ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN KHAI PHÁ LUẬT KẾT HỢP TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG

EVALUATING THE EFFECTIVENESS OF ASSOCIATION RULES MINING ALGORITHMS IN PARALLEL PROCESSING ENVIRONMENT

Nguyễn Đăng Cẩm, Nguyễn Thành Sơn
Trường Đại học Sư phạm Kỹ thuật TP.HCM, Việt Nam

Ngày toà soạn nhận bài 30/11/2018, ngày phản biện đánh giá 14/2/2019, ngày chấp nhận đăng 2/4/2019.

TÓM TẮT

Luật kết hợp chỉ ra mối quan hệ, sự kết hợp hay mối tương quan giữa các đối tượng trong cơ sở dữ liệu. Khai phá luật kết hợp là bài toán đã và đang được quan tâm nghiên cứu trong lĩnh vực khai phá dữ liệu. Các thuật toán thường được sử dụng trong khai phá luật kết hợp là Apriori, FP-Growth. Mục đích của bài báo là đánh giá hiệu quả của các thuật toán Apriori, FP-Growth và Apriori cải tiến trong môi trường xử lý song song. Việc so sánh các thuật toán dựa vào hai yếu tố thời gian thực thi và hiệu suất của thuật toán sử dụng. Kết quả thực nghiệm trên các bộ dữ liệu thực cho thấy rằng trong môi trường xử lý song song, thuật toán PF-Growth thực thì hiệu quả nhất.

Từ khóa: *Khai phá dữ liệu; Khai phá luật kết hợp; Apriori; FP-Growth; Apriory cải tiến.*

ABSTRACT

An association rule indicates the relationship, association, or correlation between objects in the database. Association Rule Mining is a problem which has received an increasing amount of attention lately in data mining. Appriori and FP-Growth have commonly used algorithms in association rule mining. The aim of the paper is to evaluate the effectiveness of the Apriori, FP_Growth and Improved Apriori in a parallel processing environment. The comparison is based on execution time and the performance. The experimental results showed that in the parallel processing environment the FP-growth algorithm is the most efficient one.

Keywords: *Data Mining; Association Rule Mining; Apriori; FP-Growth; Improved Apriory.*

1. GIỚI THIỆU

Khai phá dữ liệu là một quá trình đầy hứa hẹn và phát triển trong phân tích dữ liệu và nó được ứng dụng trong nhiều lĩnh vực. Khai phá dữ liệu là cốt lõi của quá trình "Phát hiện tri thức từ cơ sở dữ liệu" (Knowledge Discovery in Database-KDD), là quá trình khai phá, trích xuất, khai thác và sử dụng những dữ liệu có giá trị tiềm ẩn từ bên trong lượng lớn dữ liệu được lưu trữ trong các cơ sở dữ liệu (CSDL), kho dữ liệu, trung tâm dữ liệu lớn.

Các thuật toán khai phá luật kết hợp tuần tự khi sử dụng với dữ liệu lớn sẽ mất nhiều thời gian. Vì vậy, yêu cầu cần có những thuật toán song song hiệu quả cho việc phát hiện các luật kết hợp trong khai phá dữ liệu là rất cần thiết.

Hai hướng tiếp cận chính trong thiết kế thuật toán khai phá luật kết hợp song song là mô hình song song dữ liệu, mô hình song song thao tác.

Bài báo này nhằm mục đích đánh giá hiệu quả giữa các thuật toán Apriori, Apriori cải tiến, Apriori cải tiến và FP-Growth trong môi trường xử lý song song.

Phần còn lại của bài báo bao gồm: Phần 2 trình bày các khái niệm cơ bản và các công trình liên quan. Phần 3 giới thiệu về các giải thuật khai phá luật kết hợp. Cách tiếp cận khai phá luật kết hợp sử dụng giải thuật song song được trình bày trong phần 4. Phần 5 là

kết quả thực nghiệm. Phần 6 là kết luận và hướng phát triển của đề tài.

2. CÁC KHÁI NIỆM CƠ BẢN VÀ CÁC CÔNG TRÌNH LIÊN QUAN

2.1. Các khái niệm cơ bản

Cho $D = \{t_1, t_2, \ldots, t_m\}$ là cơ sở dữ liệu các giao dịch. Mỗi giao dịch t_i bao gồm một tập n thuộc tính $I = \{i_1, i_2, \ldots, i_n\}$ và có một định danh duy nhất TID.

- Một luật định nghĩa sự kéo theo có dạng X ⇒ Y trong đó X,Y ⊆ I và X ∩ Y = Ø. Trong đó, X gọi là phần mệnh đề điều kiện và Y gọi là mệnh đề kết quả của luật tương ứng.

