COOC-CFI: THUẬT TOÁN HIỆU QUẢ KHAI THÁC TẬP PHỔ BIẾN ĐÓNG TRÊN DỮ LIỆU GIAO DỊCH

*Phan Thành Huấn**

Title: *Cooc-cfi: An efficient mining algorithm for closed frequent itemsets in transaction databases*

Từ khóa: *Từ khóa: Luật kết hợp, tập phổ biến đóng, tập mục đồng xuất hiện.*

Keywords: *Association rule, closed frequent itemsets, co-occurrence itemset.*

Thông tin chung:
Ngày nhận bài: 29/9/2016;
Ngày nhận kết quả bình duyệt: 13/3/2017;
Ngày chấp nhận đăng bài: 06/9/2017.

Tác giả:
ThS., Đại học Khoa học Xã hội và Nhân văn, Đại học Quốc gia Tp. Hồ Chí Minh

huanphan@hcmussh.edu.vn

TÓM TẮT

Khai thác luật kết hợp là một trong những kỹ thuật quan trọng và được nghiên cứu nhiều trong khai thác dữ liệu. Khai thác tập phổ biến đóng là một trong những vấn đề cơ bản trong khai thác luật kết hợp. Hầu hết các thuật toán sinh không gian tìm kiếm dựa trên tập mục thỏa ngưỡng phổ biến tối thiểu và không dùng lại cho lần khai thác tiếp theo. Để khắc phục vấn đề này, chúng tôi đề xuất một cách tiếp cận mới để tìm tập phổ biến đóng trên dữ liệu giao dịch dùng cấu trúc dữ liệu lưu trữ dạng bit và tập chỉ mục chứa tập mục đồng xuất hiện để chiếu tính nhanh tập phổ biến đóng. Sau cùng, chúng tôi trình bày kết quả thực nghiệm, cho thấy thuật toán đề xuất tốt hơn so với các thuật toán hiện hành.

ABSTRACT

Association rule mining is one of the most important and well-researched techniques of Data Mining. Mining closed frequent itemsets is one of the most fundamental problems in association rule mining. Most of algorithms in literature used to find frequent itemsets on search space items, which have a support greater than minsup and not reuse for mining next time. To overcome this problem, we propose a new approach to fast dectect closed frequent itemsets using data structure on bit and array co-occurrence itemset of kernel item for fast mining closed frequent itemsets. Finally, the result showed the proposed algorithm which was better than the existing algorithms.

1. Giới thiệu

Khai thác luật kết hợp là một kỹ thuật quan trọng trong lĩnh vực khai thác dữ liệu. Mục tiêu khai thác là phát hiện những mối quan hệ giữa các giá trị dữ liệu trong cơ sở dữ liệu (CSDL). Mô hình đầu tiên của bài toán khai thác luật kết hợp là mô hình nhị phân hay còn gọi là mô hình cơ bản (Agrawal, R., & Imilienski, T., & Swami, A., 1993), phân tích cơ sở dữ liệu giao dịch, phát hiện các mối quan hệ giữa các tập mục hàng hoá đã bán được tại các siêu thị. Từ đó có kế hoạch bố trí, sắp xếp, kinh doanh hợp lý, đồng thời tổ chức sắp xếp các quầy gần nhau như thế nào để có doanh thu cao trong các phiên giao dịch tiếp theo. Ngoài ra, có thể áp dụng tri thức này để dự đoán số lượng các mặt hàng được bán chạy trong thời gian sắp tới. Tổng hợp các tri thức này để lên kế hoạch cho hoạt động, sản xuất, kinh doanh một cách thuận tiện hơn nhằm giảm bớt thời gian thống kê, tìm hiểu thị trường,...

Các thuật toán được đề xuất để khai thác luật kết hợp chia thành 2 giai đoạn (Agrawal, R., & Imilienski, T., & Swami, A., 1993 ; Agrawal, R., & Srikant, R., 1994):

Giai đoạn 1: Tìm tất cả các tập phổ biến (**FI**) từ CSDL nghĩa là tìm tất cả các tập mục X có tần số xuất hiện lớn hơn hoặc bằng

ngưỡng phổ biến tối thiểu. Đây là giai đoạn tốn khá nhiều thời gian xử lý.

Giai đoạn 2: Sinh các luật tin cậy kết hợp từ các tập phổ biến tìm thấy ở giai đoạn thứ nhất. Giai đoạn này tương đối đơn giản và tốn kém ít thời gian hơn so với giai đoạn trên.

