CẢI TIẾN THUẬT TOÁN KHAI PHÁ DỮ LIỆU TUẦN TỰ CMSPAM CHO TRƯỜNG HỢP DỮ LIỆU THƯA

Nguyễn Mạnh Sơn [*], Đặng Ngọc Hùng[+]
[*] Khoa CNTT1 – Học Viện Công Nghệ Bưu Chính Viễn Thông
Email: sonnm@ptit.edu.vn
[+] Khoa CNTT1 – Học Viện Công Nghệ Bưu Chính Viễn Thông
Email: hungdn@ptit.edu.vn

***Abstract* — Khai phá mẫu tuần tự (SPM) được ứng dụng rộng rãi trong các bài toán thương mại điện tử và ra quyết định. Các thuật toán SPM tiêu biểu đã được áp dụng trong nhiều hệ thống tư vấn, dự báo … như GSP, SPAM, CMSPAM. Bài báo sẽ phân tích ưu nhược điểm của các thuật toán và đề xuất một cải tiến cho thuật toán CMSPAM. Thuật toán cải tiến được đặt tên là CMSPAME cho hiệu quả tốt hơn đối với trường hợp dữ liệu thưa và vẫn giữ nguyên được hiệu năng như thuật toán CMSPAM trong các trường hợp khác.**

Keywords— Khai phá dữ liệu tuần tự, SPM, cải tiến CMSPAM, thuật toán CMSPAME.

I. GIỚI THIỆU

Bài toán khai phá mẫu tuần tự (Sequential Pattern Mining - SPM) được R. Agrawal và R. Srikant giới thiệu vào năm 1995 [1]. Cho một tập các dãy tuần tự, trong đó mỗi dãy bao gồm một tập các giao dịch, và mỗi giao dịch bao gồm một tập các phần tử, cùng một ngưỡng phổ biến (minsup), khai phá mẫu tuần tự tìm ra tất cả các chuỗi (subsequence) phổ biến, là dãy tuần tự xuất hiện trong tập dữ liệu với tần số không nhỏ hơn ngưỡng phổ biến. SPM ngày càng được sử dụng rộng rãi trong thương mại điện tử (phân tích, dự báo xu hướng mua sắm, quản lý kho hàng, …) và cũng được ứng dụng hiệu quả cho nhiều lĩnh vực khác như phân tích DNA, tư vấn điều trị bệnh, dự báo thiên tai, phân tích mẫu truy cập website ….

Phần lớn các thuật toán ban đầu cho bài toán khai phá mẫu tuần tự đều dựa trên tính chất Apriori được sử dụng trong khai phá luật kết hợp ([1],[2],[3]). Tính chất này cho rằng: mọi mẫu con (sub-pattern) của một mẫu phổ biến (frequent pattern) cũng chính là một mẫu phổ biến. Dựa trên tính chất này, rất nhiều các thuật toán được đề xuất như: AprioriAll, AprioriSome, DynamicSome (Agrawal và Srikan 1995), GSP (Skrikant và Agrawal 1996) với phương pháp định dạng bộ nhớ theo chiều ngang *(horizontal database format)* ([2],[3]). Tuy nhiên khi các CSDL ngày càng lớn, thì phương pháp định dạng bộ nhớ theo chiều ngang tỏ ra thiếu hiệu quả [3]. Các phương pháp định dạng bộ nhớ theo chiều dọc *(vertical database format)* mà tiêu biểu là thuật toán SPAM (Sequential PAttern Mining using A Bitmap Representation) [4] với ý tưởng chính là sử dụng bitmap để lưu trữ CSDL đồng thời hỗ trợ tính toán giá trị hỗ trợ mà không phải quét lại CSDL. Các thử nghiệm cho thấy SPAM tìm được toàn bộ kết quả trùng khớp với thuật toán GSP nhưng với tốc độ nhanh hơn đáng kể [4].

Các thuật toán sử dụng bitmap sau này như

CMSPAM(2014) và CMSPADE(2014) đều dựa trên ý tưởng của SPAM ([5],[6],[7]).

Cơ sở dữ liệu bitmap theo chiều dọc *(Vertical Database Bitmap-VDB)* có thể được hiểu đơn giản là một CSDL mà mỗi hàng đại diện cho một item và đưa ra danh sách thứ tự xuất hiện của item đó trong CSDL

Thuật toán SPAM có 3 điểm đáng chú ý [4]:

SPAM sử dụng bitmap để lưu trữ cơ sở dữ liệu theo chiều dọc: đặc điểm này giúp tính toán giá trị hỗ trợ cho mỗi item một cách nhanh chóng mà không cần duyệt lại toàn bộ cơ sở dữ liệu như các thuật toán sử dụng cơ sở dữ liệu theo chiều ngang. Việc sử dụng bitmap để lưu trữ dữ liệu giúp giảm kích thước bộ nhớ và tăng khả năng tính toán cho các phép cắt tỉa chuỗi tuần tự của thuật toán.