- Độ phổ biến

$$\text{Supp(X)} = |\text{X}| / |\text{D}| \quad (1)$$

$$Supp(X \Rightarrow Y) = \frac{\left|\{T \subseteq D : X \cup Y \subseteq T\}\right|}{|D|} \quad (2)$$

- Độ tin cậy

$$\text{Conf}(\text{X} \Rightarrow \text{Y}) = \frac{\text{Supp}(\text{X} \Rightarrow \text{Y})}{\text{Supp}(\text{X})} \quad (3)$$

2.2. Các công trình liên quan

Một trong những thuật toán đầu tiên khai phá luật kết hợp, thuật toán Apriory, do RaKesh Agrawal và các cộng sự đề suất năm 1993 [1]. Thuật toán sau đó trở thành nền tảng cho sự phát triển các thuật toán sau này.

Năm 2000, Jiawei Hai và các cộng sự đề xuất thuật toán FP-Growth. Thuật toán này sử dụng FP-tree để tìm các tập mục phổ biến, nhằm giảm số lần quét qua cơ sở dữ liệu [2].

Trong [3], Z. H. Deng và các cộng sự đề xuất một kiểu dữ liệu mới theo chiều dọc gọi là N-list, bắt nguồn từ FP-tree-like gọi là PPC-Tree. Dựa trên cấu trúc dữ liệu N-list, phát triển một thuật toán khai thác hiệu quả là PrePost, để khai thác tập mục phổ biến. Các quả thực nghiệm cho thấy thuật toán PrePost hiệu quả nhanh nhất trong tất cả các trường hợp. Mặc dù thuật toán sẽ tốn nhiều bộ nhớ khi các bộ dữ liệu nhỏ.

Trong [4], Aiman Moyaid Said và các cộng sự đã cài đặt và so sánh thuật toán cải tiến của Fpgrowth, AFOPT, Nonordfp, Fpgrowth với dữ liệu T10I4D100k, T40I10D100K, Mushroom, Connect4. Với thuật toán Fpgrowth chạy trên dữ liệu T10I4D100k, T40I10D100K, Mushroom, Connect 4. Kết luận hiệu suất của thuật toán APFOPT là ổn định nhất trên hầu hết các loại dữ liệu.

Trong [5], Zhi-Hong Deng và cộng sự trình bày một thuật toán hiệu quả được gọi là FIN để khai thác các tập mục thường xuyên. Để đánh giá hiệu suất của FIN, họ đã tiến hành các thực nghiệm để so sánh nó với PrePost và FP-growth trên nhiều bộ dữ liệu thực và tổng hợp. Kết quả thử nghiệm cho thấy rằng FIN có hiệu suất cao trên cả thời gian chạy và mức sử dụng bộ nhớ.

Trong [6], Dawen Xia, Yanhui Zhou, Zhuobo Rong, và Zili Zhang đề xuất một thuật toán FP-Growth sử dụng giải thuật song song cải tiến (IPFP), sử dụng MapReduce để thực hiện thuật toán FP-Growth sử dụng giải thuật song song. Do đó cải thiện hiệu suất tổng thể và hiệu quả khai phá các tập mục phổ biến.

Một số giải thuật khai phá luật kết hợp sử dụng trong môi trường xử lý song song cũng đã được đề xuất nhằm tăng tốc độ xử lý của thuật toán. Chẳng hạn, trong [7] Yanbin Ye and Chia-Chu Chiang giới thiệu giải thuật song song để khai phá các tập mục phổ biến sử dụng thuật toán Apriory; trong [8], Yi Wang và các công sự sử dụng giải thuật FP-Growth trong môi trường song song để khai phá các tập mục phổ biến. Cách tiếp cận chung của các giải thuật khai phá luật kết hợp thường được thực hiện qua hai giai đoạn:

(1) Tìm tất cả các tập mục dữ liệu có độ hỗ trợ thỏa ngưỡng tối thiểu cho trước, gọi là các tập mục dữ liệu phổ biến.

(2) Tìm ra những luật kết hợp từ những tập mục dữ liệu phổ biến thỏa độ tin cậy cho trước.

Các công trình nghiên cứu về bài toán khai phá luật kết hợp thường tập trung đề xuất mới hoặc cải tiến thuật toán thực hiện trong giai đoạn 1 tìm tất cả các tập mục phổ biến. Còn giai đoạn 2 thường là xử lý như nhau.

3. MỘT SỐ GIẢI THUẬT KHAI PHÁ LUẬT KẾT HỢP

3.1. Giải thuật Apriori

Thuật toán sinh tập mục ứng viên từ những tập mục phổ biến ở bước trước, sử dụng kỹ thuật "tỉa" để bỏ đi tập mục ứng viên không thỏa mãn ngưỡng hỗ trợ cho trước.

