THUẬT TOÁN KHAI THÁC MẪU PHỔ BIẾN HIỆU QUẢ TRÊN CÂY FP-TREE

Trần Anh Duy[1], Phạm Đức Thành[2]

[1,2] Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM

duy.ta@huflit.edu.vn, phamducthanh@huflit.edu.vn

TÓM TẮT— Khai thác mẫu phổ biến là bài toán quan trọng trong lĩnh vực khai thác dữ liệu. Mẫu phổ biến được ứng dụng trong nhiều lĩnh vực như hệ thống bán hàng, truy tìm mẫu thâm nhập hệ thống, xác định mẫu lặp lại trong phân tích gen di truyền, … Đã có rất nhiều thuật toán được đề xuất để khai thác mẫu phổ biến. Các phương pháp đã đề xuất đa phần sử dụng chiến lượt phát sinh mẫu và kiểm thử. Do đó, xuất hiện rất nhiều mẫu cần phải kiểm tra. Bên cạnh đó, việc phát sinh mẫu sẽ dẫn đến trường hợp các mẫu phát sinh không có trong cơ sở dữ liệu. Bài viết này đề xuất một phương pháp khai thác hiệu quả dựa trên cấu trúc dữ liệu FP-Tree giúp giảm số lượng mẫu phát sinh trong quá trình khai thác. Kết quả thực nghiệm cho thấy hiệu quả của thuật toán tốt hơn so với một số phương pháp hiện có.

Từ khóa— Khai thác mẫu phổ biến, luật kết hợp, dữ liệu giao tác, cấu trúc FP-Tree, khai thác tập sự kiện.

I. GIỚI THIỆU

Khai thác dữ liệu đã và đang thu hút rất nhiều sự chú ý trong cộng đồng nghiên cứu cơ sở dữ liệu, vì khả năng ứng dụng trong nhiều lĩnh vực. Khai thác mẫu phổ biến là một kỹ thuật khai thác dữ liệu, đóng vai trò thiết yếu trong nhiều nhiều lĩnh vực khai thác như khai thác mối liên hệ, khai thác tính tương quan, xác định quan hệ nhân quả, khai thác mẫu tuần tự, khai thác tập phổ biến đa chiều, khai thác tập mẫu tối đại, xác định tính tuần hoàn, v.v…

Khai thác tập phổ biến được Agrawal [1-2]đề xuất lần đầu tiên trong một bài viết về phân tích thị trường ứng dụng kỹ thuật khai thác luật kết hợp. Nhiệm vụ của luật khai thác luật kết hợp là tìm ra các mối liên hệ của các mặt hàng khác nhau được khách hàng chọn mua, thông qua việc xác định sự xuất hiện đồng thời của các mặt hàng đó trong cùng hóa đơn mua hàng tại cửa hàng. Ví dụ như dựa vào việc phân tích hóa đơn mua hàng, người ta phát hiện ra rằng những người mua tã thường mua kèm bia trong cùng hóa đơn. Thông tin về những sản phẩm xuất hiện trong cùng hóa đơn có thể được sử dụng để phân tích thị trường riêng biệt của từng cửa hàng, lên kế hoạch kinh doanh, sắp xếp gian hàng sao cho tiện dụng nhất cho khách hàng, phân nhóm khách hàng, …

Kể từ khi xuất hiện, bài toán khai thác dữ liệu phổ biến đã nhận được sự quan tâm của giới nghiên cứu với hàng trăm công trình khác nhau, từ nghiên cứu thuật toán khai thác [3-10] cho đến các lĩnh vực ứng dụng. Mặc dù có nhiều công trình được đề xuất để giải quyết vấn đề khai thác tập phổ biến, nhưng thiết kế phương pháp khai thác hiệu quả với từng loại dữ liệu vẫn là bài toán còn nhiều thách thức.

Trong bài báo này, chúng tôi đề xuất một phương pháp khai thác mới FP-Extend dựa trên cấu trúc FP-Tree [10] đã được Han đề xuất năm 2004. FP-Extend sử dụng cấu trúc Node-List để lưu trữ vị trí của các mẫu trên cây FP-Tree phục vụ cho quá trình phát sinh mẫu từ cây. Bằng cách duyệt ngược cây kết hợp với việc lưu trữ thông tin mẫu trên Node-List, phương pháp sẽ tìm ra tất cả các tập phổ biến mà không cần tạo ra ứng viên mới.