Trong thực tế, giai đoạn thứ nhất chiếm hầu hết thời gian cho toàn quá trình khai thác luật kết hợp. Nhằm cải tiến về mặt thời gian, đề xuất thay thế tập **FI** bằng tập nhỏ hơn, gọi là tập hợp các tập phổ biến đóng (**CFI**), tập **CFI** vẫn đầy đủ thông tin cho giai đoạn thứ hai.

Một số thuật toán khai thác tập phổ biến đóng **CFI** đã được các tác giả trên thế giới đề xuất: **Charm** (Zaki, M. J., & Hsiao, C., 2002), **CLOSET+** (Wang, J., & Han, J., & Pei, J., 2003) và gần đây là thuật toán **DBV-Miner** (Vo, B., & Hong, T. P., & Le, B., 2012). Các thuật toán trên, mỗi lần khai thác tập phổ biến đóng chỉ xem xét các mục hàng thỏa ngưỡng phổ biến tối thiểu *minsup*. Các thuật toán này chưa đáp ứng thực tế, khi cần khai thác luật kết hợp thì người dùng có thể yêu cầu thực hiện khai thác luật kết hợp thỏa ngưỡng *minsup* và *minconf* trong nhiều chuỗi thao tác liên tiếp nhau. Vì vậy, tác giả đề xuất thuật toán khai thác hiệu quả tập phổ biến đóng **COOC-CFI**, gồm các thuật toán con như sau:

– Xây dựng mảng **Index_COOC** gồm tập mục đồng xuất hiện của từng mục hàng;

– Thuật toán khai thác hiệu quả tập phổ biến đóng **COOC-CFI** dựa trên mảng **Index_COOC** chứa các tập mục đồng xuất hiện.

Trong phần 2, bài báo trình bày các vấn đề liên quan về tập phổ biến đóng và cấu trúc lưu trữ dữ liệu giao dịch. Phần 3, xây dựng thuật toán xác định mảng chứa tập mục đồng xuất hiện của từng mục hàng và thuật toán hiệu quả khai thác tập phổ biến đóng. Kết quả thực nghiệm được trình bày trong phần 4 và kết luận ở phần 5.

2. Các vấn đề liên quan

2.1 Một số khái niệm cơ bản

Cho $I = \{I_1, I_2, ..., I_m\}$ là tập gồm m thuộc tính riêng biệt, mỗi thuộc tính gọi là *item*. Tập mục $X \subseteq I$ gọi là *itemset*, tập mục có k mục gọi là *k-itemset*. $\mathcal{D}$ là dữ liệu giao dịch, gồm n bản ghi phân biệt gọi là tập các giao dịch $T = \{T_1, T_2, ..., T_n\}$, mỗi giao dịch $T_i = \{ I_{k_1}, I_{k_2}, ..., I_{k_j} \}, I_{k_j} \in I (1 \le k_j \le m)$.

Định nghĩa 1: *Độ phổ biến* (support) của *itemset* $X \subseteq I$, ký hiệu *sup(X)*, là số các giao dịch trong $\mathcal{D}$ có chứa X.

Định nghĩa 2: Cho $X \subseteq I$, X gọi là *itemset* phổ biến nếu *sup(X)* ≥ *minsup*, trong đó *minsup* là độ phổ biến tối thiểu. Ký hiệu **FI** là tập hợp các tập mục phổ biến.

Định nghĩa 3: Cho $X \subseteq I$, X được gọi là *itemset* phổ biến đóng nếu X là tập mục phổ biến và không có tập cha cùng độ phổ biến. Tập các *itemset* phổ biến đóng gọi là tập hợp các tập mục phổ biến đóng, ký hiệu là **CFI**.

Dữ liệu giao dịch $\mathcal{D}$

Mã giao dịch	Tập item
T1	A C E F
T2	A C G
T3	E H
T4	A C D F G
T5	A C E G
T6	E
T7	A B C E
T8	A C D
T9	A B C E G
T10	A C E F G

Ví dụ 1: Cho dữ liệu giao dịch $\mathcal{D}$ như trong *Bảng 1*, có 8 *item* riêng biệt $I = \{A, B, C, D, E, F, G, H\}$ và 10 giao dịch $T = \{T1, T2, T3, T4, T5, T6, T7, T8, T9, T10\}$ phân biệt.