Thuật toán gồm các bước sau:

- Tìm tất cả các tập mục phổ biến 1- phần tử (C_1).
- Tạo các tập ứng viên có k – phần tử (k - candidate itemset) từ các tập phổ biến có (k-1) – phần tử. Ví dụ, tạo tập ứng viên C_2 từ tập phổ biến C_1.
- Kiểm tra độ phổ biến của các ứng viên trên CSDL và loại các ứng viên không phổ biến ta được các tập mục phổ biến L_i, với mọi $1 \le i \le k$.
- Dừng khi không tạo được tập mục phổ biến hay tập ứng viên, i.e., $L_k = \{\}$ hay $C_k = \{\}$.

Mã giả thuật toán Apriori

Dữ liệu vào: Tập các giao dịch D, ngưỡng hỗ trợ minsup

Dữ liệu ra: Tập trả lời bao gồm các tập mục phổ biến trên D

Giải thuật:

L_1 = {large 1-itemset};
for (k = 2; $L_{k-1} \neq \phi$; k++) **do begin**
  C_k = apriori_gen(L_{k-1}); // sinh tập mục ứng viên mới C_k;
  For all giao dịch $t \in D$ **do begin**
    C_t = subset(C_k, t); // các tập mục ứng viên chứa trong t;
    For all tập mục ứng viên $c_i \in C_t$ **do**
      c_i.count ++ ;
    end;
    $L_k = \{c_i \in C_k \mid c_i.\text{count} \ge minsup\}$
  end;

Return tất cả các tập mục phổ biến L_k;

Ưu điểm thuật toán Apriori

- Là thuật toán đơn giản, dễ hiểu và dễ cài đặt.
- Thuật toán Apriori tìm tập mục phổ biến thực hiện tốt bởi rút gọn kích thước các tập ứng viên nhờ kỹ thuật "tỉa".

Nhược điểm thuật toán Apriori

- Phải duyệt CSDL nhiều lần.
- Số lượng lớn tập ứng viên được tạo ra làm gia tăng sự phức tạp không gian.
- Để xác định độ support của các tập ứng viên, thuật toán luôn phải quyét lại toàn bộ CSDL.

3.2. Thuật toán Apriori cải tiến

Để nâng cao hiệu quả khai phá các itemset phổ biến, Girja Shankar và Latita Bargadiya [9] đã thảo luận về hai vấn đề của thuật toán Apriori.

Đầu tiên, thuật toán cần phải quét cơ sở dữ liệu nhiều lần và ở lần thứ hai, nó sẽ tạo ra itemset ứng viên lớn và làm tăng thời gian xử lý, cũng như độ phức tạp về không gian. Để khắc phục những khuyết điểm trên, các tác giả đề xuất cải tiến như sau: đầu tiên tìm frequent_one_itemset của cơ sở dữ liệu sau đó tạo ra tập power của frequent_one_itemset và khởi tạo itemset count = 0. Gọi power set này là Global power set. Khi thuật toán quét qua cơ sở dữ liệu để đếm itemset, thì xóa các item từ giao dịch không có mặt trong danh sách frequent_one_itemset. Sau khi quá trình xóa thuật toán tạo ra Local Power set từ các item còn lại của giao dịch và so sánh với các Global power set. Khi phù hợp thì tăng số lượng itemset lên một. Bước này sẽ làm giảm số lần quét qua cơ sở dữ liệu.

Nội dung thuật toán:

Input:

1) Cơ sở dữ liệu D với định dạng (TID, itemset), trong đó *TID* là một định danh của giao dịch và itemset là một tập mục tương ứng

2) Ngưỡng hỗ trợ tối thiểu: min-sup.

Output: các tập mục phổ biến trong D.

Các bước xử lý:

1) Tìm tập mục phổ biến 1 phần từ
   $L1$ = frequent_one_itemset (D)
2) Tạo power set của $L1$ và khởi tạo itemset count = 0, và gọi nó là Global power set;

3) Quét cơ sở dữ liệu *D* đến hết.

a) Đọc itemset từ giao dịch và xóa các item không ở trong *L*1 và sau đó tạo ra local power set từ các item còn lại của giao dịch.

b) So sánh local power set với Global power set từng cái một và nếu itemset phù hợp thì tăng số lượng itemset lên 1 trong Global power set.

Tỉa các ứng cử itemset.

4) Quét Global power set và đếm số ứng viên itemset;

Nếu độ hỗ trợ của ứng viên itemset nhỏ hơn minsup thì xóa item set đó từ Global power set.

5) Các itemset còn lại của Global power set sẽ là itemset phổ biến được yêu cầu.

3.3. Thuật toán FP-Growth

Thuật toán FP-Growth được đề xuất nhằm khắc phục được các nhược điểm của thuật toán Apriori.