SPAM sử dụng các phép mở rộng S-step, I-Step và các phép cắt tỉa S-Step Pruning, I-Step Pruning để tăng tốc độ xử lý. Phương pháp này giúp cho thuật toán sinh ra ít ứng cử viên hơn và vẫn đảm bảo tính chính xác. SPAM kiểm tra các ứng cử thỏa mãn giá trị minsup một cách nhanh chóng thông qua các phép toán trên dãy bit.

Tập ứng cử của thuật toán SPAM vẫn chưa tối ưu. Tuy giảm thiểu được số lượng lớn các ứng viên sinh ra sau mỗi bước nhờ các phép mở rộng, tập ứng cử của thuật toán SPAM vẫn chứa nhiều giá trị không phổ biến, hoặc không xuất hiện trong CSDL.

Năm 2014 các nhà khoa học P. Fournier-Viger, A. Gomariz, M. Campos và Rincy Thomas đã đề xuất một thuật toán mới có tên CMSPAM, khắc phục được những nhược điểm của thuật toán SPAM [5]. Bài báo sẽ tập trung đi sâu phân tích và thử nghiệm thuật toán CMSPAM, sau đó đề xuất một cải tiến thuật toán này cho kết quả về hiệu năng tốt hơn trong trường hợp dữ liệu thưa.

Vì sao trường hợp dữ liệu thưa ngày càng được quan tâm, nhất là trong các bài toán tư vấn cho thương mại điện tử? Trong các hệ thống thương mại điện tử nói chung, số lượng người dùng ngày càng lớn và còn tiếp tục tăng nhanh trong thời gian tới. Tuy nhiên, tỉ lệ người dùng thực hiện nhiều giao dịch không lớn và các giao dịch của một người dùng có thể cách xa nhau về mặt thời gian. Và như vậy, nói chung các hệ thống thương mại điện tử sẽ đều gặp phải trường hợp dữ liệu thưa, tức là số giao dịch trung bình trên một người dùng và số sản phẩm trung bình trong một lần giao dịch không cao.

Thuật toán cải tiến được chúng tôi đặt tên là CMSPAME đưa ra một số thay đổi riêng cho trường hợp dữ liệu thưa. Các thử nghiệm đã được thực hiện trên bộ dữ liệu chuẩn của P. Fournier-Viger [8] và cho ra kết quả tốt hơn khá rõ ràng về mặt hiệu năng (thời gian chạy thuật toán).

II. THUẬT TOÁN KHAI PHÁ DỮ LIỆU CMSPAM

Thuật toán CMSPAM được đưa ra với mục tiêu giảm bớt số lượng ứng cử được sinh ra trong mỗi bước mà vẫn đảm bảo kết quả đúng đắn.

Thay vì phải sinh ra tập ứng cử sau mỗi bước mở rộng của thuật toán SPAM, CMSPAM sẽ sinh ra tập ứng cử có thể cho mỗi item ngay sau khi quét CSDL mà vẫn đảm bảo không bỏ sót bất cứ ứng viên thích hợp nào. Như vậy CMSPAM sẽ làm giảm chi phí bộ nhớ cũng như giảm thời gian thực hiện của thuật toán.

Thuật toán CMSPAM

Input	Một cơ sở dữ liệu tuần tự S và giá trị ngưỡng phổ biến
Output	Tập đầy đủ các mẫu tuần tự F

Pamameters:
S: Tập dữ liệu
Minsup: Giá trị ngưỡng phổ biến

Method: Nội dung hàm SPAM(minsup, S)

```
// Bước1: Quét Cơ sở dữ liệu SDB để tạo cơ sở dữ liệu theo chiều dọc VDB
sid = tid = 0;

FOR(each item s∈SDB){
        IF(s is end of transation){
           tid++

        }ELSE IF(s is end of sequence){
           sid++
           tid = 0

        }ELSE{
           bitmapItem = VDB.get[s]

                    IF(bitmapItem       =
NULL){
                    VDB.add(s,new
```

```
       bitmap())

                    }
            bitmapItem.registerBit(sid,
       tid);

           }
}

// Bước 2: Quét Cơ sở dữ liệu chiều dọc
VDB để loại bỏ những item không phổ
biến, tập F chứa danh sách các item phổ
biến .
FOR(each item s ∈VDB){
         IF(s is frequent) {
            F = F ∪ s

         }ELSE
           VDB remove s

         }

// Bước 3 : Thực hiện khởi tạo CMAP
CREATECMAP(SDB, F, minsup)

// Bước 4 : Thực hiện mở rộng và cắt tỉa
chuỗi phổ biến

FOR(each item s ∈F)
        DFS-Pruning(<s>,        CMAPi[s],
CMAPs[s],s)
```

Bảng 1: Thuật toán CMSPAM

CMPSPAM bổ sung khái niệm *cơ sở dữ liệu đồng thời CMAP*: là một cấu trúc ánh xạ mỗi item k ∈ I với tập item được mở rộng của k [5]. Thuật toán định nghĩa hai CMAP là CMAPi và CMAPs.

