhau, điều này khác so với các loại dữ liệu khác. Năm 2003, Tung và các cộng sự của ông [3] đã đưa ra một kỹ thuật nhằm tạo sự phụ thuộc đó nhờ sử dụng một cửa sổ trượt kích thước *w,* những bản ghi nằm trong phạm vi của cửa sổ trượt có thể được nhóm lại thành một bản ghi mới, điều này sẽ được chúng tôi trình bày cụ thể trong phần II của bài viết này. Việc sinh các luật kết hợp, bên cạnh các thuật toán khai luật kết hợp cổ điển như thuật toán Apriori [1] , được thực hiện dựa trên nguyên tắc tập hợp có k phần tử là phổ biến thì tất cả tập con của nó cũng là phổ biến. Thuật toán này dựa trên việc sinh tất cả các tập phổ biến có một phần tử, với k > 2, thực hiện phép nối giữa 2 tập phổ biến có (k-1) phần tử như là một ứng viên, kiểm tra các tập ứng viên đó và dừng lại khi không có sinh ra ứng viên nào nữa. Nhược điểm của thuật toán này tốn chi phí cho việc sinh ra tập ứng viên là rất lớn. Thuật toán EApriori (Extended Apriori) và EH-Apriori (Extended Hash Apriori) của nhóm tác giả Lu et. al [2], đã nghiên cứu mở rộng thuật toán Apriori cho khai phá trên liên giao dịch, trong đó EH-Apriori sử dụng hàm băm làm giảm số lượng ứng viên chứa 2 phần tử. Những hướng tiếp cận khác thay vì việc sinh và kiểm tra các tập ứng viên, nhiều thuật toán khác lại dựa trên việc không sinh ra các ứng viên nhằm làm giảm thời gian kiểm tra chúng như FITI (First Intra Then Inter) của nhóm tác giả Tung et al [3]. Đầu tiên xác định tất cả các tập phần tử phổ biến trong giao dịch cổ điển, và sử dụng chúng để xác định tất cả các tập phần tử phổ biến trong liên giao dịch. Thuật toán ITP-Miner (Inter-Transaction Patterns Miner) của nhóm tác giả Lee và Wang (xem trong [4]) thực hiện việc quét dữ liệu trong một lần và được đánh giá là có thời gian giảm đáng kể so với Apriori hay EHApriori. Yo-Ping Huang, Li-Jen Kao, Frode-Eika Sandnes [5-6] đề xuất thuật toán Reduced Prefix-Projected Itemsets (RPPI) tại mỗi lần quét loại bỏ các phần tử không phổ biến ra khỏi cơ sở dữ liệu.

Trong bài báo này, chúng tôi đề xuất một kĩ thuật mới dựa trên ý tưởng không sinh tập các ứng viên để tìm ra các tập phổ biến. Tại mỗi nút sử dụng một cấu trúc đầu và đuôi và một tập để lưu các phần tử phổ biến, trong đó phần đầu mỗi nút lưu trữ phần tử đã kiểm tra mà nó là phổ biến. Khi thu được một tập các phần tử phổ biến tối đa từ tập lưu các phần tử phổ biến của nút gốc. Để giảm được chi phí quét cơ sở dữ liệu, chúng tôi đề xuất phương pháp mới sử dụng một cửa sổ trượt trên một chiều thuộc tính dựa trên trục thời gian để chuyển các giao dịch trên các khoảng thuộc tính riêng rẽ vào trong cùng một giao dịch mới được gọi là liên giao dịch nhờ mỗi phần tử sẽ lưu trữ tập các giao dịch mà chứa nó và việc tạo nút con chỉ yêu cầu quét cơ sở dữ liệu của các phần tử nên việc quét sẽ nhanh hơn nhiều so với thực hiện quét trên toàn bộ giao dịch. Phần I của bài báo đưa ra vấn đề và các hướng tiếp cận cách giải quyết đã được đề xuất, trên cơ sở đó chúng tôi đưa ra một hướng tiếp cận mới cho bài toán. Phần II đưa ra một số khái niệm, định nghĩa được sử dụng để mô hình bài toán. Phần III trình bày thuật toán đề xuất, cách tạo một cấu trúc cây phần tử và thuật toán khai phá để tìm tập các phần tử phổ biến. Trong phần IV, chúng tôi đưa ra một số kết quả thực nghiệm trên một số bộ dữ liệu lớn. Trong phần kết luận chúng tôi đưa ra cách đánh giá kết quả, thảo luận kết quả nghiên cứu và đề xuất hướng phát triển của giải thuật.