Tập FI, CFI với minsup = 3 và minsup = 5 trên dữ liệu giao dịch Ɗ

k-itemset	Tập phổ biến **FI** (*minsup*=3)	Tập phổ biến đóng **CFI** (*minsup*=3)	Tập phổ biến **FI** (*minsup*=5)	Tập phổ biến đóng **CFI** (*minsup*=5)
1	F, G, E, A, C	***E***	G, E, A, C	***E***
2	FA, FC, EG, GA, GC, EA, EC, AC	***AC***	GA, GC, EA, EC, AC	***AC***
3	FAC, GEA, GEC, GAC, EAC	***FAC, GAC, EAC***	GAC, EAC	***GAC, EAC***
4	GEAC	***GEAC***		

Trong *Bảng 2*, cho thấy tập phổ biến **FI** và tập phổ biến đóng **CFI** chứa *k-itemset* với *minsup* = 3 (3 giao dịch) và *minsup* = 5 (5 giao dịch). Trường hợp *minsup* = 3, $|FI|$ = 19 và $|CFI|$ = 6, tỷ suất $|CFI|/|FI| = 6/19 \times 100\% = 31\%$; *minsup* = 5, tỷ suất $|CFI|/|FI| = 4/11 \times 100\% = 36\%$. Qua đó, ta thấy số lượng tập phổ biến đóng nhỏ hơn rất nhiều so với số lượng tập phổ biến.

2.2. Tổ chức lưu trữ dữ liệu giao dịch

Lưu trữ dữ liệu giao dịch dạng *bit* là cấu trúc dữ liệu hiệu quả trong khai thác tập phổ biến (Dong, J., & Han, M., 2007 ; Song, W., & Yang, B., 2008). Chuyển đổi dữ liệu giao dịch thành ma trận nhị phân **BiM**, trong đó mỗi dòng tương ứng với một giao dịch và mỗi cột tương ứng với một *item*. Nếu *item* thứ *i* xuất hiện trong giao dịch *t* thì *bit* thứ *i* của dòng *t* trong **BiM** sẽ mang giá trị *1*, ngược lại sẽ mang giá trị *0*.

Mã giao dịch	A	B	C	D	E	F	G	H
T1	1	0	1	0	1	1	0	0
T2	1	0	1	0	0	0	1	0
T3	0	0	0	0	1	0	0	1
T4	1	0	1	1	0	1	1	0
T5	1	0	1	0	1	0	1	0
T6	0	0	0	0	1	0	0	0
T7	1	1	1	0	1	0	0	0
T8	1	0	1	1	0	0	0	0
T9	1	1	1	0	1	0	1	0
T10	1	0	1	0	1	1	1	0

Hình 1. *Biểu diễn dạng bit của dữ liệu giao dịch 'D*

3. Các thuật toán

3.1. Thuật toán sinh itemset đồng xuất hiện

3.1.1. Tập chiếu và mảng itemset đồng xuất hiện

Tập chiếu của *item* I_k trên dữ liệu giao dịch $\mathcal{D}$: $\pi(I_k)=\{t \in \mathcal{D} \mid I_k \in t\}$ là tập các giao dịch có chứa *item* I_k (π-đơn điệu giảm).

Ví dụ 2: Theo Bảng 1, có π(A) = {1, 2, 4, 5, 7, 8, 9, 10} và π(B) = {7, 9}. Để tính π(AB), chúng ta chỉ cần lấy phần giao của π(A) với π(B), nghĩa là π(AB) = π(A)∩π(B)= {1, 2, 4, 5, **7**, 8, **9**, 10}∩{**7**, **9**} = {**7**, **9**}, π(B) ⊆ π(A).

Định nghĩa 4: Cho $I_k \in I$, ta gọi I_k là *item hạt nhân.* Tập $X_{cooc} \subseteq I$ gọi *đồng xuất hiện* với I_k: X_{cooc} là tập các *item* xuất hiện cùng I_k thì $\pi(I_k) \equiv \pi(I_k \cup X_{cooc})$. Ký hiệu, $cooc(I_k) = X_{cooc}$.

Ví dụ 3: Cho dữ liệu giao dịch $\mathcal{D}$ như trong *Bảng 1*. Xem *item B* là *item hạt nhân*, ta xác định được *itemset* đồng xuất hiện cùng độ phổ biến với item *B* là *cooc*(*B*) = {A, C, E} và *sup*(B) = *sup*(B*ACE*) = 2 (*theo định nghĩa 4*).