Nội dung thuật toán:

Bước 1: Xây dựng FP-Tree:

- Duyệt CSDL lần một, xác định các tập mục phổ biến *L* và sắp xếp chúng theo độ hỗ trợ.
- Duyệt qua CSDL lần hai, với mỗi giao dịch *T* sắp xếp các tập mục theo thứ tự tập *L*. Giả sử các tập mục phổ biến trong *T* có dạng [p|P] với p là tập mục cần đưa vào FP-Tree và P là danh sách các tập mục còn lại, *N* là nút cần chèn. Nếu nút con của *N* giống *p*, tăng biến *count* nút con đó lên 1. Ngược lại, tạo nút con mới cho *N* có tên mục là *p* và count = 1. Tiếp tục chèn *P* vào nút con vừa xét.

Bước 2: Xây dựng cơ sở mẫu điều kiện (Conditional Patern Bases) cho mỗi tập mục phổ biến.

Bước 3: Xây dựng FP-Tree điều kiện (Conditional FP-Tree) cho mỗi tập mục phổ biến trong cơ sở mẫu điều kiện.

Bước 4: Đệ quy xây dựng FP-Tree điều kiện đến khi FP-Tree điều kiện còn một nhánh duy nhất (single path), sau đó tiến hành sinh tất cả tổ hợp mục phổ biến.

4. KHAI PHÁ LUẬT KẾT HỢP TRONG MÔI TRƯỜNG XỬ LÝ SONG SONG

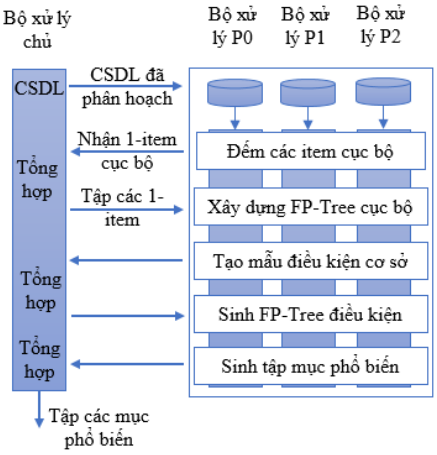
4.1. Khai phá luật kết hợp sử dụng giải thuật song song

Khai phá luật kết hợp sử dụng giải thuật song song dựa trên ý tưởng của khai phá luật kết hợp, thực hiện song song hóa nhằm đáp ứng sự tăng lên nhanh chóng của dữ liệu và giảm thời gian thực hiện.

Các giải thuật xử lý song song được áp dụng trong giai đoạn tìm các tập mục phổ biến nhằm giảm thời gian thực thi của giai đoạn này. Trong các thuật toán dùng trong khai phá luật kết hợp, thuật toán FP-Growth thường được sử dụng trong các giải thuật xử lý song song vì tính hiệu quả của nó.

Khai phá luật kết hợp trong môi trường xử lý song song được thực hiện qua các bước sau: (1) Cơ sở dữ liệu ban đầu được phân hoạch cho các bộ xử lý; (2) Mỗi bộ xử lý thực hiện thuật toán FP-Growth để phát sinh tập mục phổ biến cục bộ; (3) Bộ xử lý chủ tổng hợp các tập mục phổ biến cục bộ từ các bộ xử lý khác để phát sinh các tập mục phổ biến toàn cục; (4) Các luật kết hợp được phát sinh từ tập mục phổ biến toàn cục.

Hình 1 minh họa các bước thực hiện của thuật toán khai phá luật kết hợp FP-Growth trong môi trường xử lý song song.



Hình 1. *Mô hình giải thuật song song dùng thuật toán FP-Growth*

4.2. Thuật toán song song sử dụng Apriori cải tiến

Dựa vào thuật toán Apriori cải tiến, thuật toán này sử dụng mô hình "Chủ-Tớ".

Nội dung thuật toán:

Input:

1) Cơ sở dữ liệu *D* với định dạng (*TID*, itemset), trong đó *TID* là một định danh của giao dịch và itemset là một tập mục tương ứng với công việc kinh doanh. D_1, D_2,…, D_p: các phân hoạch CSDL, *p* là số bộ xử lý.

2) Ngưỡng hỗ trợ tối thiểu: min-sup.

Output: Các tập mục phổ biến trong *D*

Các bước xử lý:

1) Với mọi bộ xử lý *i*,
*L*1(*i*) = tìm frequent_one_itemset (*Di*);

2) Nhận L1(*i*) từ bộ xử lý *i*

Tạo power set của *L*1 và khởi tạo itemset count = 0, và gọi nó là Global power set;

3) Bộ xử lý chủ quét cơ sở dữ liệu *D* đến hết.

Đọc itemset từ giao dịch và xóa các item không ở trong *L*1 và sau đó tạo ra local power set từ các item còn lại của giao dịch.

Gửi local power set và Global power set (*i*) tới các bộ xử lý tớ.