II. LIÊN GIAO DỊCH

Trong các thuật toán sinh luật cổ điển, chủ yếu thực hiện trên các giao dịch độc lập nhau [1]. Trong bộ dữ liệu về thời tiết, các đặc tính như, các dữ liệu về nhiệt độ, độ ẩm, áp suất… được thu thập tại vị trí địa lý nào đó và vào thời điểm x nào đó. Từ đó hãy xem xét hai ví dụ hai luật thu được sau:

Ví dụ 1: Nếu tại A đang mưa, thì tại A gió thổi từ hướng Đông.

Ví dụ 2: Nếu tại A đang mưa to, thì trong 1h tới tại B sẽ có mưa vừa.

Qua hai luật trên nếu khai phá theo các thuật toán cổ điển với các "intra-transaction" thì luật thu được không mang ý nghĩa. Vì thế chúng ta cần tạo được sự phụ thuộc giữa các "intra-transaction" với nhau thành liên giao dịch (inter-transaction). Trong phần II này, chúng tôi trình bày về một kỹ thuật mà Tung et al 2003 đã đề xuất [3].

Định nghĩa 1: Cho $I=\{a_1,a_2,...,a_k\}$ là tập các phần tử. D là thuộc tính thời gian, được đánh nhãn từ 0, 1,…n.

T là tập các giao dịch trong cơ sở dữ liệu.

Để đặc trưng cho mức độ phụ thuộc giữa các giao dịch chúng ta dùng một khái niệm là cửa sổ trượt.

Định nghĩa 2. Một cửa sổ trượt W kích thước w đặt trên một tập giao dịch nhằm chuyển đổi w giao dịch liên tiếp thành 1 giao dịch mới (w gọi là maxspan). W[0], W[1],..,W[w]

Định nghĩa 3. Tập phần tử mở rộng: $I'=\{a_1(0),...,a_1(w-1),a_2(0),...,a_2(w-1),...,a_k(0),...,a_k(w-1)\}$

Trong đó: $a_k(j)$ là phần tử a_k thuộc về khoảng W[j].

Định nghĩa 4. Liên giao dịch: $M=\{a_i(t)|a_i\in W[t];\ 1\le i\le k;\ 0\le t\le w-1\}$

Định nghĩa 5. Một luật kết hợp liên giao dịch có dạng $X\Rightarrow Y$ trong đó:

1. $X\subseteq I'$, $Y\subseteq I'$
2. $\exists a_i(0)\in X, 1\le i\le k$
3. $\exists a_i(j)\in Y, 1\le i\le k, j\ne 0$
4. $X\cap Y=\varnothing$

Định nghĩa 6. Cho T_{xy} là một liên giao dịch mà chứa $X\cup Y$ (X, Y là hai tập phần tử mở rộng). T_x là tập liên giao dịch chứa X. S là số lượng liên giao dịch trong cơ sở dữ liệu. Khi đó độ hỗ trợ (support) và độ tin cậy (confidence) của một luật kết hợp liên giao dịch là:

$$support=\frac{|T_{xy}|}{S}$$

$$confidence=\frac{|T_{xy}|}{|T_x|}$$

Định nghĩa 7. Cho $a_i(k)$ và $a_j(l)$ là hai item mở rộng.

Nếu $i=j,k=l$ thì $a_i(k)=a_j(l)$.

Nếu $(i=j,k<l)$ hoặc $(i<j)$ thì $a_i(k)<a_j(l)$.