Định nghĩa 5: Cho $I_k \in I$ ($I_1 \prec I_2 \prec \ldots \prec I_m$) thứ tự theo độ phổ biến, ta gọi I_k là *item hạt nhân.* Tập $X_{lexcooc} \subseteq I$ gọi *đồng xuất hiện có thứ tự* với *item* I_k: $X_{lexcooc}$ là tập các *item* xuất hiện cùng I_k và $\pi(I_k) \equiv \pi(I_k \cup X_{lexcooc})$, $I_k \prec I_j$,$\forall I_j \in X_{lexcooc}$. Ký hiệu, $lexcooc(I_k) = X_{lexcooc}$.

Bổ đề 1: $\forall I_k \prec I_j$, nếu $sup(I_k) = minsup$ và $I_j \notin lexcooc(I_k)$ thì $sup(I_k \cup I_j) < minsup$.

Chứng minh: $sup(I_k \cup I_j) < sup(I_k)$, hiển nhiên $\pi(I_k \cup I_j) = \pi(I_k) \cap \pi(I_j) \subset \pi(I_k)$ ■.

Ví dụ 4: *minsup* = 2, xét *item* B và D. Ta có, *sup*(B) = *minsup* và *sup*(BD) = 0 < *sup(B)*.

Bổ đề 2: $lexcooc(I_k) = X_{lexcooc}$ thì $sup(I_k \cup Y_{sub}) = sup(I_k)$, $\forall Y_{sub} \subseteq X_{lexcooc}$.

Chứng minh: $lexcooc(I_k) = X_{lexcooc}$, giả sử $X_{lexcooc}$ gồm k *item* thì có $2^k - 1$ tập con. Với $Y_{sub} \subseteq X_{lexcooc}$ thì ta có $\pi(I_k \cup Y_{sub}) = \pi(I_k) \cap \pi(Y_{sub}) = \pi(I_k)$ ■.

Ví dụ 5: Xét *item* G, với *sup*(G)=5. Ta có, *lexcooc*(**G**) = {A, C} thì 3 *itemset* kết hợp {A, C, AC} và *sup*(**G**) = *sup*(**G**A) = *sup*(**G**C) = *sup*(**G**AC) = 5.

Bổ đề 3: $\forall I_k \prec I_j$ và $I_j \notin X_{lexcooc}$ thì $sup(I_j \cup I_k) = sup(I_j \cup I_k \cup Y_{sub})$ và $sup(I_j \cup I_k \cup Y_{sub}) < sup(I_k \cup Y_{sub})$, $\forall Y_{sub} \subseteq X_{lexcooc}$.

Chứng minh: (theo Bổ đề 2) $\pi(I_k \cup Y_{sub}) = \pi(I_k)$ thì $\pi(I_j \cup I_k \cup Y_{sub}) = \pi(I_j \cup I_k)$ ■.

Ví dụ 6: Xét *item* G, với *sup*(G)=5 và I_j = E. Ta có, *lexcooc*(**G**) = {A, C} thì 3 *itemset* kết hợp {A, C, AC} và *sup*(**GE**) = *sup*(**GE**A) = *sup*(**GE**C) = *sup*(**GE**AC) = 3 < *sup*(**G**AC) = 5.

3.1.2. Thuật toán sinh itemset đồng xuất hiện

Dưới đây là thuật toán sinh các *item* đồng xuất hiện với từng *item* trong dữ liệu giao dịch và lưu trữ vào mảng **Index_COOC**. Mỗi phần tử trong **Index_COOC** gồm 3 thành phần sau:

– **Index_COOC[j].***item*: Lưu trữ *item hạt nhân thứ j*;

– **Index_COOC[j].***sup*: Lưu trữ *độ phổ biến* của *item hạt nhân thứ j*;

– **Index_COOC[j].***cooc*: Lưu *itemset* đồng xuất hiện cùng *item hạt nhân thứ j* dạng bit;

Ngoài ra, thuật toán 1 còn thực hiện nén dữ liệu giao dịch vào ma trận **BiM**.