Thuật toán FP-Extend đạt được hiệu quả so với thuật toán FP-Growth ở các khía cạnh sau. Đầu tiên là thuật toán duyệt cấu trúc FP-Tree để phát sinh mẫu nên không phát sinh mẫu không tồn tại trong CSDL. Thứ hai, trong quá trình khai thác, thuật toán có kiểm tra độ hỗ trợ của các mẫu trước khi thực hiện bước khai thác kế tiếp. Do đó, thuật toán không phát sinh tất cả các mẫu có thể có giống như thuật toán FP-Tree, giúp giảm đáng kể thời gian khai thác. Thứ 3, việc lưu vết mẫu phổ biến trong Node-List cho phép nhanh chóng phát sinh mẫu kế tiếp. Đồng thời độ hỗ trợ của các mẫu được tính thông qua việc cộng dồn thuộc tính đếm của nút trong cây FP-Tree nên việc kiểm tra trở nên đơn giản.

Để đánh giá hiệu suất của FP-Extend chúng tôi tiến hành thực nghiệm so sánh với thuật toán FP-Growth [10] và PrePost [9]. Kết quả thực nghiệm cho thấy thuật toán hiệu quả hơn so với các thuật toán khảo sát về thời gian thực hiện và vùng nhớ sử dụng.

Phần còn lại của bài báo được sắp xếp như sau. Mục II trình bày về các công trình có liên quan. Mục III trình bày về thuật toán khai thác đề xuất. Mục IV trình bày kết quả thực nghiệm. Mục V là phần kết luận và hướng phát triển trong tương lai.

II. CÁC CÔNG TRÌNH LIÊN QUAN

Hầu hết các thuật toán được đề xuất trước đây để khai thác các tập phổ biến có thể được nhóm lại thành hai nhóm: phương pháp dựa trên nguyên lý Apriori [2] và phương pháp cải tiến dựa trên cây FP-Tree [7][10-12]. Nguyên lý Apriori nói rằng nếu bất kỳ itemset k độ dài nào không phổ biến, thì super-itemset độ dài (k + 1) của

nó cũng không phổ biến. Phương pháp Apriori-like sử dụng chiến lược tạo và kiểm tra tập ứng viên để khám phá các tập phổ biến. Nghĩa là, nó tạo ra các tập phổ biến có độ dài ứng viên (k + 1) trong lần kiểm tra thứ (k + 1) bằng cách sử dụng các tập phổ biến có độ dài k được tạo trong lần kiểm tra trước đó và đếm các độ support của các itemset ứng viên này trong cơ sở dữ liệu. Khác với phương pháp dựa trên nguyên lý Apriori, phương pháp cải tiến dựa trên cây FP-Tree khai thác các tập phổ biến mà không cần tạo ứng viên và đã được chứng minh là rất hiệu quả. Phương pháp FP-growth đạt được hiệu quả ấn tượng bằng cách áp dụng cấu trúc dữ liệu mật độ cao được gọi là FP-tree (cây tập phổ biến) để lưu trữ cơ sở dữ liệu và sử dụng cách tiếp cận dựa trên phân vùng, chiến lược chia để trị để khai thác các tập phổ biến.

Phương pháp Apriori-like đạt được hiệu suất tốt bằng cách giảm kích thước của ứng viên. Tuy nhiên, các nghiên cứu trước đây cho thấy rằng phương pháp Apriori-like rất tốn kém khi quét liên tục cơ sở dữ liệu và kiểm tra một tập hợp lớn các ứng cử viên bằng cách đối sánh tập phổ biến. Để giải quyết những vấn đề này, một số thuật toán khai thác dọc đã được đề xuất [7] [10-12]. Không giống như định dạng cơ sở dữ liệu giao dịch dạng ngang truyền thống, mỗi item trong cơ sở dữ liệu dọc được liên kết với Tidset tương ứng của nó, tập hợp tất cả id giao dịch nơi nó xuất hiện. Ưu điểm của định dạng cơ sở dữ liệu dọc là việc đếm số lượng hỗ trợ của các itemset có thể được thu thập thông qua Tid-set, tránh việc quét toàn bộ cơ sở dữ liệu. Tid-set đơn giản hơn nhiều so với bảng băm hoặc cây phức tạp được sử dụng trong thuật toán ngang và cũng hiệu quả hơn trong việc đếm hỗ trợ của các tập phổ biến. Các thuật toán khai thác theo chiều dọc đã được chứng minh là rất hiệu quả và thường tốt hơn các phương pháp khai thác theo chiều ngang.