Một đặc tính quan trong mà chúng ta sử dụng trong thuật toán là:

Đặc tính 1: Cho W là một cửa sổ trượt với w khoảng. Nếu 1-itemset $\{a_x(0)\}$ nào đó không phải là phổ biến thì bất kỳ 1-itemset $\{a_x(1)\},\{a_x(2)\},...,\{a_x(w)\}$ đều không phải tập phổ biến.

Chứng minh: Khi trượt cửa sổ dọc theo các giao dịch trong cơ sở dữ liệu, khi đó $a_x(0)$ sẽ xuất hiện trong W[0] mỗi khi trượt, tuy nhiên $a_x(t)$ ***có thể*** sẽ xuất hiện trong W[t] mà thôi. Do đó:

$$support(\{a_x(t)\})\le support(\{a_x(0)\})$$

Mặt khác, nếu $\{a_x(0)\}$ không phải là tập phổ biến thì $support(\{a_x(0)\}) \le \min\_sup$, từ đó

$support(\{a_x(t)\}) \le support(\{a_x(0)\}) \le \min\_sup$ từ đó suy ra điều phải chứng minh.

III. THUẬT TOÁN KHAI PHÁ ĐỀ XUẤT

Với thuật toán Apriori, chi phí sinh và kiểm tra tập các ứng viên là rất lớn cho nên ảnh hưởng đến tốc độ tính toán. Vì vậy để tránh điều đó, chúng tôi tiếp cận bài toán theo hướng tiếp cận tập cha là phổ biến thì tất cả các tập con của nó cũng phải là tập phổ biến. Hướng giải quyết đó là tìm kiếm tất cả những tập phổ biến tối đa cho mục tiêu tìm kiếm. Sau đó, sử dụng lưu trữ cơ sở dữ liệu dưới dạng lưu trữ Tid của mỗi bản ghi cho mỗi phần tử.

Bắt đầu với mỗi phần tử $a_k(i)$, nếu chúng ta lưu trữ tập chỉ số các giao dịch chứa nó, và trên mỗi giao dịch T(X) chứa $a_k(i)$ chúng ta lại tìm kiếm các phần tử $a_k(j)$ mà có thể trở thành phổ biến. Chiến lược dò từng bước và mở rộng như thế không sinh ra tập các ứng viên như Apriori. Đặc biệt việc kiểm tra chỉ thực hiện trên một kích thước cơ sở dữ liệu nhỏ hơn nhiều so với toàn bộ dữ liệu. Từ đó tiết kiệm chi phí.

Để hiện thực hóa ý tưởng, chúng tôi xây dựng một cấu trúc cây mà chúng tôi gọi là cây phần tử. Mỗi node có dạng $\langle X|Y \rangle$. Trong đó X *(head)* là tập các phần tử phổ biến mà chúng ta đã kiểm tra là phổ biến, *Y (tail)* là tập các phần tử còn lại chưa được xét. T(X) là tập tất cả những giao dịch chứa X. *(Khi $X = \varnothing$ thì T(X) chính là toàn bộ giao dịch trong cơ sở dữ liệu).* Bên cạnh đó tại mỗi nút chúng tôi sử dụng một danh sách *maximal_element* lưu trữ các phần tử nằm trong tập maximal mà dựa vào đó để quyết định việc có phải tạo nút mới hay không.

Thuật toán

Input: Tập các phần tử $I' = \{a_1(0),...,a_1(w-1),a_2(0),..., a_2(w-1),...,a_k(0),..., a_k(w-1)\}$, tập các giao dịch T.

Output: Tập các phần tử phổ biến lớn nhất

```
List<Item> findMaximal(Node node){
 //Từ tập giao dịch T(X) chứa X: Loại bỏ những phần tử trong Y không thỏa mãn ngưỡng
 1.     for(item i: Y){
 2.             for(Transaction t: T(X)){
3.                     if(t.contain(i)
4.                             count_the_support(i)++;
5.             }
 6.             if(count_the_support(i) < min support)
 7.                     Y.remove(i);
 8.     }
 9.     while (Y vẫn còn phần tử){
 10.            //Nếu Y nằm trong maximal_element thì không cần tạo nút mới
 11.            if(maximal_element.contain(Y))
 12.                    break;
 13.            //chọn phần tử a[i] đầu tử đầu tiên của Y
 14.            //Tạo nút mới
 15.            Node next_node(X = a[i], Y = Y \ a[i]);
 16.            //Đệ quy tạo nút mới
 17.            findMaximal(next_node);
 18.            //Cập nhật số phần tử trong Y
 19.            Y của node = Y của (next_node);
 20.    }
 21.    return node.maximal_element = node.maximal_element.add(X);
```