- $CMAP_i$ ánh xạ mỗi item k với tập $cm_i(k)$ chứa tất cả cái item j ∈ I, j là item được mở rộng bằng phép mở rộng i-step và giá trị hỗ trợ không nhỏ hơn minsup.
- $CMAP_s$ ánh xạ mỗi item k với tập $cm_s(k)$ chứa tất cả cái item j∈ I, j là item được mở rộng bằng phép mở rộng s-step và giá trị hỗ trợ không nhỏ hơn minsup.

Thuật toán CMSPAM và ý nghĩa từng bước của thuật toán đã trình bày trong Bảng 1.

Trong bước 3 của thuật toán CMSPAM, hàm CREATECMAP sẽ được gọi để tạo CSDL đồng thời CMAP. Thủ tục này được trình bày trong Bảng 2. Với mỗi item trong chuỗi giao dịch, thuật toán sẽ sử dụng phép mở rộng i-step hoặc s-step để bổ sung vào CMAPi hoặc CMAPs.

Trong bước 4 của thuật toán, thủ tục cắt tỉa DFS-Pruning được gọi đến. Bảng 3 trình bày thủ tục này. Bản chất thủ tục này là một thao tác duyệt theo chiều sâu kiểu đệ quy có phân nhánh dựa trên việc xét phần tử thuộc về một trong hai tập S và I đã được xây dựng từ thủ tục CREATECMAP ở trên.

Hàm CREATECMAP(SDB, F)	
Input	Một cơ sở dữ liệu tuần tự SDB và tập các item phổ biến F
Output	Tập ứng cử $CMAP_i$, CMAPs

Pamameters: Ý nghĩa các tham số
1. SDB : tập cơ sở dữ liệu tuần tự
2. F: tập các item phổ biến

```
CREATECMAP()
CMAPi= CMAPs= ∅
FOR transaction k ∈ VDB{
   equalSet = ∅
     FOR item i ∈ k {
        IF i∉ equalSet
           equalSet add i
        IF i∉ F
           Continue;
        FOR item j>i ∈ k{
           IF j∉ F
             Continue;
             IF i,j ∈ same itemset{
               IF i≠ j
                 CMAPi[i] add j
                 support++
                 equalSet add j
                           }
               ELSE{
                 CMAPs[i] add j
                 support++;
               }
         }
    }
}
```

Bảng 2: Hàm CREATECMAP

Hàm DFS-Pruning(<s>, S_n, I_n,k)	
Input	Chuỗi tiền tố s, tập ứng cử S_n, I_n

Pamameters: Ý nghĩa các tham số
1. s: chuỗi tiền tố hiện tại
2. S_n: Tập ứng cử theo phép mở rộng S-Step
3. I_n : Tập ứng cử theo phép mở rộng I-Step
4. k: là item cuối cùng trong chuỗi <s>

DFS-Pruning $((s_1,s_2 \ldots, s_k),S_n, I_n,k)$

```
S_temp = Ø
I_temp = Ø
FOR (each i ∈ S_n)
    IF ((s_1, …, s_k,{i}) is frequent & i ∈ CMAP_s[k] & support(i) >= minsup)
      S_temp = S_temp ∪ {i}
FOR (each i ∈ S_temp)
     DFS-Pruning((s_1, …, s_k,{i}) , S_temp , S_temp )
FOR (each i ∈ I_n)
     IF ((s_1, …, s_k,{i}) is frequent & & i ∈ CMAP_i[k]& support(i) >= minsup)
       I_temp = I_temp ∪ {i}
FOR (each i ∈ I_temp )
     DFS-Pruning((s_1, …, s_k,{i}) , S_temp , I_temp )
```

Bảng 3: Hàm DFS-Pruning

Để đánh giá hiệu quả của CMSPAM, chúng tôi thử nghiệm trên bộ dữ liệu SPMF [8]. Cách thức thử nghiệm và đánh giá tương tự như trong [6].

Tất cả các tập dữ liệu dùng để kiểm thử thuật toán đều tuân theo cấu trúc sau:

<T> <end of transaction> …….<T> <end of transaction> <end of sequence>

Trong đó :

<T>: Là một giao dịch, các item trong cùng một giao dịch phân cách nhau bởi dấu cách.

<end of transaction> : Là một chữ số dùng để phân biệt các giao dịch với nhau (trong dữ liệu chọn số -1).

