GIẢI THUẬT HIỆU NĂNG CAO KHAI THÁC TẬP SINH CỦA TẬP ĐÓNG PHỔ BIẾN

HIGH-PERFORMANCE ALGORITHM IN MINING GENERATORS OF CLOSED FREQUENT ITEMSETS

Phan Thành Huấn

Đại học Quốc gia Tp. Hồ Chí Minh; huanphan@hcmussh.edu.vn

Tóm tắt - Trong khai thác dữ liệu, khai thác luật kết hợp là một trong những kỹ thuật quan trọng và được nghiên cứu nhiều. Đặc biệt là kỹ thuật khai thác luật kết hợp chính xác và không dư thừa, một số tác giả đã đề xuất khai thác luật kết hợp này từ tập sinh của tập đóng phổ biến trên dữ liệu giao dịch.Trong bài viết này, chúng tôi đề xuất giải thuật song song MCP-GCFI khai thác nhanh tập sinh của tập đóng phổ biến trên bộ xử lý đa nhân. Giải thuật đề xuất dễ dàng mở rộng trên nhiều hệ thống tính toán phân tán như Hadoop, Spark. Kết quả thực nghiệm trên bộ dữ liệu thực có mật độ dày của UCI và bộ dữ liệu giả lập có mật độ thưa của trung tâm nghiên cứu IBM Almaden, cho thấy giải thuật đề xuất hiệu quả.

Từ khóa - Bộ xử lý đa nhân; tập đóng phổ biến; tập sinh của tập đóng; giải thuật song song MCP-GCFI

Abstract - Association-rule mining is one of the most important and well-researched techniques of Data Mining. Particularly, the techniques of mining are exact association rules and non-redundant, some authors have proposed mining these association rules from generators of closed frequent item sets. In this paper, we propose the parallel MCP-GCFI algorithm to fast mining generators of closed frequent item sets on Multi-Core processor, as well as to expand the algorithm on distributed computing systems such as Hadoop, Spark. The experimental results show that the proposed algorithms perform faster than other existing algorithms on both real-life datasets of UCI and synthetic datasets generated by IBM Almaden.

Key words - Multi-core processor; closed frequent itemset; generator itemsets; parallel MCP-GCFI algorithm.

1. Giới thiệu

Năm 1993, R. Agrawal đã đề xuất mô hình cơ bản khai thác luật kết hợp từ dữ liệu giao dịch dạng nhị phân theo hai giai đoạn [1, 2]: Phát sinh *tập phổ biến* (*Frequent Itemset* - **FI**) và sinh *luật kết hợp* từ tập phổ biến. Khi đó, số lượng itemset phổ biến được sinh ra là rất lớn và chiếm rất nhiều thời gian. Nhiều tác giả có đề xuất giải thuật sinh *tập đóng phổ biến* (*Closed Frequent Itemset* - **CFI**) [3-5], đây là tập không dư thừa và có kích thước nhỏ hơn **FI** rất nhiều lần (**CFI**⊆**FI**). Tuy nhiên, để sinh các *luật kết hợp chính xác* cũng như *không dư thừa,* nhiều tác giả có đề xuất khai thác tập luật kết hợp từ tập sinh của tập đóng phổ biến (*Minimal/ Generator Closed Frequent Itemset* – **mGCFI/GCFI**) [6-10] như giải thuật **GrGrowth** [6], **Zart** [8] và **DefMe** [9, 10]. Các giải thuật trên tựa **Apriori**, tìm kiếm theo chiều sâu và dùng cấu trúc **FP-Tree**, **IT-Tree** – các giải thuật này tốn nhiều bộ nhớ/ đọc dữ liệu nhiều lần và không hiệu quả trên các loại dữ liệu giao dịch có mật độ *dày* hoặc *thưa*.

Trong nghiên cứu, tác giả đề xuất giải thuật *khai thác song song tập sinh*. Giải thuật đề xuất theo hướng tiếp cận song song dữ liệu và cả chức năng, dưới đây là các giải thuật liên quan trong bài viết:

- Phát sinh mảng **IndexCOOC** lưu trữ *items xuất hiện đồng thời* và *items xuất hiện trong ít nhấtmột giao dịch*với từng *item-hạt-nhân*;

- Dựa trên **IndexCOOC** phát sinh **nLOOCTree**;

- Giải thuật tuần tự **SEQ-GCFI** khai thác *tập sinh* của tập đóng phổ biến dựa trên cây **nLOOCTree**;

- Giải thuật song song **MCP-GCFI** khai thác nhanh *tập sinh* trên bộ xử lý đa nhân (BXLĐN).