Từ năm 2012 đến nay, một số công trình đã được công bố chủ yếu dựa trên việc cải tiến cấu trúc cây FP-Tree như PrePost [9] [11], FIN [8], DFIN [7], negFIN [12], … các biến thể này dựa trên cây FP-Tree cải tiến để đưa ra các cấu trúc trung gian nhằm thực hiện thao tác khai thác mà không cần thực hiện trực tiếp trên cây FP-Tree. Kết quả thực nhiệm của các thuật toán này được chứng minh là tốt hơn so với thuật toán FP-Growth [10] ban đầu.

III. THUẬT TOÁN KHAI THÁC MẪU PHỔ BIẾN

A. BÀI TOÁN KHAI THÁC MẪU PHỔ BIẾN

Cho tập $I=\{i_1, i_2, i_3, \dots, i_n\}$ là tập các sự kiện khác nhau. Cơ sở dữ liệu giao dịch $DB=\{t_1,t_2,t_3, \dots,t_m\}$ là tập hợp các giao dịch t_i với $i \in [0,m]$, và $t_i \subset I$. Mẫu $P \subset I$ được gọi là xuất hiện bên trong giao dịch T nếu và chỉ nếu mọi phần tử của P đều thuộc tập T. Ví dụ: $P=\{a,b,c\}$ và $T=\{a,b,c,d\}$ khi đó P được gọi là xuất hiện trong tập T. Độ hỗ trợ của mẫu P là số lần xuất hiện trong cơ sở dữ liệu giao dịch DB, mỗi giao dịch Ti có chứa P được tính là một lần. Độ dài của mẫu P được tính bằng số item có trong P. Một mẫu có độ dài k còn được gọi là tập *k-itemsets*. Độ hỗ trợ của P được ký hiệu là $sup(P)$.Độ hỗ trợ tối tiểu φ là một ngưỡng do người dùng đặt ra.

Bài toán khai thác mẫu phổ biến được phát biểu như sau: Cho cơ sở dữ liệu giao dịch DB và một ngưỡng hỗ trợ tối tiểu φ yêu cầu tìm ra tất cả các mẫu $P \subset I$ sau cho $\sup(P) \geq \varphi$.

B. THUẬT TOÁN ĐỀ XUẤT

1. CẤU TRÚC CÂY FP-TREE

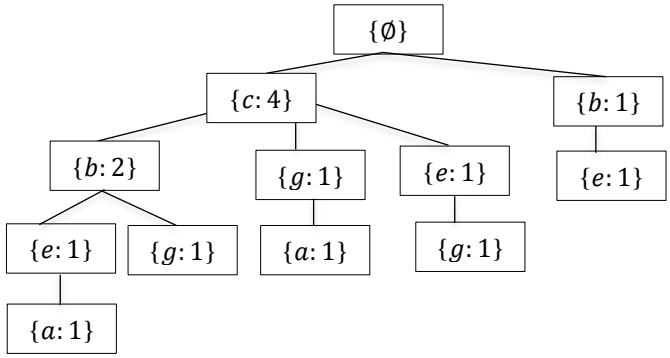
FP-Tree là cấu trúc dữ liệu được J.Han đề xuất năm 2004 [10]. FP-Tree sẽ mô tả lại cơ sở dữ liệu dưới dạng cây mà mỗi nút trong cây sẽ bao gồm các thông tin về sự kiện i và số lần xuất hiện của sự kiện i. Từ nút gốc của cây, đi theo mỗi nhánh, đến nút lá sẽ biểu diễn cho một giao dịch trong cơ sở dữ liệu DB.

Bảng 1: *Bảng ví dụ CSDL*.