Ở thuật toán trên, các phần tử trong Y được sắp xếp theo định nghĩa 7, nghĩa là nó có dạng: $I' = \{a_1(0),a_2(0),...,a_k(0),a_1(1),a_2(1),...,a_1(m),...,a_k(m)\}$. Việc sắp xếp này có ý nghĩa rất lớn do từ định nghĩa 5.2, mỗi tập phổ biến phải chứa ít nhất một phần tử $a_k(0)$, nên sẽ bắt đầu việc mở rộng từ những phần tử có dạng $a_k(0)$, mở rộng liên tục với các phần tử còn lại trong *I'*. Nếu sắp xếp các phần tử $a_k(i) \in I'$ theo định nghĩa 7 thì khi kết thúc duyệt các phần tử dạng $a_k(0)$, có thể kết thúc việc tạo cây, mà không cần quan tâm đến các phần tử còn lại.

Tại nút gốc: X = null, Y = $I' = \{a_1(0),a_2(0),...,a_k(0),a_1(1),a_2(1),...,a_1(m),...,a_k(m)\}$

Dòng 2, 3, 4 thực hiện việc tính toán support của các phần tử. Tuy nhiên, nhờ đặc tính 1 mà chúng ta đề cập ở trên, nếu $a_k(0)$ không là phổ biến thì ta có thể loại bỏ toàn bộ phần tử dạng $a_k(i)$ ra khỏi cơ sở dữ liệu, giúp làm giảm kích thước của dữ liệu.

Việc tạo nút (dòng 14) từ nút cha, theo cơ chế, phần tử đầu tiên trong Y của nút cha sẽ là X của nút con, phần còn lại của Y nút cha sẽ là Y của nút con. Tuy nhiên việc có cần tạo nút hay không dựa vào điều kiện dòng 11, nghĩa là nếu Y của nút có thể được tạo là tập con của tập *maximal_element* của nút cha thì nó không cần tạo, vì hiển nhiên mọi tập con của tập phổ biến đều là phổ biến. Nếu nút được tạo, nó sẽ tự cập nhật lại Y của mình (dòng 6,7) có nghĩa là loại bỏ những phần tử có độ support trong tập giao dịch T(X) nhỏ hơn ngưỡng. Thuật toán dừng lại khi không có nút nào được tạo.

Để cụ thể hóa thuật toán, chúng tôi lấy một minh họa đơn giản cho thuật toán như sau:

Giả sử ta có một cơ sở 5 giao dịch với 4 phần tử: a, b, c, d. Cho ngưỡng support = 40% (supp = 2), kích thước cửa sổ trượt bằng 2, khi đó số phần tử tăng từ 4 lên 8. (Hình 1)

Tid	Giao dịch
0	a, b
1	a, c
2	c
3	a, b
4	c, d

w = 2

Tid	Giao dịch
0	a[0], b[0], a[1], c[1]
1	a[0], c[0], c[1]
2	c[0], a[1], b[1]
3	a[0], b[0], c[1], d[1]
4	c[0], d[0]