<end of sequence >: Là một chữ số dùng để ký hiệu kết thúc một chuỗi tuần tự (trong dữ liệu chọn số -2).

Việc kiểm thử các thuật toán sẽ sử dụng 3 CSDL được liệt kê trong Bảng 4. Các thử nghiệm sẽ thực hiện trên từng bộ dữ liệu và thống kê thời gian chạy của thuật toán khi thay đổi ngưỡng minsup. Dựa trên kết quả, chúng tôi sẽ vẽ biểu đồ so sánh hiệu năng thuật toán với trục hoành là ngưỡng minsup (giảm dần), trục tung là thời gian. Chúng tôi lựa chọn ba thuật toán để so sánh gồm GSP, SPAM và CMSPAM.

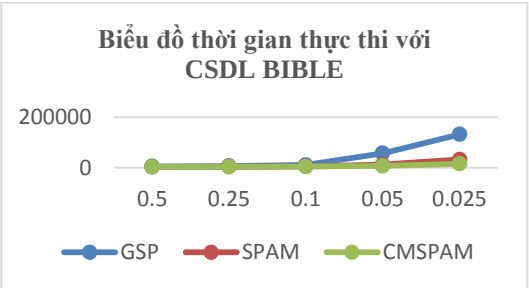
Data	Số chuỗi	Số sản phẩm	Số sản phẩm TB trong chuỗi	Số sản phẩm TB khác nhau trong chuỗi
Bible	36369	13905	21.6	17.84
KDDcup2000	77512	3340	4.62	6.07
MSNBC	31790	17	13.33	5.33

Bảng 4: Thống kê tập dữ liệu kiểm thử

Thử nghiệm 1 với CSDL BIBLE (Bảng 5 và Hình 1) cho thấy đối với CSDL có đặc trưng là có nhiều sản phẩm mà mỗi chuỗi đều gồm nhiều giao dịch, nhiều sản phẩm thì SPAM và CMSPAM nhanh hơn rất nhiều so với thuật toán GSP do không phải xử lý một số lượng lớn các mẫu sinh ra tại mỗi lần lặp đồng thời không phải quét CSDL nhiều lần. Giá trị minsup càng nhỏ thì thuật toán CMSPAM càng tỏ ra hiệu quả hơn so với thuật toán SPAM do hạn chế được số ứng cử viên sinh ra sau mỗi lần lặp.

Minsup	0.5	0.25	0.1	0.05	0.025
GSP	5421	7015	11114	57266	132568
SPAM	3615	4025	5316	12330	32756
CMSPAM	3531	3858	4831	7666	15529
Result Set	5	22	174	774	3285

Bảng 5: Thử nghiệm CMSPAM với BIBLE

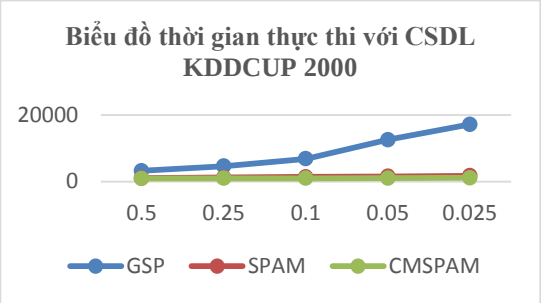


Hình 1: Biểu đồ kết quả kiểm thử thuật toán GSP, SPAM, CMSPAM với bộ dữ liệu BIBLE

Thử nghiệm 2 trên KDDCUP 2000 với 77512 khách hàng, 3340 sản phẩm, mỗi khách hàng có trung bình 6.07 sản phẩm giao dịch với 4.62 sản phẩm khác nhau. Kết quả cho trong Bảng 6 và Hình 2 cũng tương đồng như thử nghiệm với BIBLE. CMSPAM luôn cho kết quả tốt hơn về thời gian.

Minsup	0.5	0.25	0.1	0.05	0.025
GSP	3268	4661	6875	12548	17230
SPAM	1025	1241	1446	1552	1684
CMSPAM	958	971	987	1037	1124
Result Set	0	0	0	0	10

Bảng 6: Kết quả kiểm thử CMSPAM với tập dữ liệu KDDCUP 2000

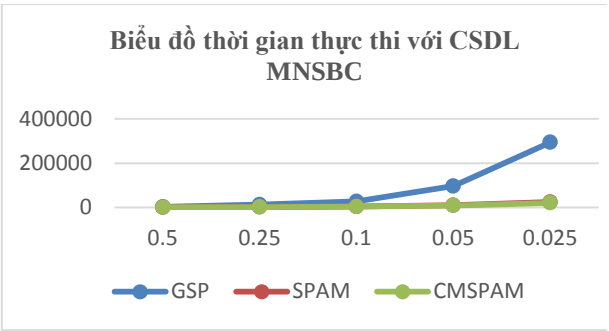


Hình 2: Kiểm thử thuật toán GSP, SPAM, CMSPAM với bộ dữ liệu KDDCUP2000

Thử nghiệm 3 trên MSNBC với 31790 khách hàng, 17 sản phẩm, mỗi khách có trung bình 13.33 sản phẩm giao dịch với 5.33 sản phẩm khác nhau. Kết quả thử nghiệm trong Bảng 7 và Hình 3.