4) Bộ xử lý tớ *i* so sánh local power set với Global power set (*i*). Nếu itemset phù hợp thì tăng số lượng itemset lên 1 trong Global power set (*i*).

5) Bộ xử lý chủ nhận Global power set (*i*) từ bộ xử lý tớ. Quét Global power set và đếm số ứng viên của từng itemset;

Nếu số ứng viên của itemset ít hơn minsup thì xóa itemset đó khỏi Global power set.

6) Các itemset còn lại của Global power set sẽ là itemset phổ biến được yêu cầu.

4.3. Thuật toán song song sử dụng FP-Growth

Thuật toán xây dựng một số Fp-tree cục bộ trong môi trường bộ nhớ phân tán dựa trên mô hình "Chủ - Tớ".

Thuật toán khai phá luật kết hợp này gồm hai nhiệm vụ chính:

- *Xây dựng song song FP-Tree*
- *Khai phá song song và sinh tập mục phổ biến.*

(1) Xây dựng song song FP-Tree

- Chia CSDL giao dịch *D* cho *P* bộ xử lý.
- Mỗi bộ xử lý tính toán độ hỗ trợ (flocal(*i*)) của mỗi mục *i* bằng cách quét phân hoạch CSDL cục bộ. Sau đó, tất cả các bộ xử lý gửi flocal(*i*) cục bộ đến bộ xử lý chủ.
- Bộ xử lý chủ kết hợp các flocal(*i*) lại để sinh độ hỗ trợ toàn cục (fglocal (*i*)).
- Tập các 1-itemset phổ biến thu được sẽ được truyền cho tất cả các bộ xử lý trong nhóm.
- Xây dựng các FP-Tree cục bộ, Mỗi bộ xử lý quét CSDL cục bộ của nó và chèn các mục phổ biến vào trong cây FP-Tree

(2) Khai phá song song và sinh tập mục phổ biến

- Đầu tiên, thuật toán duyệt toàn bộ FP-Tree và tạo ra các mẫu điều kiện cơ sở.
- Tiếp theo, thuật toán tập hợp các mẫu điều kiện cơ sở từ các bộ xử lý để xây dựng FP-Tree điều kiện cơ sở (CFPT) cho mỗi mục phổ biến.
- Cuối cùng là thực thi việc khai phá bằng cách xây dựng đệ qui các mẫu điều kiện cơ sở và các CFPTs cho đến khi nó sinh tất cả các tập mục phổ biến.

5. KẾT QUẢ THỰC NGHIỆM

5.1. Môi trường thực nghiệm

Hệ thống thực nghiệm được cài đặt bằng C# trên nền Microsoft Windows 10 Enterprise 64bit, .Net Framework 4.5, thực hiện trên CPU Intel® Core™ i5-3210M CPU @ 2.50GHz 2.50GHz, RAM 12.0GB, SSD 240GB. Hệ thống phần mềm được sử dụng: Visual Studio 2017 Enterprise, Microsoft's Message Passing Interface (MS-MPI) [10].

5.2. Các tập dữ liệu thực nghiệm

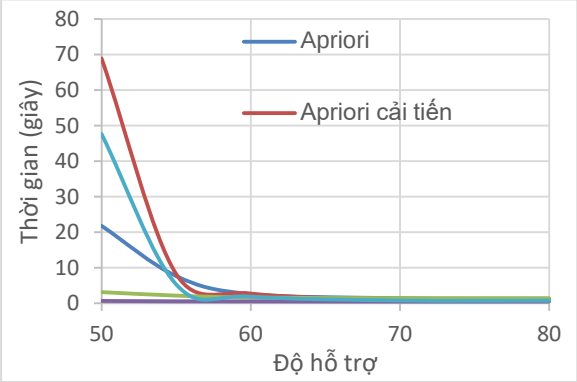
Gồm 3 tập dữ liệu: Dữ liệu mushroom.dat được lấy từ bộ dữ liệu của UCI, có 8124 giao dịch; dữ liệu T10I4D100K được tạo ra bằng cách sử dụng trình tạo từ nhóm nghiên cứu của IBM Almaden Quest, có 100000 giao dịch; dữ liệu BMS_WebView_1 chứa 59.602 giao dịch dữ liệu nhấp chuột từ một trang web thương mại điện tử.