Trong nghiên cứu này tác giả trình bày: Các vấn đề cơ bản về tập phổ biến, tập đóng phổ biến và *tập sinh* của tập đóng phổ biến; Giải thuật xây dựng mảng chứa *itemset* đồng xuất hiện và xuất hiện với *item-hạt-nhân* ít nhất trong một giao dịch, giải thuật sinh cây **nLOOCTree** và giải thuật tuần tự **SEQ-GCFI** khai thác *tập sinh;* Trên cơ sở giải thuật tuần tự xây dựng giải thuật song song **MCP-GCFI** khai thác hiệu năng của bộ xử lý đa nhân; Và cuối cùng là kết quả thực nghiệm, cũng như hướng phát triển cho các nghiên cứu tiếp theo.

2. Vấn đề cơ bản về tập phổ biến

Cho $\boldsymbol{I} = \{i_1, i_2,..., i_m\}$ là tập gồm m thuộc tính, mỗi thuộc tính được gọi là *item*. Tập các $itemX = \{i_1, i_2,..., i_k\}$, $\forall i_j \in I$ $(1 \le j \le k)$ được gọi là *itemset,* itemset gồm có k item thì được ký hiệu là *k-itemset*. $\mathcal{D}$ là dữ liệu chứa các giao dịch, gồm có n bản ghi dữ liệu được gọi là tập chứa các giao dịch $T = \{t_1, t_2,..., t_n\}$, với giao dịch $t_k = \{i_{k1}, i_{k2},..., i_{km}\}$, $i_{kj} \in I$ $(1 \le k_j \le m)$.

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

Định nghĩa 2: Cho itemset $X \subseteq \boldsymbol{I}$, nếu $sup(X) \ge minsup$ thì X được gọi là *itemset* phổ biến (*minsup:* ngưỡng phổ biến tối thiểu cho trước).

Định nghĩa 3: Cho $X \in$ **FI**, X gọi là *itemset* phổ biến đóng nếu X không có tập cha cùng độ phổ biến. Ký hiệu, **CFI** là tập chứa các *itemset* phổ biến đóng.

Định nghĩa 4: Cho $X \in$ **CFI**, tất cả các *itemset* con thực sự của X có *bằng độ phổ biến* với X gọi là *itemset* sinh của X. Ký hiệu, **GCFI** là tập chứa các *itemset* sinh.

Cho tập dữ liệu giao dịch $\mathcal{D}$ như trong Bảng 1.

Bảng 1. *Dữ liệu $\mathcal{D}$ sử dụng cho Ví dụ*

TID	Items							
	O	***Q***	***R***	***S***	***U***	***V***	***W***	***Z***
t1	1		1				1	
t2	1		1		1	1		
t3	1		1	1		1	1	

t4					1			1
t5					1			
t6	1		1		1		1	
t7	1		1	1				
t8	1	1	1		1			
t9	1		1		1	1	1	
t10	1	1	1		1		1	

Dữ liệu ở Bảng 1, gồm 8 *items* I ={O; Q; R; S; U; V; W; Z} và có 10 giao dịch T = {t1; t2; t3; t4; t5; t6; t7; t8; t9; t10}.

Bảng 2. *Sinh tập CFI và GCFI trên Ɗ với minsup = 3*

k-itemset	Tập CFI (#CFI=6)	Tập GCFI (#GCFI=13)
1	U	V, W, O, R
2	OR	VO, VR, WO, WR, WU, UO, UR
3	VOR, WOR, UOR	WUO, WUR
4	WUOR	

Bảng 2, liệt kê theo *k-itemset* của tập **CFI** và **GCFI** ở ngưỡng *minsup* = 3. Số phần tử trong các tập lần lượt |**CFI**| = 6 và |**GCFI**| = 13. Tỷ suất |CFI|/|GCFI| = 6/13×100% ≈ 46%.

3. Giải thuật đề xuất

3.1. Tập chiếu và items xuất hiện ít nhất trên cùng một giao dịch với item-hạt-nhân có thứ tự [11]

Tập chiếu của $itemi_k$ trên dữ liệu giao dịch Ɗ: $\pi(i_k)=\{t\in Ɗ \mid i_k\in t\}$ là tập các giao dịch có chứa item i_k.

$$sup(i_k) = |\pi(i_k)| \quad (1)$$

Tập chiếu của itemset X={i_1, i_2,..., i_k}, $\forall i_j\in I$ ($1 \le j \le k$): $\pi(X) = \{\pi(i_1) \cap \pi(i_2) \dots \pi(i_k)\}$.

$$sup(X) = |\pi(X)| \quad (2)$$

Để rút gọn *không gian sinh*, tác giả đưa ra Định nghĩa 5 và 6 ($\mathcal{P}_k(X)$ – *powerset của X có k item*):