Minsup	0.5	0.25	0.1	0.05	0.025
GSP	2523	12053	26754	96054	293921
SPAM	1121	1666	4038	9701	22800
CMSPAM	748	1361	3465	8972	20047
Result Set	5	44	338	1478	6068

Bảng 7. Kiểm thử với tập dữ liệu MNSBC



Hình 3. Kiểm thử thuật toán GSP, SPAM, CMSPAM với bộ dữ liệu MNSBC

Các thử nghiệm cho thấy tốc độ của SPAM và CMSPAM vẫn nhanh hơn rất nhiều so với thuật toán GSP trong cả hai trường hợp số chuỗi tuần tự nhiều hay ít. Và thuật toán CMSPAM luôn cho kết quả hiệu năng tốt hơn SPAM.

III. THUẬT TOÁN CMSPAME

Sau khi cài đặt và thử nghiệm thuật toán CMSPAM, chúng tôi nhận thấy trong bước 1 của thuật toán: *quét cơ sở dữ liệu SDB để tạo cơ sở dữ liệu theo chiều dọc VDB*, thuật toán sẽ quét toàn bộ CSDL để tính toán giá trị hỗ trợ của từng item. Ngay sau đó trong bước 2 thuật toán sẽ loại bỏ những item có giá trị hỗ trợ nhỏ hơn minsup. Những item còn lại sẽ là những item phổ biến. Như vậy trong bước 1 việc quét và tính toán giá trị hỗ trợ của các *item không phổ biến* là không hiệu quả vì ngay trong bước 2 những item này sẽ bị loại bỏ và không được sử dụng trong phần còn lại của thuật toán.

Chúng tôi nhận định việc giảm thiểu những phép duyệt và tính toán giá trị hỗ trợ của các item này sẽ cải thiện được tốc độ xử lý của thuật toán. Đặc biệt là với những bộ dữ liệu thưa, tức là CSDL chứa một số lượng lớn các item và chuỗi tuần tự nhưng số giao dịch trung bình với một khách hàng là thấp.

Câu hỏi đặt ra là làm sao chúng ta biết những item nào là item phổ biến, những item nào là không phổ biến trong khi chưa tính toán được giá trị hỗ trợ của các item này.

Để giải quyết câu hỏi này, chúng tôi đề xuất một cải tiến cho thuật toán CMSPAM dựa trên ý tưởng đánh giá cận trên và đánh dấu để loại bỏ việc phải tính toán lại với các item không thể cho giá trị tốt hơn ngưỡng minsup. Cụ thể:

- Trong mỗi bước duyệt item từ lần quét CSDL đầu tiên, chúng ta sẽ tính toán giá trị hỗ trợ lớn nhất có thể đối với item đó (giả sử item đó xuất hiện trong tất cả các chuỗi tuần tự còn lại).
- Nếu giá trị hỗ trợ tốt nhất này không thể vượt quá giá trị minsup thì item đó chắc chắn sẽ là item không phổ biến và việc tính toán giá trị hỗ trợ thực tế của item trong các chuỗi tuần tự còn lại là không cần thiết.
- Gắn thêm một thuộc tính đánh dấu (Flag) trong mỗi đối tượng item.
  - Flag có giá trị bằng false có nghĩa là item đó không thể là một item phổ biến và sẽ không được tiếp tục tính toán giá trị hỗ trợ.
  - Flag có giá trị bằng true nghĩa là chưa thể xác định được tính phổ

biến của item đó và tiếp tục tính toán giá trị hỗ trợ.

Ví dụ minh họa với CSDL như trong Bảng 8, minsup = 50%

Customer ID	Sequence Data
1	<{30} {60}>
2	<{10, 20} {30} {40, 50, 60, 20}>
3	<{30, 50}>
4	<{90} ,{70},{80}>
5	<{90},{70},{80},{30},{20}>

Bảng 8: CSDL tuần tự minh họa

Xét chuỗi tuần tự s = <{90},{70},{80}>. Với item i = {90} ta thấy dù i xuất hiện trong chuỗi cuối cùng thì giá trị hỗ trợ tối đa của item i = {90} cũng chỉ là 2 nhỏ hơn giá trị minsup của bài toán. Như vậy việc duyệt và tính toán giá trị hỗ trợ của item i trong chuỗi tuần tự cuối cùng theo cách thông thường là không cần thiết. Tương tự là với các item {70}, {80}.