Mã Giao Dịch	Tập sự kiện	Tập sự kiện đã sắp xếp	Tập sự kiện đã lọc(minsup =2)
1	a, c, b, e	c, b, e, a	c, b, e, a
2	a, c, g	c, g, a	c, g, a
3	b, c, f, g, h	c, b, g, f, h	c, b, g
4	b, d, e	b, e, d	b, e
5	e, c, g, i	c, e, g, i	c, e, g

Để xây dựng câu trúc FP-Tree, cơ sở dữ liệu gốc phải được sắp xếp theo thứ tự giảm dần của độ hỗ trợ. Trong bài viết này, chúng tôi sử dụng cấu trúc cây nhị phân để thực hiện thao tác tạo ra tập sự kiện đã sắp xếp. Với mỗi giao dịch trong DB, một cây nhị phân được tạo ra để sắp xếp các sự kiện có trong giao dịch đang xét. Mỗi sự kiện của một giao dịch sẽ được thêm vào cây nhị phân. Dựa vào số lần xuất hiện của từng sự kiện để sắp xếp chúng vào bên trái hoặc phải. Nguyên tắc là nút bên trái sẽ có số lần xuất hiện nhỏ hơn nút ở giữa. Nút bên phải sẽ có số lần xuất hiện lớn hơn nút ở giữa. Nếu số lần xuất hiện bằng nhau thì dựa vào thứ tự xuất hiện trước đó trong DB. Ví dụ: Sự kiện e và sự kiện g đều có số lần xuất hiện là 3 tuy nhiên sự kiện e xuất hiện từ giao dịch có mã là 1 còn g xuất hiện ở giao dịch có mã là 2 nên trong giao dịch số 5 sự kiện g sắp sau sự kiện e.

Sau đó, mỗi sự kiện trong từng giao dịch sẽ được duyệt qua để tạo nút trong cây FP-Tree. Trong đó, nếu như sự kiện được duyệt qua nhiều hơn một lần, thì cứ mỗi lần duyệt như vậy sẽ tăng biến đếm trong cây lên. Hình 1 thể hiện cơ sở dữ liệu trên ở dạng FP-Tree.



Ví dụ cây FP-Tree

2. THUẬT TOÁN

Thuật toán khai thác sử dụng cấu trúc bảng băm BT để lưu trữ các mẫu phổ biến. Trong đó, khóa của bảng băm lưu thông tin về mẫu phổ biến. Dữ liệu ánh xạ của bảng băm là tập địa chỉ của các nút tương ứng trong cây FP-Tree và độ hỗ trợ của mẫu. Thuật toán khai thác dựa trên đặc điểm sau của cây FP-Tree.

Nhận xét 1: Từ một mẫu x cho trước, tất cả các mẫu phát sinh từ x đều có thể được phát sinh dựa vào thao tác duyệt ngược cây FP-Tree. Ví dụ: Từ mẫu a trong Hình 1, dựa vào bảng băm BT có thể tìm ra được hai nút có mẫu a. Sau đó, từ một nút có mẫu a, có thể xác định được các mẫu ae, aeb, aebc, … dựa vào thao tác duyệt ngược về gốc cây.

Nhận xét 2: Độ hỗ trợ của mẫu có thể được tính bằng cách cộng dồn thuộc tính đếm bên trong mỗi nút. Ví dụ: Mẫu a có hai nút, vậy độ hỗ trợ của mẫu a có thể đạt được bằng cách cộng dồn thuộc tính cout của mỗi nút.

Nhận xét 3: Nếu như mẫu có độ dài k không phổ biến, thì mẫu phát sinh có độ dài (k+1) từ mẫu có độ dài k không phổ biến cũng không phổ biến.

Từ các nhận xét trên, Thuật toán khai thác được đề xuất như sau.