Định nghĩa 5: Cho $i_k \in I$ ($i_1 \prec i_2 \prec \dots \prec i_m$) thứ tự theo *sup*, ta gọi i_k là *item-hạt-nhân*. $X_{lexcooc} \subseteq I$ gọi *xuất hiện đồng thời có thứ tự* với i_k: $X_{lexcooc}$ là tập các *item* có thứ tự theo *sup*, xuất hiện đồng thời với i_k nghĩa là $\pi(i_k) \equiv \pi(i_k \cup x_{lexcooc})$, $\forall x_{lexcooc} \in \mathcal{P}_{\ge 1}(X_{lexcooc})$, $\forall i_j \in X_{lexcooc}$, $i_k \prec i_j$. Ký hiệu, $lexcooc(i_k) = X_{lexcooc}$.

Định nghĩa 6: Cho $i_k \in I$ ($i_1 \prec i_2 \prec \dots \prec i_m$) thứ tự theo *sup*, ta gọi i_k là *item-hạt-nhân*. $X_{lexlooc} \subseteq I$ gồm *items* có thứ tự và xuất hiện cùng với i_k trong ít nhất một giao dịch, nhưng không xuất hiện đồng thời: $1 \le |\pi(i_k \cup x_{lexlooc})| < |\pi(i_k)|$, $\forall x_{lexlooc} \in \mathcal{P}_{\ge 1}(X_{lexlooc})$, $\forall i_j \in X_{lexlooc}$, $i_k \prec i_j$. Ký hiệu, $lexlooc(i_k) = X_{lexlooc}$.

Trong mục này, tác giả trình bày giải thuật sinh mảng **IndexCOOC**. Mỗi phần tử trong mảng **IndexCOOC** có 4 trường thông tin:

- **IndexCOOC[k].***item*: lưu trữ *item-hạt-nhân* i_k;

- **IndexCOOC[k].***sup*: lưu trữ *độ phổ biến của item-hạt-nhân* i_k;

- **IndexCOOC[k].***cooc*: *items* xuất hiện đồng thời cùng với *item-hạt-nhân* i_k;

- **IndexCOOC[k].***looc*: *items* xuất hiện với *item-hạt-nhân* i_k trong *ít nhất là một giao dịch*.

Giải thuật 1. Tạo dựng mảng IndexCOOC

Đầu vào*:* Tập dữ liệu *Ɗ*.

Đầu ra*:* **IndexCOOC,** Ma trận **BiM**

1. Với từng phần tử *k* của **IndexCOOC**:
2. **IndexCOOC**[k].*item*= i_k
3. **IndexCOOC**[k].*sup = 0*
4. **IndexCOOC**[k].*cooc*= 2^m - 1
5. **IndexCOOC**[k].*looc*= 0
6. Với từng giao dịch t_i thực hiện:
7. Lưu t_i vào **BiM**
8. Với từng item *k* có trong t_i thực hiện:
9. ***IndexCOOC***[k].*cooc*&=*vectorbit*(t_i)
10. ***IndexCOOC***[k].*looc* |=*vectorbit*(t_i)
11. ***IndexCOOC***[k].*sup* + +
12. Mảng **IndexCOOC** được sắp xếp tăng theo *sup*
13. Với từng phần tử *k* của **IndexCOOC**:
14. **IndexCOOC**[k].*cooc*= *lexcooc*(i_k)
15. **IndexCOOC**[k].*looc*= *lexlooc*(i_k)
16. Trả về **IndexCOOC, BiM**

Minh họa giải thuật 1: thực hiện từ *dòng 1* đến *11*.

Khởi tạo đầu tiên cho mảng **IndexCOOC**: (*cooc* và *looc được minh họa theo hexa*) số *item* từ dữ liệu *Ɗ* đã cho ở Bảng 1 là m = 8.

item	**O**	**Q**	**R**	**S**	**U**	**V**	**W**	**Z**
sup	0	0	0	0	0	0	0	0
cooc	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
looc	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Duyệt giao dịch đầu tiênt_1: {O,R, V} ở dạng bit tương ứng là **10100010**(0xA2)

item	**O**	**Q**	**R**	**S**	**U**	**V**	**W**	**Z**
sup	1	0	1	0	1	1	0	0
cooc	**0xA2**	0xFF	**0xA2**	0xFF	**0xA2**	**0xA2**	0xFF	0xFF
looc	**0xA2**	0x00	**0xA2**	0x00	**0xA2**	**0xA2**	0x00	0x00