Đối với các CSDL nhỏ việc giảm thiểu các phép toán này là không đáng kể nhưng với những CSDL có số chuỗi tuần tự lên tới hàng nghìn, trăm nghìn thì chúng ta có thể giảm thiểu một số lượng lớn các phép toán.

Giả sử ta có CSDL có 1.000.000 bản ghi với giá trị minsup = 40%, nếu có 1 item j chỉ xuất hiện ở bản ghi thứ 600.001 và xuất hiện trong toàn bộ các bản ghi sau đó (từ bản ghi 600.002 đến bản ghi 999.999). Vậy giá trị hỗ trợ tối đa của item j sẽ là 399.999 < minsup. Trong khi thuật toán CMSPAM vẫn duyệt qua và tính toán giá trị hỗ trợ cho 399.998 lần xuất hiện phía sau của item j. Việc này gây lãng phí và làm giảm tốc độ xử lý của thuật toán. Vì thế nhóm chúng tôi tin rằng thực hiện cải tiến như trên sẽ giúp tăng hiệu năng của CMSPAM.

Thuật toán CMSPAME	
Input	Một cơ sở dữ liệu tuần tự S, và giá trị ngưỡng minsup do người sử dụng đặt ra
Output	Tập đầy đủ các mẫu tuần tự F
Method	Chương trình chính: Gọi hàm CMSPAME(minsup , S)
Pamameters: Ý nghĩa các tham số 1. S:Tập dữ liệu 2. Minsup:Giá trị minsup	

```
Method: Nội dung hàm CMSPAME(minsup , S)
// Bước1:Quét Cơ sở dữ liệu SDB để tạo cơ sở dữ liệu theo chiều dọc VDB
sid = tid = 0;
FOR(each item s ∈ SDB){
   IF(s is end of transation){
                        tid++

   }ELSE IF(s is end of sequence){
                        sid++
                        tid = 0

   }ELSE{
       IF (s.flag = false) continue
            bitmapItem = VDB.get[s]
       IF(bitmapItem = NULL){
            VDB.add(s,new bitmap())
       }


       IF(bitmapItem.getSupport +
    (sequencesSize - sid ) < minsup){
              s.flag = false
                        continue
        }

       bitmapItem.registerBit(sid, tid);
             }
       }

// Bước 2 :Quét Cơ sở dữ liệu chiều dọc VDB để loại bỏ những item không phổ biến, tập F chứa danh sách các item phổ biến .
       FOR(each item s ∈ VDB){
              IF(s is frequent) {
                        F = F ∪ s
              }ELSE
                        VDB remove s
       }

// Bước 3 : thực hiện khởi tạo CMAP
       CREATECMAP(SDB,F,minsup)

// Bước 4 : Thực hiện mở rộng và cắt tỉa chuỗi
       FOR(each item s ∈ F)
           DFS-
       Pruning(<s>,CMAP_i[s],CMAP_s[s])
```

Bảng 9: Thuật toán CMSPAME

Thuật toán cải tiến được chúng tôi đặt tên là CMSPAME. Bảng 9 mô tả thuật toán, một số hàm và chương trình con vẫn giữ nguyên như trong CMSPAM đã trình bày trong mục II.

Cải tiến của chúng tôi không thay đổi các tính toán chính của CMSPAM nên vẫn đảm bảo tính chính xác như CMSPAM. Độ phức tạp tính toán của thuật toán cũng không tốt hơn vì trường hợp xấu nhất thuật toán vẫn

phải quét hết các item trong cơ sở dữ liệu SDB. Tuy nhiên, với trường hợp dữ liệu thưa như đã phân tích, bước đánh dấu và ước lượng cận trên sẽ giúp giảm không gian tính toán và tăng hiệu năng của thuật toán. Các thử nghiệm trong phần IV sẽ minh chứng cho nhận định này.

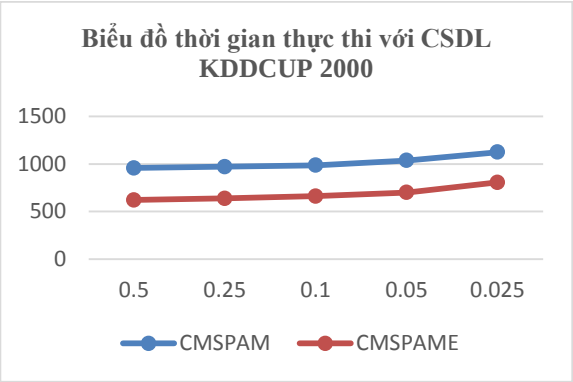
IV. THỬ NGHIỆM VÀ ĐÁNH GIÁ CMSPAME

Để so sánh hai thuật toán CMSPAME và CMSPAM, chúng tôi tiến hành thử nghiệm với 3 bộ dữ liệu tương tự như đã thực hiện trong phần II.