Thuật toán 1: **EFP-Growth** Khai thác mẫu phổ biến

Input Cơ sở dữ liệu giao dịch ***DB***, Ngưỡng hỗ trợ tối tiểu **φ**
Output Tập các mẫu phổ biến
Begin
Duyệt DB tạo cây FP-Tree và tạo bảng băm BT mẫu 1-item phổ biến.
FOR-EACH mẫu 1-itemsets p **IN** BT
EFP-Mining(p, φ**,** FP-Tree, BT)
End

Thuật toán 2: **EFP-Mining** Khai thác mẫu phổ biến với 1 item cho trước

Input Mẫu k-itemsets p, Ngưỡng hỗ trợ tối tiểu **φ, FP-Tree,** Bảng băm BT
Output Tập các mẫu phổ biến có chứa p
Begin
Tạo Bảng băm tmp.
List = BT[p]
FOR-EACH nút n **IN List**
IF n == root **THEN CONTINUE**
q = nút cha của nút n
DO
l = nhãn sự kiện của nút q
c = số lần xuất hiện trong nút n.
$newpattern = p \cup l$
Cộng c cho $tmp[newpattern].count$
Thêm q vào $tmp[newpattern].list$
Gán nút cha của q cho nút q

```
        WHILE q != root
    FOR-EACH mẫu k IN tmp
        IF tmp[k].count < φ
                Xóa k khỏi tmp
        ELSE
                Xác định k là mẫu phổ biến.
    IF còn mẫu trong tmp
        FOR-EACH mẫu k IN tmp
        EFP-Mining(k, φ, FP-Tree, tmp)
```

End

Thuật toán khởi đầu bằng việc tạo ra cây FP-Tree và bảng băm lưu trữ thông tin mẫu có chiều dài bằng một. Sau đó, thuật toán duyệt qua từng item để phát sinh mẫu liên quan đến item đó. Bằng cách duyệt ngược cây FP-Tree, thuật toán tìm ra tất cả các mẫu có độ dài bằng hai (hay còn gọi là tập 2-itemset) và lưu trong bẳng băm tmp. Sau đó, xóa hết các mẫu không phổ biến trong bảng băm tmp đi. Lặp lại chu trình với bảng băm các mẫu có độ dài hai để phát sinh mẫu có độ dài lớn hơn. Cứ tiếp tục như vậy cho đến khi phát sinh hết tất cả các mẫu.

Do thuật toán có thực hiện kiểm tra độ phổ biến của từng mẫu ở độ dài xác định nên tránh được việc phải phát sinh tất cả các mẫu có thể có. Bên cạnh đó, việc thực hiện duyệt ngược trên cây FP-Tree đảm bảo sẽ không phát sinh những mẫu không có trong cơ sở dữ liệu. Độ hỗ trợ của mẫu cũng được tính toán thông qua quá trình duyệt cây nên dễ dàng xác định được độ hỗ trợ của một mẫu bất kỳ khi kiểm tra tính phổ biến của một mẫu mới được phát sinh. Kết quả thực nghiệm của thuật toán được trình bày bên dưới.

IV. KẾT QUẢ THỰC NGHIỆM

A. CHUẨN BỊ

Thực nghiệm được thực hiện trên máy tính có cấu hình Intel(R) Core (TM) i7-450U CPU @2.00GHz, 2.6GHz, 12GB RAM, chạy trên nền hệ điều hành Windows 10 64-bit. Thuật toán được cài đặt bằng ngôn ngữ lập trình Python trên Jupyter Notebook (Anaconda3 64-bit) với các thư viện có sẵn của Anaconda3. Thuật toán được tìm hiểu và cải tiến trong bài báo là FP-growth ∗ [13].

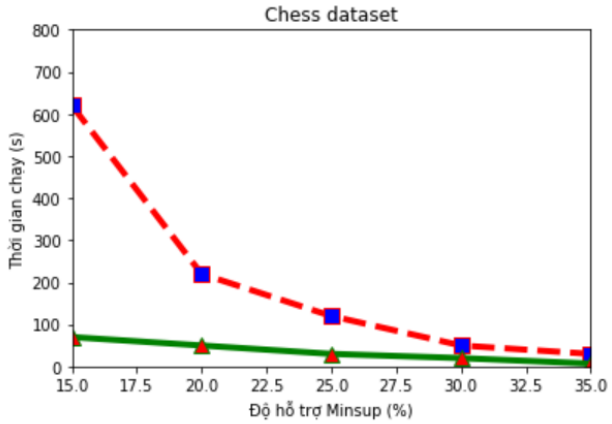
Bài báo sử dụng dữ liệu theo bảng 2 dưới đây:

Bảng 2. *Mô tả các CSDL được sử dụng*

Dataset	Type	*Items*	*Transactions*	*Avg. Length*
chess	Real	75	3,196	37
mushroom	Real	119	8,124	23
T10I4D100K	Synthetic	949	98,487	10