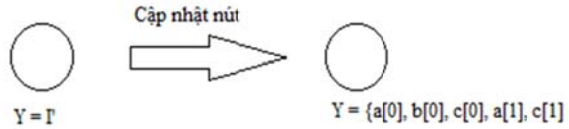
Hình 1. Chuyển đổi giao dịch

Tập các phần tử mở rộng I' = {a[0], b[0], c[0], d[0], a[1], b[1], c[1], d[1]}

Việc sinh nút gốc bắt đầu với X = null, Y = I', T(null) = {0,1,2,3,4,5}

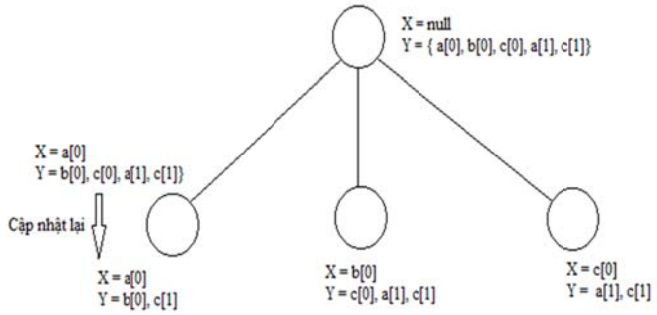
Khi mỗi nút được sinh ra, chúng tự động cập nhật để loại bỏ phần tử không phải là phổ biến, như hình 2. Trên tập các giao dịch T(null): supp(a[0]) = 3, supp(b[0]) = 2, supp(c[0]) = 3, supp(d[0]) = 1, supp(a[1]) = 2, supp(b[1]) = 1, supp(c[1]) = 2, supp(d[1]) = 1

Với min supp = 2, thì nút gốc tự động cập nhật: X = null, Y = { a[0], b[0], c[0], a[1], c[1]} (Hình 2)

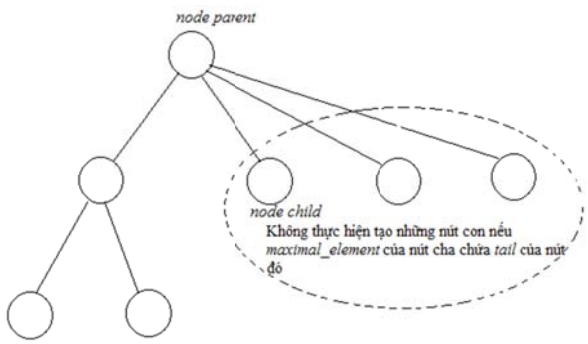


Hình 2. Cập nhật lại nút

Với mỗi $\{a_x(i)\} \in Y$ tạo nút con (dòng 11) được thể hiện như hình 3:

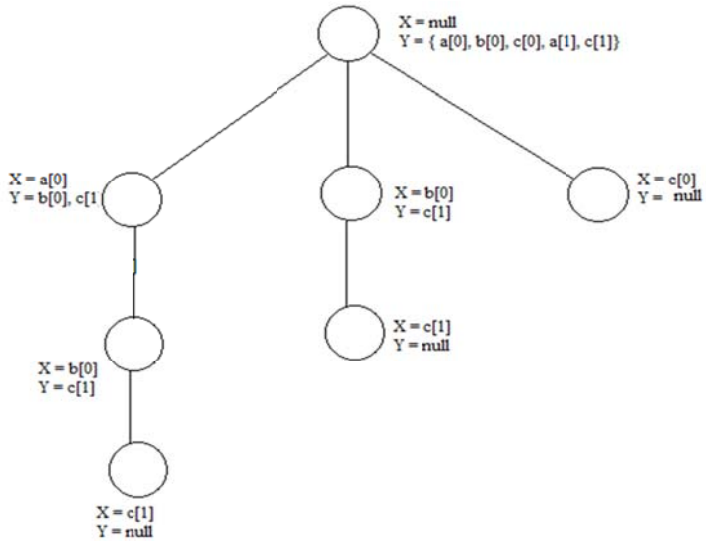


Hình 3. Tạo nút mới



Hình 4. Điều kiện tạo nút mới

Kết thúc việc tạo cây nếu Y của tất cả các nút đã tạo giống như trên hình 5.