5.3. Kết quả thực nghiệm

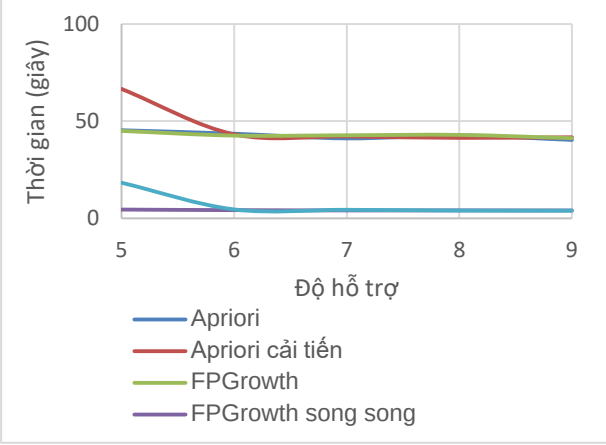
Chúng tôi cài đặt các thuật toán Apriori, Apriori cải tiến, Apriori cải tiến sử dụng giải thuật song song, FP-Growth và FP-Growth sử dụng giải thuật song song, so sánh các thuật toán dựa vào thời gian thực thi và số lượng tài nguyên mà thuật toán sử dụng.

5.3.1. Thời gian thực thi

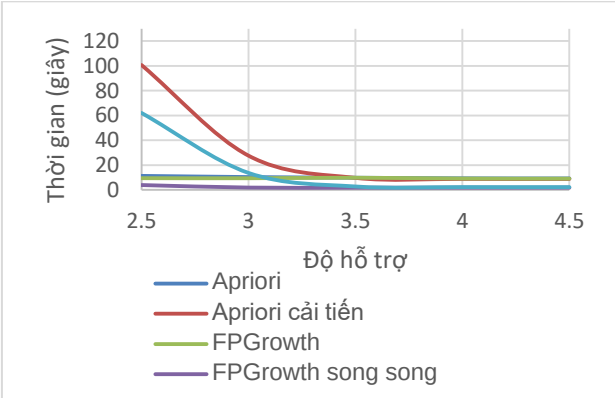
Hình 2, 3, 4 Mô tả kết quả thực nghiệm về thời gian thực thi của các thuật toán được tính bằng giây.



***Hình 2**. Thời gian thực thi của các thuật toán với bộ dữ liệu mushroom.*



***Hình 3**. Thời gian thực thi của các thuật toán với bộ dữ liệu T10I4D100K.*



***Hình 4**. Thời gian thực thi của các thuật toán với bộ dữ liệu BMS_WebView_1.*

Kết quả thực nghiệm cho thấy thời gian thực thi của thuật toán FP-Growth sử dụng giải thuật song song là nhanh nhất, thời gian thực thi của thuật toán Apriori cải tiến và Apriori cải tiến sử dụng giải thuật song song thời tăng đột biến khi độ hỗ trợ nhỏ.

5.3.2. Tài nguyên thuật toán sử dụng

Bảng 1, 2, 3 trình bày kết quả thực nghiệm về hiệu suất (số lượng tài nguyên sử dụng) của các thuật toán.

***Bảng 1**. Kết quả về số lượng tài nguyên (mà thuật toán cần sử dụng) với độ hỗ trợ được chọn là 60% với bộ dữ liệu mushrom.*

Độ hỗ trợ (%)	60	
Tài nguyên	CPU (%)	RAM (MB)
Apriori	24.94	68.15
Apriori cải tiến	24.86	67.41
FPGrowth	24.97	71.54
FPGrowth song song	44.21	132.75
Apriori cải tiến song song	51.72	148.08

***Bảng 2**. Kết quả về số lượng tài nguyên (mà thuật toán cần sử dụng) với độ hỗ trợ được chọn là 7% với bộ dữ liệu T10I4D100K*

Độ hỗ trợ (%)	7	
Tài nguyên	CPU (%)	RAM (MB)
Apriori	24.96	87.42
Apriori cải tiến	24.93	86.09
FPGrowth	24.97	89.65
FPGrowth song song	50.32	283.68
Apriori cải tiến song song	51.40	286.83

***Bảng 3**. Kết quả về số lượng tài nguyên (mà thuật toán cần sử dụng) với độ hỗ trợ được chọn là 3.5% với bộ dữ liệu BMS_WebView_1*

Độ hỗ trợ (%)	3.5	
Tài nguyên	CPU (%)	RAM (MB)
Apriori	24.90	79.61
Apriori cải tiến	24.95	80.09
FPGrowth	24.97	81.42
FPGrowth song song	42.79	199.75
Apriori cải tiến song song	46.73	196.61

Kết quả thực nghiệm cho thấy số lượng tài nguyên sử dụng của các thuật toán trong môi trường xử lý song song gần gấp hai lần thuật toán tuần tự cả về bộ nhớ RAM (Mb) và phần trăm % CPU.

5.3.3. Độ chính xác của thuật toán.