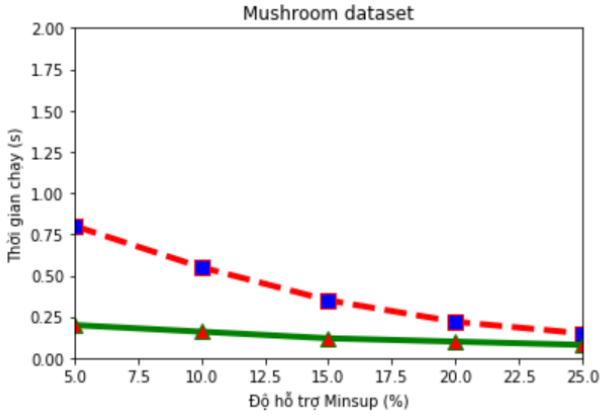
Thuật toán FP-growth ∗ [13] được sử dụng để so sánh với thuật toán cải tiến EFP-Growth mà chúng tôi đề xuất, sử dụng từ thư viện được xây dựng sẵn của Python. Chúng tôi có hai đánh giá sẽ được thực hiện trên CSDL bảng 2, gồm có đánh giá thời gian thực thi chương trình và đánh giá việc hao tốn bộ nhớ.

B. ĐÁNH GIÁ THỜI GIAN THỰC THI CHƯƠNG TRÌNH



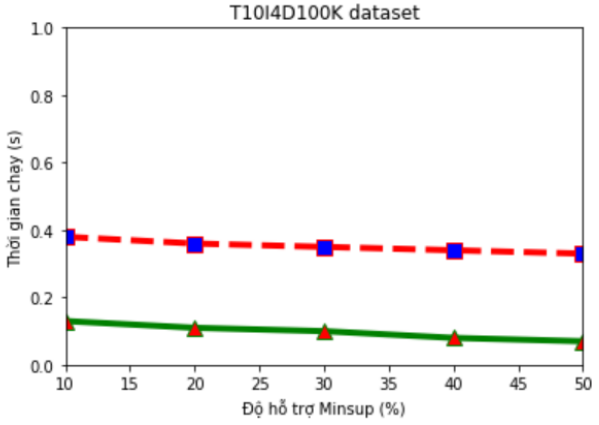
Hình 2. Biểu diễn thời gian thực thi trên Chess dataset

Hình 2 biểu diễn thời gian thực thi của hai thuật toán: FP-growth * (đường gạch đứt, hình vuông) và FP-growth * cải tiến EFP-Growth (đường liên tục, tam giác), trên Chess dataset.



Hình 3. Biểu diễn thời gian thực thi trên Mushroom dataset

Hình 3 biểu diễn thời gian thực thi của hai thuật toán: FP-growth * (đường gạch đứt, hình vuông) và FP-growth * cải tiến EFP-Growth (đường liên tục, tam giác), trên Mushroom dataset.

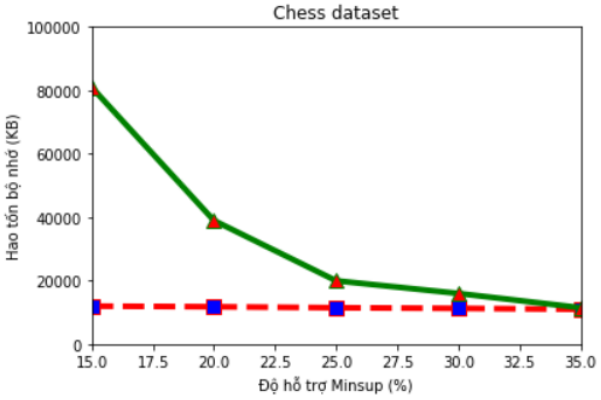


Hình 4. Biểu diễn thời gian thực thi trên T10I4D100K dataset

Hình 4 biểu diễn thời gian thực thi của hai thuật toán: FP-growth * (đường gạch đứt, hình vuông) và FP-growth * cải tiến EFP-Growth (đường liên tục, tam giác), trên T10I4D100K dataset.