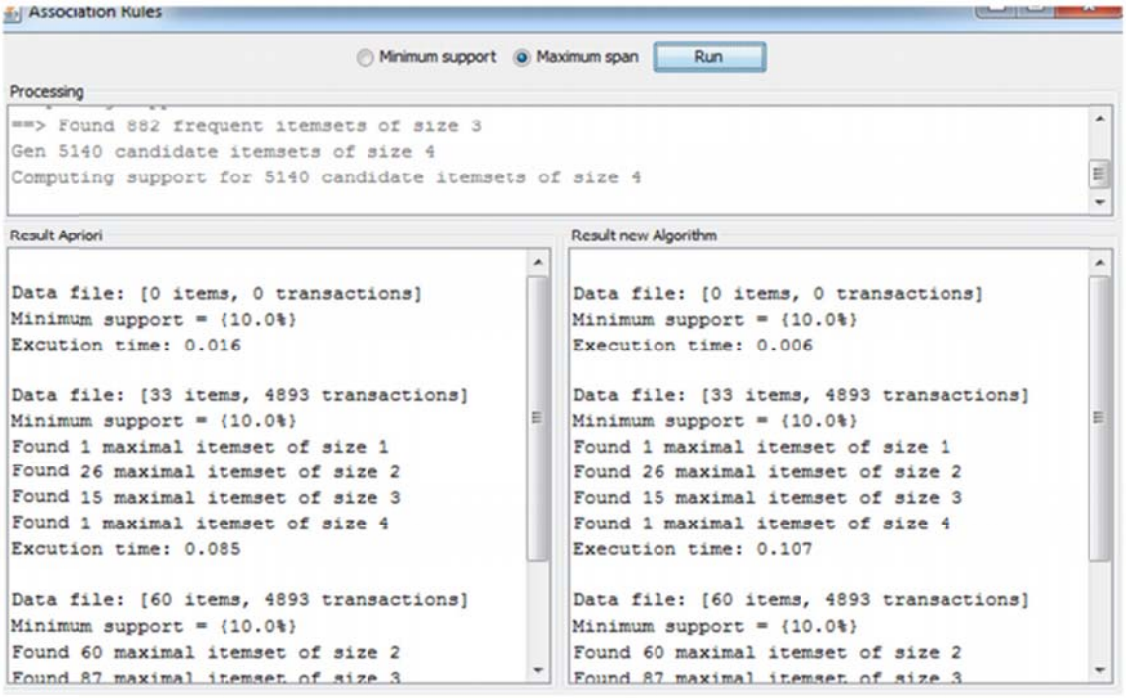


Hình 5. Cây khi tạo xong

Khi cây được hoàn thành thì *maximal_element* của nút gốc sẽ là tập phần tử phổ biến lớn nhất.