Thử nghiệm 1 với KDDCUP 2000 gồm 77512 khách hàng, 3340 sản phẩm, mỗi khách hàng có trung bình 6.07 sản phẩm giao dịch với 4.62 sản phẩm khác nhau, tổng cộng có **358278** lượt đọc item từ CSDL. Kết quả thử nghiệm 1 cho trong Bảng 10 và Hình 4.

Minsup	0.5	0.25	0.1	0.05	0.025
CMSPAM	958	971	987	1037	1124
CMSPAME	621	638	661	716	806
Kết quả	0	0	0	0	10
Số lượt đọc item bỏ qua	176319	86727	34241	15178	5554
Tỉ lệ lượt đọc item bỏ qua(%)	49.21	24.27	9.55	4.23	1.55
Thời gian chênh lệch	337	333	326	321	318
Tỉ lệ thời gian giảm thiểu(%)	35.17	34.29	33.03	30.95	28.29

Bảng 10: Kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu KDDCUP 2000



Hình 4: Biểu đồ kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu KDDCUP 2000

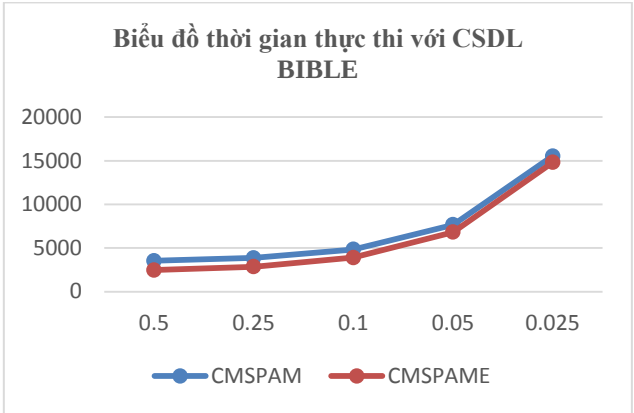
Kết quả thử nghiệm 1 cho thấy với bộ dữ liệu có đặc trưng thưa như KDDCUP2000, thời gian chạy của thuật toán CMSPAME được cải thiện rõ rệt so với CMSPAM, tất cả các giá trị thời gian chạy của CMSPAME đều nhỏ hơn so với CMSPAM.

Thử nghiệm 2 với BIBLE: gồm 36369 khách hàng, 13905 sản phẩm, mỗi khách hàng có trung bình 21.6 sản phẩm giao dịch với 17.84 sản phẩm khác nhau, tổng cộng có **787066** lượt đọc item từ CSDL. Kết quả thử nghiệm 2 cho trong Bảng 11 và Hình 5.

Đối với CSDL trên có đặc trưng là có nhiều sản phẩm mà mỗi chuỗi đều gồm nhiều giao dịch và nhiều sản phẩm. Tốc độ xử lý của thuật toán CMSPAME cải thiện hơn so với thuật toán CMSPAM tuy nhiên chênh lệch là không nhiều. Giá trị minsup càng lớn thì thời gian xử lý chênh lệch giữa 2 thuật toán càng rõ hơn.

Minsup	0.5	0.25	0.1	0.05	0.025
CMSPAM	3531	3858	4831	7666	15529
CMSPAME	2476	2847	3894	6808	14811
Kết quả	5	22	174	774	3285
Số lượt item bỏ qua	278112	112219	33823	12533	4216
Tỉ lệ lượt đọc item bỏ qua(%)	35.33	14.26	4.30	1.59	0.06
Thời gian chênh lệch	1055	1011	937	858	718
Tỉ lệ thời gian giảm thiểu(%)	29.87	26.20	19.39	11.19	4.62

Bảng 11: Kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu BIBLE



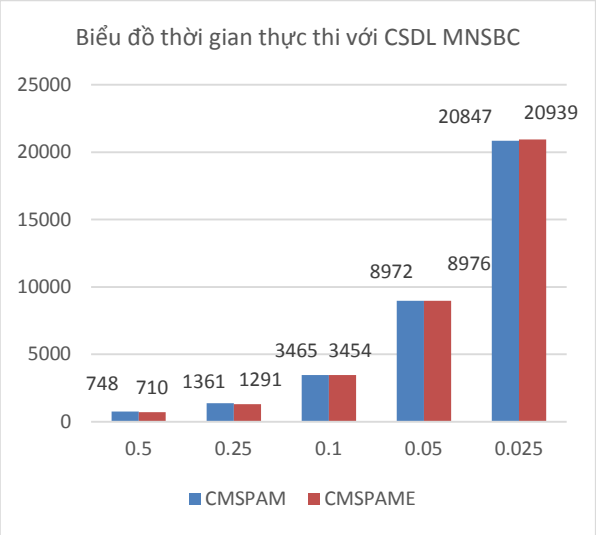
Hình 5: Biểu đồ kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu BIBLE

Thử nghiệm 3 với MSNBC gồm 31790 khách hàng, 17 sản phẩm, mỗi khách hàng có trung bình 13.33 sản phẩm giao dịch với

5.33 sản phẩm khác nhau, tổng cộng có **423776** lượt đọc item từ CSDL.