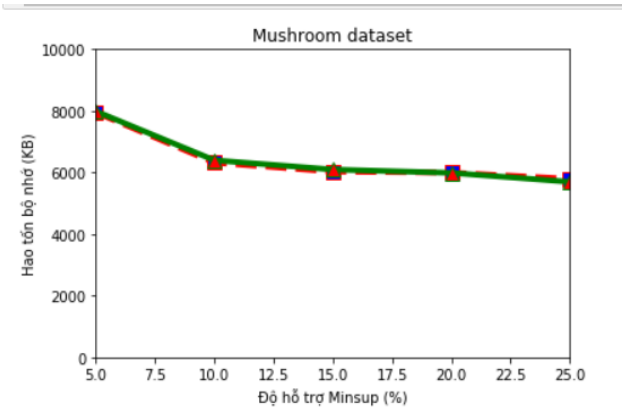
Thực nghiệm cho thấy thời gian chạy của thuật toán FP-growth * cải tiến EFP-Growth của chúng tôi (đường liên tục và tam giác) thực thi nhanh hơn thuật toán gốc FP-growth * (đường gạch đứt, hình vuông).

C. ĐÁNH GIÁ HAO TỐN BỘ NHỚ



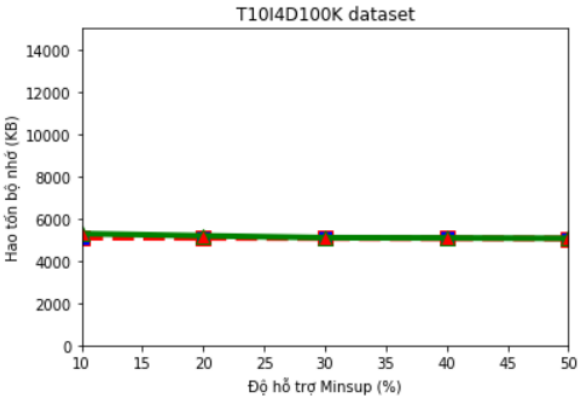
Hình 5. Biểu diễn việc hao tốn bộ nhớ khi thực thi trên Chess dataset

Hình 5 biểu diễn việc hao tốn bộ nhớ khi thực thi của hai thuật toán: FP-growth ∗ (đường gạch đứt, hình vuông) và FP-growth ∗ cải tiến (đường liên tục, tam giác), trên Chess dataset.



Hình 6. Biểu diễn việc hao tốn bộ nhớ khi thực thi trên Mushroom dataset

Hình 6 biểu diễn việc hao tốn bộ nhớ khi thực thi của hai thuật toán: FP-growth ∗ (đường gạch đứt, hình vuông) và FP-growth ∗ cải tiến EFP-Growth (đường liên tục, tam giác), trên Mushroom dataset.



Hình7. Biểu diễn việc hao tốn bộ nhớ khi thực thi trên T10I4D100K dataset

Hình 7 biểu diễn việc hao tốn bộ nhớ khi thực thi của hai thuật toán: FP-growth ∗ (đường gạch đứt, hình vuông) và FP-growth ∗ cải tiến (đường liên tục, tam giác), trên T10I4D100K dataset.

Thực nghiệm cho thấy tốc độ xử lý và việc hao tốn bộ nhớ tùy thuộc vào dữ liệu đầu vào đang xử lý.

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài viết này, chúng tôi trình bày phương pháp sử dụng cấu trúc cây nhị phân để thực hiện thao tác tạo ra tập sự kiện đã sắp xếp. Với mỗi giao dịch trong DB, một cây nhị phân được tạo ra để sắp xếp các sự kiện có trong giao dịch đang xét. Mỗi sự kiện của một giao dịch sẽ được thêm vào cây nhị phân. Dựa vào số lần xuất hiện của từng sự kiện để sắp xếp chúng vào bên trái hoặc phải. Nguyên tắc là nút bên trái sẽ có số lần xuất hiện nhỏ hơn nút ở giữa. Nút bên phải sẽ có số lần xuất hiện lớn hơn nút ở giữa. Nếu số lần xuất hiện bằng nhau thì dựa vào thứ tự xuất hiện trước đó trong DB.