Vì không biết trước trong các tập dữ liệu thực nghiệm có chính xác bao nhiêu tập mục phổ biến, nên để đánh giá về độ chính xác của các thuật toán chúng tôi liệt kê danh sách các tập mục phổ biến được trả về từ các thuật toán đối với mỗi tập dữ liệu và so sánh chúng với nhau. Nếu chúng giống nhau thì độ chính xác là như nhau. Ngược lại là không chính xác. Do giới hạn số trang của bài báo, trong Bảng 4 chúng tôi chỉ trình bày kết quả các tập mục phổ biến tìm được của các thuật toán trên bộ dữ liệu mushroom với độ hỗ trợ 80%. Kết quả cho thấy các thuật toán đều sinh 23 tập mục phổ biến giống nhau. Thực nghiệm trên các bộ dữ liệu còn lại cũng cho kết quả tương tự. Như vậy có thể kết luận độ chính xác của các thuật toán là như nhau.

5.4. Nhận xét kết quả thực nghiệm

Kết quả thực nghiệm cho thấy thời gian thực thi của thuật toán FP-Growth, FP-Growth song song nhỏ hơn thuật toán Apriori tuần tự, thuật toán Apriori cải tiến, Apriori cải tiến song song trên các bộ dữ liệu. Ta thấy FP-Growth song song thời gian thực hiện nhanh hơn FP-Growth và ổn định trên các bộ dữ liệu.

Thuật toán Apriori cải tiến thời gian thực thi nhanh hơn thuật toán Apriori tuần tự với độ hỗ trợ lớn, nhưng với độ hỗ trợ nhỏ sẽ phát sinh số lượng tập các 1-item lớn, nên thuật toán Apriori sinh ra rất nhiều tập mục ứng viên, do đó thời gian thực thi của Apriori cải tiến lớn hơn rất nhiều so với Apriori tuần tự.

Tuy nhiên về mặt tài nguyên, các thuật toán song song cần sử dụng số lượng tài nguyên lớn hơn khoảng 2 lần so với số lượng tài nguyên mà thuật toán tuần tự cần dùng. So sánh các giải thuật song song thì FP-Growth song song sử dụng ít hơn thuật toán Apriori cải tiến song song.

***Bảng 4.** Kết quả các tập mục phổ biến thu được khi chạy các thuật toán trên bộ dữ liệu mushroom*

Supp (%)	Apriori	Apriori cải tiến	FPGrowth	FPGrowth song song	Apriori cải tiến song song
80	{85} (s:100%) {86} (s:97.54%) {85, 86} (s:97.54%) {34} (s:97.42%) {85, 34} (s:97.42%) {86, 34} (s:97.32%) {86, 85, 34} (s:97.32%) {90} (s:92.17%) {85, 90} (s:92.17%)	{85} (s:100%) {86} (s:97.54%) {85, 86} (s:97.54%) {34} (s:97.42%) {85, 34} (s:97.42%) {86, 34} (s:97.32%) {86, 85, 34} (s:97.32%) {90} (s:92.17%) {85, 90} (s:92.17%)	{85} (s:100%) {86} (s:97.54%) {85, 86} (s:97.54%) {34} (s:97.42%) {85, 34} (s:97.42%) {86, 34} (s:97.32%) {86, 85, 34} (s:97.32%) {90} (s:92.17%) {85, 90} (s:92.17%)	{85} (s:100%) {86} (s:97.54%) {85, 86} (s:97.54%) {34} (s:97.42%) {34, 85} (s:97.42%) {34, 86} (s:97.32%) {34, 85, 86} (s:97.32%) {90} (s:92.17%) {85, 90} (s:92.17%)	{85} (s:100%) {86} (s:97.54%) {85, 86} (s:97.54%) {34} (s:97.42%) {85, 34} (s:97.42%) {86, 34} (s:97.32%) {86, 85, 34} (s:97.32%) {90} (s:92.17%) {85, 90} (s:92.17%)