Mã giả thuật toán 1. Xây dựng bảng Index_COOC

Đầu vào*:* Dữ liệu giao dịch $\mathcal{D}$

Đầu ra*:* Mảng **Index_COOC**, ma trận **BiM**

1. Với mỗi phần tử *j* của mảng **Index_COOC** thực hiện:
2. ***Index_COOC***[j].*item* = Ij
3. ***Index_COOC***[j].*sup* = 0
4. ***Index_COOC***[j].*cooc*= 2^m - 1
5. Với mỗi giao dịch T_i thực hiện:
6. Nén giao dịch T_i vào ma trận **BiM**
7. Với mỗi item *j* có trong giao dịch T_i thực hiện:
8. ***Index_COOC***[j].*cooc* = ***Index_COOC***[j].*cooc* & *vectorbit*(T_i)
9. ***Index_COOC***[j].*sup* = ***Index_COOC***[j].*sup* + 1
10. Sắp xếp mảng **Index_COOC** tăng dần theo *sup*
11. Với mỗi phần tử *j* của mảng **Index_COOC**:
12. ***Index_COOC***[j].*cooc*= *lexcooc*(I_j)
13. Trả về mảng **Index_COOC,** ma trận **BiM**

Từ *dòng 1* đến *dòng 4* là các bước khởi tạo cho mảng **Index_COOC**. *Dòng 5* duyệt dữ liệu giao dịch, ứng với từng giao dịch ta xem xét có chứa *item* thứ *j* thì thực hiện phép toán **AND** trên *bit* để xác định các phần tử cùng xuất hiện với *item j*.

Khởi tạo mảng **Index_COOC**: (*thành phần cooc biểu diễn dạng bit*) số *item* là m = 8

item	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
sup	0	0	0	0	0	0	0	0
cooc	*11111111*	*11111111*	*11111111*	*11111111*	*11111111*	*11111111*	*11111111*	*11111111*

Đọc giao dịch T1: {A, C, E, F} có biểu diễn dạng bit là **10101100**

item	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
sup	**1**	0	**1**	0	**1**	**1**	0	0
cooc	**10101100**	11111111	**10101100**	11111111	**10101100**	**10101100**	11111111	11111111

Đọc giao dịch T2: {A, C, G} có biểu diễn dạng bit là **10100010**

item	A	B	C	D	E	F	G	H
sup	**2**	0	**2**	0	1	1	**1**	0
cooc	**10100000**	11111111	**10100000**	11111111	10101100	10101100	**10100010**	11111111

Đọc giao dịch T3: {E, H} có biểu diễn dạng bit là **00001001**

item	A	B	C	D	E	F	G	H
sup	2	0	2	0	**2**	1	1	**1**
cooc	10100000	11111111	10100000	11111111	**00001000**	10101100	10100010	**00001001**

Đọc giao dịch T4: {A, C, D, F, G} có biểu diễn dạng bit là **10110110**

item	A	B	C	D	E	F	G	H
sup	**3**	0	**3**	**1**	2	**2**	**2**	1
cooc	**10100000**	11111111	**10100000**	**10110110**	00001000	**10100100**	**10100010**	00001001

Đọc giao dịch T5: {A, C, E, G} có biểu diễn dạng bit là **10101010**

item	A	B	C	D	E	F	G	H
sup	**4**	0	**4**	1	**3**	2	**3**	1
cooc	**10100000**	11111111	**10100000**	10110110	**00001000**	10100100	**10100010**	00001001

Đọc giao dịch T6: {E} có biểu diễn dạng bit là **00001000**

item	A	B	C	D	E	F	G	H
sup	4	0	4	1	**4**	2	3	1
cooc	10100000	11111111	10100000	10110110	**00001000**	10100100	10100010	00001001

Đọc giao dịch T7: {A, B, C, E} có biểu diễn dạng bit là **11101000**

item	A	B	C	D	E	F	G	H
sup	**5**	**1**	**5**	1	**5**	2	3	1
cooc	**10100000**	**11101000**	**10100000**	10110110	**00001000**	10100100	10100010	00001001

Đọc giao dịch T8: {A, C, D} có biểu diễn dạng bit là **10110000**

item	A	B	C	D	E	F	G	H
sup	**6**	1	**6**	**2**	5	2	3	1
cooc	**10100000**	11101000	**10100000**	**10110000**	00001000	10100100	10100010	00001001

Đọc giao dịch T9: {A, B, C, E, G} có biểu diễn dạng bit là **11101010**

item	A	B	C	D	E	F	G	H
sup	**7**	**2**	**7**	2	**6**	2	**4**	1
cooc	**10100000**	**11101000**	**10100000**	10110000	**00001000**	10100100	**10100010**	00001001

Đọc giao dịch T10: {A, C, E, F, G} có biểu diễn dạng bit là **10101110**

item	A	B	C	D	E	F	G	H
sup	**8**	2	**8**	2	**7**	**3**	**5**	1
cooc	**10100000**	11101000	**10100000**	10110000	**00001000**	**10100100**	**10100010**	00001001

Kết thúc vòng lặp *ở dòng 5*, trả về mảng **Index_COOC** tương ứng là *cooc*(A) = {C}, *cooc*(B) = {A, C, E}, *cooc*(C) = {A}, *cooc*(D) = {A, C}, *cooc*(E) = {}, *cooc*(F) = {A, C}, *cooc*(G) = {A, C} và *cooc*(H) = {E}.