Hạn chế của đề tài là khi xử lý dữ liệu lớn việc phát sinh cây sẽ tiêu tốn thời gian và bộ nhớ. Dữ liệu thực nghiệm có kích thước còn khiêm tốn. Do đó, hướng tới xây dựng chương trình khai thác trên hệ thống phân tán như: Hadoop (map, reduce của Python), Apache Spark (ngôn ngữ Scala). Chúng tôi sẽ thực nghiệm trên nhiều dữ liệu với kích thước lớn hơn. Bên cạnh đó, nhóm sẽ nghiên cứu thêm về hướng cải tiến tương tự đã có hiện nay (sử dụng mảng, danh sách liên kết, nén, …).

VI. TÀI LIỆU THAM KHẢO

[1] I. T. S. A. Agrawal R, "Mining association rules between sets of items in large databases," in *The 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93)* , Washington, 1993.

[2] R. a. R. S. Agarwal, "Fast algorithms for mining association rules," in *The 20th International Conference on Very*, Santiago de Chile, 1994.

[3] M. J. Zaki, "Scalable algorithms for association mining.," *IEEE transactions on knowledge and data engineering*, pp. 372-390, 2000.

[4] M. J. a. K. G. Zaki, "Fast vertical mining using diffsets.," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[5] T. M. K. a. H. A. Uno, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets.," *Fimi(Vol. 126)*, 2004.

[6] J. J. H. H. L. S. N. S. T. a. D. Y. Pei, "H-Mine: Fast and space-preserving frequent pattern mining in large databases.," *IIE transactions*, pp. 593-605, 2007.

[7] Z.-H. Deng, "DiffNodesets: An efficient structure for fast mining frequent itemsets," *Applied Soft Computing*, pp. 214-223, 2016.

[8] Z.-H. a. S.-L. L. Deng, "Fast mining frequent itemsets using Nodesets," *Expert Systems with Applications*, pp. 4505-4512, 2014.

[9] Z. Z. W. a. J. J. Deng, "A new algorithm for fast mining frequent itemsets using N-lists.," *Science China Information Sciences*, pp. 2008-2030, 2012.

[10] J. J. P. Y. Y. a. R. M. Han, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, pp. 53-87, 2004.

[11] Z.-H. a. S.-L. L. Deng, "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children–Parent Equivalence pruning," *Expert Systems with Applications*, pp. 5424-5432, 2015.

[12] N. B. M.-B. a. M. T. Aryabarzan, "negFIN: An efficient algorithm for fast mining frequent itemsets," *Expert Systems with Applications*, pp. 129-143, 2018.

[13] G. a. J. Z. Grahne, "Fast algorithms for frequent itemset mining using fp-trees.," *IEEE transactions on knowledge and data engineering.*, vol. 17, pp. 1347-1362, 2005.

AN EFFECTITIVE ALGORITHM FOR FREQUENT ITEMSETS MINING BASED ON FP-TREE

Tran Anh Duy, Pham Duc Thanh

ABSTRACT— In Frequent itemset mining is an important problem in the field of data mining. Frequent itemsets are used in many fields such as sales systems, tracking system penetration patterns, identifying repeat patterns in genetic analysis, ... Many algorithms have been proposed. Most of the proposed methods use sample generation and testing strategies. There are a lot of samples that need testing. In addition, the sample generation will lead to the case that the generated samples are not in the database. This article proposes an efficient mining method based on FP-Tree data structure that helps to reduce the number of samples generated during the mining process. Experimental results show that the efficiency of the algorithm is better than that of some existing methods.

Họ tên **: Trần Anh Duy**
Nhận học vị thạc sĩ Khoa học máy tính trường Đại học Khoa Học Tự Nhiên năm 2017. Hiện là giảng viên khoa Công Nghệ Thông Tin trường Đại Học Ngoại Ngữ Tin Học thành phố Hồ Chí Minh. Lĩnh vực nghiên cứu đang quan tâm là : Khai thác dữ liệu.

Họ tên: **Phạm Đức Thành**.
Nhận học vị Thạc sĩ năm 2006 tại Đại học Quốc gia Thành phố Hồ Chí Minh; hiện đang là Giảng viên công tác tại khoa Công nghệ Thông tin trường Đại học Ngoại ngữ Tin học Tp. Hồ Chí Minh; lĩnh vực nghiên cứu đang quan tâm là: khai thác dữ liệu.