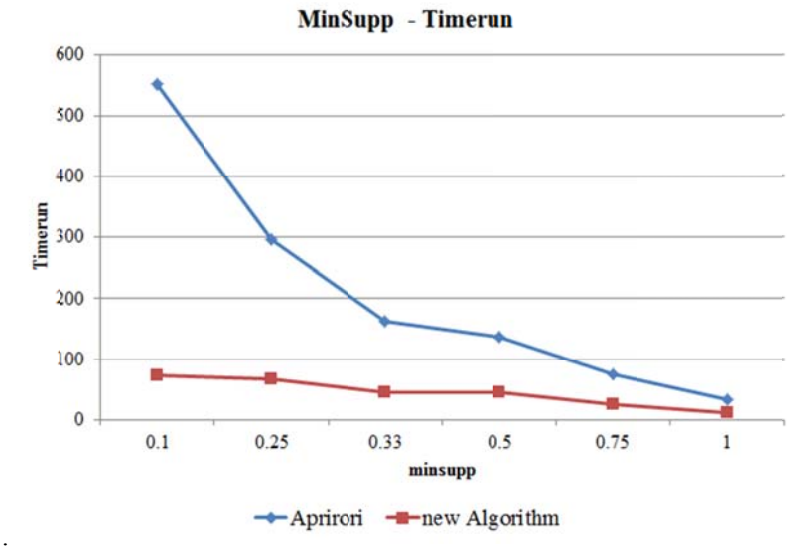
IV. KẾT QUẢ THỬ NGHIỆM

Chúng tôi tiến hành thu thập bộ dữ liệu về thời tiết tại Hà Nội theo từng giờ trong vòng 3 năm (2008-2010) và theo từng ngày trong 15 năm (2000 - 2014) trên website: http://www.wunderground.com. Bộ dữ liệu gồm 8 thuộc tính: Temp, Humidity, Pressure, Visibility, Wind Direct, Wind Speed, Events, Conditions. Để xử lý tiền dữ liệu, chúng tôi thực hiện làm đầy dựa trên giá trị của các giá trị lân cận cùng thuộc tính mà theo thời gian. Bằng cách này, dữ liệu được đưa vào khai phá sẽ không có bản ghi nào bị thiếu và trên không gian khai phá sẽ không chứa lỗ hổng dữ liệu nào. Ở đây, chúng tôi lựa chọn khai phá dọc theo trục thời gian tức là các bản ghi sẽ phụ thuộc vào nhau theo biến thời gian. Sau đó chúng tôi tiến hành tiền xử lý bộ dữ liệu, rời rạc hóa các dữ liệu loại định lượng như: Temp, Humidity, Pressure, Visibility, Wind Speed bằng công cụ Weka 3.6.9. Chương trình thực nghiệm mô tả như hình 6.



Hình 6. Chương trình thực nghiệm so sánh thuật toán đề xuất với Aprori

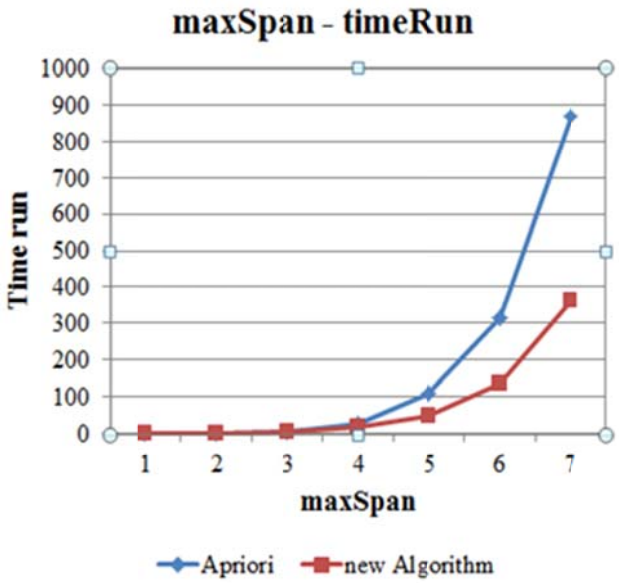
Trong thí nghiệm thứ nhất chúng tôi chạy giải thuật với một của sổ trượt kích thước bằng 3, cho các ngưỡng support khác nhau trên bộ dữ liệu thời tiết thu thập theo giờ và so sánh thời gian chạy với thuật toán Apriori. Kết quả chạy thực nghiệm được thể hiện trên hình 7.



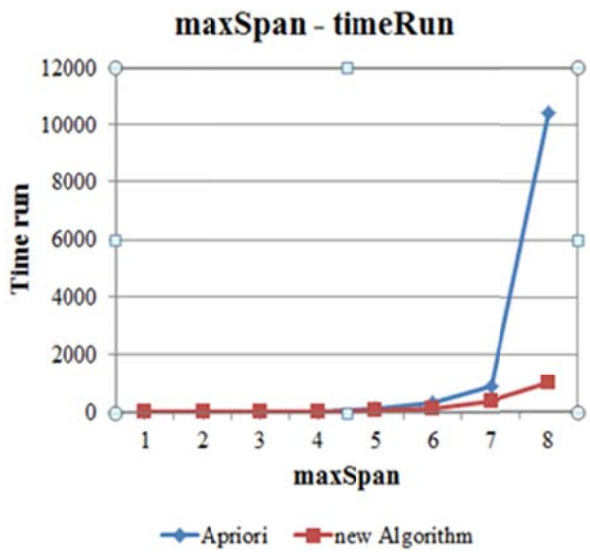
Hình 7. Kết quả theo MinSupp - TimeRun

Kết quả thực nghiệm cho thấy với cùng bộ dữ liệu và cùng ngưỡng support đủ nhỏ thì thời gian chạy của thuật toán cải tiến vượt trội hơn so với thuật toán Apriori.