Trả về mảng Index_COOC sắp tăng theo độ phổ biến của item.

item	**H**	**B**	**D**	**F**	**G**	**E**	**A**	**C**
sup	1	2	2	3	5	7	8	8
cooc	E	A, C, E	A, C	A, C	A, C	Ø	C	A

Thực hiện *dòng 10*: Ta có kết quả trong Bảng 3, mảng **Index_COOC** chứa *itemset* đồng xuất hiện của từng *item* và được sắp tăng dần theo độ phổ biến của từng *item*.

Trả về mảng Index_COOC sắp tăng theo độ phổ biến của item và thành phần cooc có thứ tự.

item	**H**	**B**	**D**	**F**	**G**	**E**	**A**	**C**
sup	1	2	2	3	5	7	8	8
cooc	E	E, A, C	A, C	A, C	A, C	Ø	C	Ø

Thực hiện *dòng 11* và 12: ta có kết quả trong Bảng 4, mảng **Index_COOC** chứa thành phần *cooc* có thứ tự (*theo định nghĩa 5*).

3.2 Thuật toán khai thác tập phổ biến đóng COOC-CFI

Theo khảo sát, một số thuật toán khai thác tập phổ biến đóng đã được các tác giả trên thế giới đề xuất như **Charm** (Zaki, M. J., & Hsiao, C., 2002), **CLOSET+** (Wang, J., & Han, J., & Pei, J., 2003) và **DBV-Miner** (Vo, B., & Hong, T. P., & Le, B., 2012) chưa đáp ứng nhu cầu thực tế: Mỗi lần cần khai thác tập phổ biến đóng với *minsup* khác thì thuật toán thực hiện chọn lại các *item* thỏa *minsup*, phát sinh lại cây tìm kiếm hoặc dàn tập mục tương ứng và xác định tập phổ biến đóng thỏa *minsup* mới. Trong thực tế, khi cần khai thác luật kết hợp thì người dùng có thể yêu cầu thực hiện khai thác luật kết hợp thỏa ngưỡng *minsup* và *minconf* trong nhiều chuỗi thao tác liên tiếp nhau.

Từ đó, tác giả đã xây dựng thuật toán **COOC-CFI** khai thác tập phổ biến đóng dựa trên mảng **Index_COOC** chứa tất cả các *itemset* đồng xuất hiện của tất cả *item* trong CSDL có thể thực hiện chuỗi khai thác nhanh tập **CFI** và dùng lại mảng **Index_COOC** cho lần sau.

Mã giả thuật toán 2. COOC-CFI khai thác tập phổ biến đóng

Đầu vào*:* mảng **Index_COOC**, *minsup*

Đầu ra*:* Tập phổ biến đóng **CFI**

1. Với mỗi *item* thỏa *minsup*, xem xét:
2. **Nếu** *sup*(*item*) = *minsup* **thì**
3. **Nếu** ***IS_CFI***(**CFI**, {*item* ∪ *cooc*(*item*)) **thì**
4. **CFI** = **CFI** ∪ {*item* ∪ *cooc*(*item*)} //bổ đề 1 và 2
5. **Ngược lại**
6. $J_k \leftarrow$ ***tập con***$(l_j \prec l_k)$ //sắp xếp tăng theo sup
7. **Nếu** ***IS_CFI***(**CFI**, {*item* ∪ *cooc*(*item*)) **thì**
8. **CFI** = **CFI** ∪ {*item* ∪ *cooc*(*item*)} //bổ đề 1 và 2
9. **Nếu** $J_k \neq \varnothing$ **thì**
10. Với mỗi *itemset* $IS_i \in J_k$ **//**bổ đề 3
11. **Nếu** ***IS_CFI***(**CFI**, IS_i ∪{*item* ∪ *cooc*(*item*)) **thì**
12. **CFI**= **CFI** ∪ IS_i ∪{*item* ∪ *cooc*(*item*)
13. Trả về tập phổ biến đóng **CFI**

Ví dụ 7: Cho dữ liệu giao dịch $\mathcal{D}$ như trong *Bảng 1* và minsup = 3

Xét *item* F, có *cooc*(F) = {A, C} (thỏa điều kiện dòng 2), sinh tập mục phổ biến đóng là (**F**AC, 3). Lúc này, **CFI** ={(**F**AC, 3)}.