Minsup	0.5	0.25	0.1	0.05	0.025
CMSPAM	748	1361	3465	8972	20847
CMSPAME	710	1291	3454	8976	20939
Kết quả	5	44	338	1478	6068
Số lượt item bỏ qua	52744	6430	304	44	7
Tỉ lệ lượt đọc item bỏ qua(%)	12.44	1.51	0.07	0.01	0.0017
Thời gian chênh lệch(ms)	38	70	11	-0.04	-92
Tỉ lệ thời gian giảm thiểu(%)	5.08	5.14	0.39	-	-

Bảng 12: Kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu MNSBC



Hình 6: Biểu đồ kết quả kiểm thử thuật toán CMSPAM và CMSPAME với bộ dữ liệu MNSBC

Kết quả trong Bảng 12 và Hình 6 cho thấy với đặc trưng của bộ dữ liệu này là có rất ít sản phẩm (chỉ có 17 giao dịch) khiến cho thuật toán CMSPAME không cải thiện nhiều tốc độ xử lý so với thuật toán CMSPAM. Đặc biệt với 2 giá trị minsup 0.05 và 0.025 thời gian thực thi của CMSPAME còn lớn hơn so với thuật toán CMSPAM. Nguyên nhân là số lượt đọc item bỏ qua là quá nhỏ (44 và 7) không đáng kể so với toàn bộ số lượt đọc item của toàn bộ CSDL.

V. KẾT LUẬN

Bài báo đã tìm hiểu về khai phá dữ dữ liệu tuần tự (SPM) và các thuật toán liên quan. Chúng tôi đã đi sâu tìm hiểu thuật toán khai phá dữ liệu CMSPAM và tiến hành thử nghiệm để chứng tỏ ưu điểm của thuật toán này với các thuật toán trước đó.

Dựa trên việc phân tích của trường hợp CSDL lớn với rất nhiều người dùng nhưng số giao dịch và số sản phẩm giao dịch trong một lần không nhiều (trường hợp dữ liệu thưa), chúng tôi đã đề xuất thuật toán cải tiến CMSPAME. Các thử nghiệm đã cho thấy thuật toán cải tiến có hiệu năng tốt hơn so với CMSPAM trong các bài toán có nhiều sản phẩm, số lượng giao dịch lớn đồng thời yêu cầu giá trị ngưỡng phổ biến (minsup) cao.

Nhóm tác giả sẽ tiếp tục phát triển thuật toán để hướng tới việc áp dụng trong các hệ thống thương mại điện tử có cùng đặc trưng dữ liệu thưa như đã phân tích.

VI. TÀI LIỆU THAM KHẢO

[1] R. Agrawal and R. Srikant. “Mining sequential patterns”. In Proc. 1995 Int. Conf. Data Engineering (ICDE’95), pages 3–14, Taipei, Taiwan, 1995.

[2] Q. Zhao and S. S. Bhowmick. “Sequential Pattern Mining: A Survey”, Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003118, 2003.

[3] J. Han, H. Cheng, D. Xin, X. Yan: “Frequent pattern mining: current status and future directions”. Springer Science+Business Media, LLC, 2007.

[4] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. “Sequential PAttern Mining using A Bitmap Representation”,SIGKDD, pp. 429–435, 2002.

[5] P. Fournier-Viger, A. Gomariz, M. Campos and R. Thomas. “Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information”, 2014.

[6] M. Verma, D. Meht. “Sequential Pattern Mining: A Comparison between GSP, SPADE and Prefix SPAN”, IJEDR, Volume 2, Issue 3, 2014.

[7] M. J. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, In: Machine Learning. Number 1/2 Vol. 42, 2001.

[8] P. Fournier-Viger. “An Open-Source Data Mining Library”, online at:

http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Nhóm tác giả:

ThS. Nguyễn Mạnh Sơn: hiện tại là giảng viên khoa CNTT1 – Học viện công nghệ Bưu chính Viễn thông.
Các hướng nghiên cứu chính: Khai phá dữ liệu từ mạng xã hội, học máy và tư vấn, tối ưu hóa thuật toán.

ThS. Đặng Ngọc Hùng: hiện tại là giảng viên khoa CNTT1 – Học viện Công nghệ Bưu chính Viễn thông.
Các hướng nghiên cứu chính: Khai phá dữ liệu từ mạng xã hội, các hệ thống thông tin.