Trong thí nghiệm thứ hai chúng tôi chạy trên bộ dữ liệu thời tiết mà được thu thập theo ngày, bên cạnh đó xác định một ngưỡng support duy nhất bằng 10 và thay đổi kích thước cửa sổ trượt từ 1 đến 8. Thuật toán cải tiến được so sánh với thuật toán Apriori, kết quả thực nghiệm cho thấy như sau: Với kích thước cửa sổ nhỏ (1 - 4) thì sự khác biệt là không đáng kể, sự khác biệt rõ ràng khi tăng maxspan từ 5 – 7, kết quả mô tả như hình 8a; Với kết quả thực nghiệm khi cho giảm maxspan thì thời gian của Apriori là kém hiệu quả hơn rất nhiều so với thuật toán đề xuất, mô tả như hình 8b.



Hình 8a.



Hình 8b.

Trong quá trình thực nghiệm trên bộ dữ liệu lớn thời tiết ước tính vài chục nghìn đến vài tỷ bản ghi, kết quả thực nghiệm cho thấy thuật toán đề xuất thực hiện tốt chi phí thời gian khi so sánh với thuật toán Apriori. Đặc biệt, thời gian giảm đáng kể đối với thuật toán đề xuất với Apriori khi cùng thực hiện trên bộ dữ liệu lớn. Như vậy, thuật toán đề xuất sử dụng hiệu quả trong khai phá dữ liệu dựa vào các luật đối với bộ dữ liệu lớn.

V. KẾT LUẬN

Trong bài báo chúng tôi đã đề xuất một cách tiếp cận mới nhằm giải quyết vấn đề tìm kiếm các tập phần tử phổ biến mà dựa trên việc mở rộng tìm kiếm tập phần tử phổ biến lớn nhất. Điều này khắc phục được nhược điểm lớn nhất của những thuật toán sinh và kiểm tra tập ứng viên như Apriori, giúp cải thiện hiệu năng do cắt giảm chi phí quét cơ sở dữ liệu. Thuật toán đề xuất này không chỉ giải quyết cho các bộ dữ liệu không gian và thời gian mà có thể được mở rộng và áp dụng cho nhiều bộ dữ liệu khác nhau, đặc biệt áp dụng thuật toán đề xuất này hiệu quả cho các thực nghiệm khai phá dữ liệu lớn.

Hướng nghiên cứu tiếp theo của nhóm dự kiến tiến hành thử nghiệm trên cửa sổ trượt hai chiều và xa hơn là đa chiều (dữ liệu được gắn kèm theo đa thuộc tính), xây dựng mô đun tiền xử lý dữ liệu đầu vào cho các bộ dữ liệu thưa, bộ dữ liệu có tính liên tục về thời gian. Để thực hiện việc này, chúng tôi cần thực hiện các mô đun sử dụng các hàm lượng hóa tham số đếm được và tham số không đếm được trong bộ dữ liệu lớn.

VI. TÀI LIỆU THAM KHẢO

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.

[2] Hongjun, Ling Feng, Jiawei Han, Beyond Intra-Transaction Association Analysis:Mining Multi-Dimensional Inter-Transaction Association Rules,

[3] Anthony K. H. Tung, Hongjun Lu, Jiawei Han, Ling Feng, Efficient Mining of Inter-transaction Association Rules, IEEE Transactions On Knowledge And Data Engineering, Vol. 15, No. 1; January/February 2003, pp. 43-56

[4] Anthony J.T. Lee *, Chun-Sheng Wang, An efficient algorithm for mining frequent inter-transaction patterns , 2007

[5] Yo-Ping Huang, Li-Jen Kao, Frode-Eika Sandnes, Efficient mining of salinity and temperature association rules from ARGO data.

[6] Yo-Ping Huang and Jung-Shian Jau, Frode Eika Sandnes, Temporal-Spatial Association Analysis of Ocean Salinity and Temperature Variations.