Xét *item* G (*ngược lại - dòng 5*), lúc này *item* G – *cooc*(G) = {A, C},có *sup(G)=5* > *minsup*. Tập J_k ={E}. Tập mục phổ biến đóng sinh ra trong bước này là *cfi* = {(**G**EAC, 3), (**G**AC, 5)}. Kết thúc bước này, ta có **CFI**={(**F**AC, 3), (**G**EAC, 2), (**G**AC, 5)}.

Xét item E, có *cooc*(E) = {} còn *item* A và C có *sup*(A) = *sup*(C) = 8 > *minsup*. Tập J_k ={A, C, AC}. Tập mục phổ biến đóng sinh ra trong bước này là *cfi* = {(**E**AC, 5)} với *sup*(EAC) = *sup*(GAC) nhưng EAC $\not\subset$ GEAC trong **CFI**, nên đưa *cfi* ={(**E**AC, 5)} vào **CFI**.

Xét *item* A, có *cooc*(A) = {C}. Tập J_k={}, sinh tập mục phổ biến đóng là *cfi*={(**A**C, 8)}. Lúc này, **CFI** =**CFI** $\cup$ {(**A**C, 8)} vì **CFI** không có tập mục phổ biến đóng có độ phổ biến là 8.

Sau cùng, chỉ còn *item* C và *cooc*(C)={}: sinh tập mục phổ biến đóng (**C**, 8), mà C $\subset$ AC và *sup*(C) = *sup*(AC), nên không thêm vào **CFI**.

Với dữ liệu giao dịch $\mathcal{D}$ ở Bảng 1 và *minsup* = 3, ta có: **CFI**={(**E**AC, 3), (**G**EAC, 3), (**G**AC, 5), (**E**AC, 5) (**E**, 7), (**A**C, 8)}.

4. Kết quả thực nghiệm

Thực nghiệm trên máy tính *Panasonic CF-74*, Core Duo 2.0 GHz, 4 GB RAM, thuật toán cài đặt trên C#, Microsoft Visual Studio 2010.

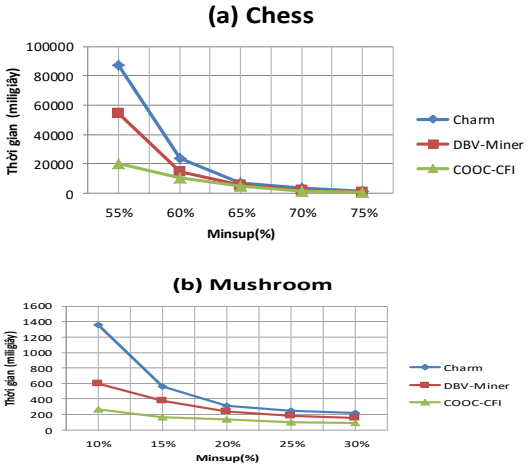
Nghiên cứu thực nghiệm trên *hai nhóm dữ liệu*:

Nhóm dữ liệu thực có mật độ dày: Sử dụng dữ liệu thực từ kho dữ liệu về học máy của trường Đại học California (Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science) gồm 2 tập **Chess** và **Mushroom**.

Nhóm dữ liệu giả lập có mật độ thưa: Sử dụng phần mềm phát sinh dữ liệu giả lập của trung tâm nghiên cứu IBM Almaden (IBM Almaden Research Center, San Joe, California 95120, U.S.A [http://www.almaden.ibm.com]) gồm 2 tập **T10I4D100K** và **T40I10D100K**.