Supp (%)	Apriori	Apriori cải tiến	FPGrowth	FPGrowth song song	Apriori cải tiến song song
	{86, 90} (s:89.71%)	{86, 90} (s:89.71%)	{86, 90} (s:89.71%)	{86, 90} (s:89.71%)	{86, 90} (s:89.71%)
	{86, 85, 90} (s:89.71%)	{86, 85, 90} (s:89.71%)	{86, 85, 90} (s:89.71%)	{85, 86, 90} (s:89.71%)	{86, 85, 90} (s:89.71%)
	{34, 90} (s:89.81%)	{34, 90} (s:89.81%)	{34, 90} (s:89.81%)	{34, 90} (s:89.81%)	{34, 90} (s:89.81%)
	{34, 85, 90} (s:89.81%)	{34, 85, 90} (s:89.81%)	{34, 85, 90} (s:89.81%)	{34, 85, 90} (s:89.81%)	{34, 85, 90} (s:89.81%)
	{34, 86, 90} (s:89.71%)	{34, 86, 90} (s:89.71%)	{34, 86, 90} (s:89.71%)	{34, 86, 90} (s:89.71%)	{34, 86, 90} (s:89.71%)
	{34, 86, 85, 90} (s:89.71%)	{34, 86, 85, 90} (s:89.71%)	{34, 86, 85, 90} (s:89.71%)	{34, 85, 86, 90} (s:89.71%)	{34, 86, 85, 90} (s:89.71%)
	{36} (s:83.85%)	{36} (s:83.85%)	{36} (s:83.85%)	{36} (s:83.85%)	{36} (s:83.85%)
	{85, 36} (s:83.85%)	{85, 36} (s:83.85%)	{85, 36} (s:83.85%)	{36, 85} (s:83.85%)	{85, 36} (s:83.85%)
	{86, 36} (s:81.49%)	{86, 36} (s:81.49%)	{86, 36} (s:81.49%)	{36, 86} (s:81.49%)	{86, 36} (s:81.49%)
	{86, 85, 36} (s:81.49%)	{86, 85, 36} (s:81.49%)	{86, 85, 36} (s:81.49%)	{36, 85, 86} (s:81.49%)	{86, 85, 36} (s:81.49%)
	{34, 36} (s:81.27%)	{34, 36} (s:81.27%)	{34, 36} (s:81.27%)	{34, 36} (s:81.27%)	{34, 36} (s:81.27%)
	{34, 85, 36} (s:81.27%)	{34, 85, 36} (s:81.27%)	{34, 85, 36} (s:81.27%)	{34, 36, 85} (s:81.27%)	{34, 85, 36} (s:81.27%)
	{34, 86, 36} (s:81.27%)	{34, 86, 36} (s:81.27%)	{34, 86, 36} (s:81.27%)	{34, 36, 86} (s:81.27%)	{34, 86, 36} (s:81.27%)
	{34, 86, 85, 36} (s:81.27%)	{34, 86, 85, 36} (s:81.27%)	{34, 86, 85, 36} (s:81.27%)	{34, 36, 85, 86} (s:81.27%)	{34, 86, 85, 36} (s:81.27%)

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đã trình bày và so sánh hiệu quả về một số thuật toán khai phá luật kết hợp và thuật toán khai phá luật kết hợp sử dụng giải thuật song song, qua đó ta thấy thuật toán sử dụng giải thuật song song đã giải quyết được vấn đề khai phá dữ liệu trên dữ liệu lớn về tốc độ xử lý.

Trong tương lai chúng tôi sẽ tiếp tục nghiên cứu sâu hơn về các thuật toán khai phá luật kết hợp sử dụng giải thuật song song, tìm cách cải tiến và khắc phục nhược điểm của các giải thuật song song hiện có, xây dựng các thuật toán mới nhằm đạt hiệu quả tốt hơn.

TÀI LIỆU THAM KHẢO

[1] R. Agrawal; , R. Srikant, "Fast algorithms for minning association rules," in In 20th VL.DBConf,, 1994.

[2] Jiawei Han , Jian Pei , Yiwen Yin, "Mining Frequent Patterns without Candidate Generation," in SIGMOD, 2000.

[3] Z. H. Deng; , Z. Wang; , J. Jiang, A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists. SCIENCE CHINA Information Sciences, 55 (9), 2008 - 2030, 2012.

[4] Aiman Moyaid SaidA; , Dr. P D D. DominicB; , Dr. Azween B AbdullahC, "A Comparative Study of FP-growth Variations," In IJCSNS International Journal of Computer Science and Network Security, no. VOL.9 No.5, pp. 266-272, May 2009.

[5] Zhi-HongDeng and Sheng-LongLv, "Fast mining frequent itemsets using Nodesets," Expert Systems with Applications, no. Volume 41, Issue 10, pp. 4505-4512, August 2014.
[6] Dawen Xia, Yanhui Zhou, Zhuobo Rong and Zili Zhang, "IPFP: An Improved Parallel FP-Growth Algorithm for Frequent Itemsets Mining," Proceedings 59th ISI World Statistics Congress, vol. Hong Kong (Session CPS026), p. 4034, 25-30 August 2013.
[7] Yanbin Ye and Chia-Chu Chiang, "A Parallel Apriori Algorithm for Frequent Itemsets Mining," in Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), Seattle, WA, USA, 2006.
[8] Yi Wang , Haoyuan Li , Dong Zhang , Ming Zhang , Edward Chang, "PFP: Parallel FP-Growth for Query Recommendation," in ACM, 2001.
[9] Girja Shankar , Latita Bargadiya, "A New Improved Apriori Algorithm For Association Rules Mining," International Journal of Engineering Research & Technology (IJERT), vol. 2, no. 6, June 2013.
[10] Douglas Gregor, Benjamin Martin, MPI.NET Tutorial in C#, Open Systems Laboratory, Indiana University., 2008.

Tác giả chịu trách nhiệm bài viết:
Nguyễn Thành Sơn
Trường Đại học Sư phạm Kỹ thuật TP.HCM
Email: sonnt@hcmute.edu.vn