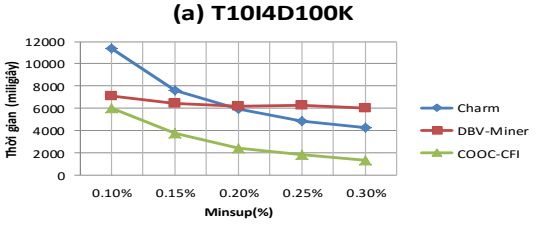
Dữ liệu thực nghiệm

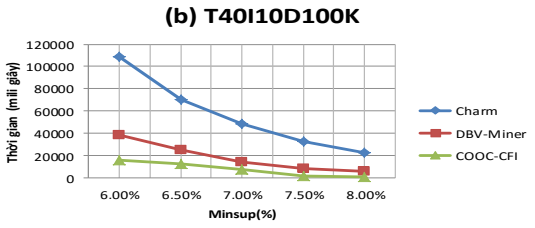
Tên dữ liệu	Số item	Số giao dịch	Số mục nhỏ nhất/giao dịch	Số mục lớn nhất/ giao dịch	Số mục trung bình/ giao dịch	Mật độ (%)
Chess	75	3.196	37	37	37	49,3%
Mushroom	119	8.142	23	23	23	19,3%
T10I4D100K	870	100.000	1	29	10	1,1%
T40I10D100K	942	200.000	4	77	40	4,2%



Hình 2. *Thời gian thực hiện* ***COOC-CFI, Charm*** *và* ***DBV-Miner*** *trên dữ liệu* ***Chess, Mushroom*** *với các minsup khác nhau.*

Tác giả sử dụng 2 tập **Chess** và **Mushroom** để so sánh hiệu suất của thuật toán **COOC-CFI** với thuật toán **Charm (**Zaki, M. J., & Hsiao, C., 2002) và **DBV-Miner (**Vo, B., & Hong, T. P., & Le, B., 2012). Hình 2, cho thấy thời gian thực hiện thuật toán **COOC-CFI** khai thác tập phổ biến đóng theo các ngưỡng *minsup* khác nhau trên 2 tập dữ liệu **Chess** và **Mushroom** nhanh hơn thuật toán **Charm** và **DBV-Miner**.





Hình 3. *Thời gian thực hiện COOC-CFI, Charm và DBV-Miner trên dữ liệu T10I4D100K và T40I10D100K với các minsup khác nhau.*

Ngoài ra, tác giả cũng sử dụng 2 tập **T10I4D100K** và **T40I10D100K** để so sánh hiệu suất của thuật toán **COOC-CFI** với thuật toán **Charm** và **DBV-Miner**. Hình 3, cho thấy thời gian thực hiện thuật toán **COOC-CFI** khai thác tập phổ biến đóng theo các ngưỡng *minsup* khác nhau trên 2 tập dữ liệu trên là nhanh hơn thuật toán **Charm** và **DBV-Miner**.

5. Kết luận

Bài báo đã đề xuất thuật toán tính nhanh mảng **Index_COOC** chứa các *itemset* đồng xuất hiện và thuật toán **COOC-CFI** khai thác hiệu quả tập phổ biến đóng dựa trên mảng **Index_COOC**. Kết quả trên cho thấy thuật toán khai thác tập phổ biến đóng **COOC-CFI** tốt hơn thuật toán **Charm**, **DBV-Miner**. Ngoài ra, thuật toán cũng cần thực nghiệm thêm trên nhiều tập dữ liệu khác.

Trong tương lai, tác giả sẽ cải tiến thuật toán trên để có thể khai thác tập phổ biến đóng trên dữ liệu giao dịch có *trọng số*, đây là hướng nghiên cứu đang được quan tâm vì khả năng ứng dụng vào nhiều lĩnh vực, đặc biệt là trong kinh doanh.

TÀI LIỆU THAM KHẢO

1. Agrawal, R., & Imilienski, T., & Swami, A. (1993). Mining association rules between sets of large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, DC*, 207-216.

2. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of International Conference on Very Large Data Base, Santiago, Chile,* 478-499.

3. Dong, J., & Han, M. (2007). BitTableFI: An efficient mining frequent itemsets algorithm. *Knowledge-Based Systems 20(4)*, 329–335.

4. Song, W., & Yang, B. (2008). Index-BitTableFI: An improved algorithm for mining frequent itemsets. *Knowledge-Based Systems 21*, 507-513.

5. Vo, B., & Hong, T. P., & Le, B. (2012). DBV-miner: A dynamic bit-vector approach for fast mining frequent closed itemsets. *Expert Systems with Applications, 39(8),*7196–7206.

6. Zaki, M. J., & Hsiao, C. (2002). CHARM: An efficient algorithm for closed association rule mining. *In 2nd SIAM International Conference on Data Mining*, 457–473.

7. Wang, J., & Han, J., & Pei, J. (2003). CLOSET+: searching for the best strategies for mining frequent closed itemsets. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC*, USA, 236–245.