Tương tự, duyệt giao dịch cuối cùng t_{10}: {O, Q, R, U, W} ở dạng bit tương ứng là **11101010**(0xEA)

item	**O**	**Q**	**R**	**S**	**U**	**V**	**W**	**Z**
sup	8	2	8	2	7	3	5	1
cooc	**0xA0**	0xE8	**0xA0**	0xB0	**0x08**	**0xA4**	**0xA2**	0x09
looc	**0xFE**	0xEA	**0xFE**	0xB6	**0xEF**	**0xBE**	**0xFE**	0x09

Dòng *12*, sắp xếp **IndexCOOC** tăng dần theo *sup* của từng item, ta có kết quả:

item	**Z**	**Q**	**S**	**V**	**W**	**U**	**O**	**R**
sup	1	2	2	3	5	7	8	8
cooc	U	O,R,U	O, R	O, R	O, R	∅	R	O
looc	∅	W	V, W	S, U, W	Q,S,U,V	O,Q,R,V,W,Z	Q,S,U,V,W	Q,S,U,V,W

Từ *dòng 13 đến 15* – cho kết quả rút gọn ở Bảng 3.

Bảng 3. *Mảng IndexCOOC có thứ tự tăng dần theo độ phổ biến của item, đồng thời cooc và looc cũng có thứ tự*

item	**Z**	**Q**	**S**	**V**	**W**	**U**	**O**	**R**
sup	1	2	2	3	5	7	8	8
cooc	U	O, R, U	O, R	O, R	O, R	∅	R	∅
looc	∅	W	V,W	W, U	U	O, R	∅	∅

3.2. Giải thuật sinh cây nLOOCTree

Từ **IndexCOOC** xây dựng các cây chứa các mẫu *xuất hiện với item-hạt-nhân trong ít nhất* một giao dịch. Mỗi cây có nút gốc là *item-hạt-nhân* và các nút con là *items xuất hiện với item-hạt-nhân* trong ít nhất trong một giao dịch. Mỗi nút có 2 trường thông tin:

- **nLOOCTree[k].***item*: lưu trữ item xuất hiện ít nhất với *item-hạt-nhân* (nút gốc);

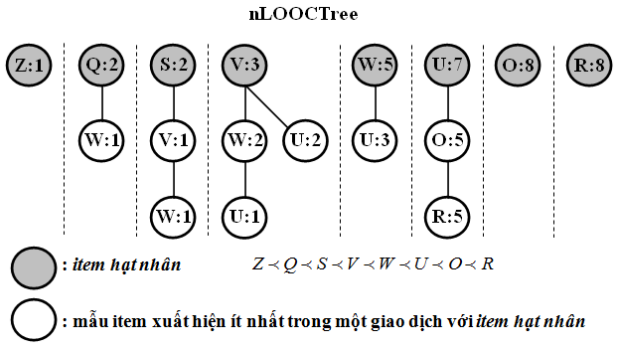
- **nLOOCTree[k].***sup*: lưu trữ độ phổ biến của item xuất hiện với *item-hạt-nhân*.

Giải thuật 2: Tạo sinh nLOOCTree

Đầu vào:Ɗ, **IndexCOOC**

Đầu ra*:* các **nLOOCTree**

1. Với từng phần tử *k* của **IndexCOOC**:
2. nLOOCTree[k].*item* = ***IndexCOOC***[k].*item*
3. nLOOCTree[k].*sup* =***IndexCOOC***[k].*sup*
4. Với từng $i_k \in$ giao dịch t_ℓ:
5. Với từng $i_j \in$**IndexCOOC**[k].*looc*:
6. Nếu $i_j \notin$ nút con của gốc **nLOOCTree**[k] thì
7. Thêm vào gốc **nLOOCTree**[k] nút con i_j
8. Ngược lại
9. Thêm vào gốc **nLOOCTree**[k] nút con i_j
10. Cập nhật *sup* của i_j của gốc **nLOOCTree**[k]
11. Trả về các **nLOOCTree**



Hình 1. *Các nLOOCTree theo IndexCOOC ở Bảng 3*

Mỗi **nLOOCTree** có các đặc trưng sau đây:

- Chiều cao tương ứng của cây *bé hơn hoặc bằng* số lượng các *item xuất hiện với item-hạt-nhân trong ít nhất* một giao dịch (*items có thứ tự theo sup*).

- Một đường đi đơn (*single-path*) là *itemset* có thứ tự từ nút gốc đến nút lá và *sup* của *itemset* chính là *sup* của nút lá ($i_k \rightarrow i_{k+1} \rightarrow \ldots \rightarrow i_\ell$).

- Phân đoạn của đường đi đơn (*sub-single-path*) là từ nút gốc đi đến nút con tùy ý trong một đường đi đơn là *itemset* thứ tự và *sup* của *itemset* đó là *sup* của nút con nằm ở cuối của phân đoạn.

3.3. Giải thuật tuần tự khai thác tập sinh SEQ-GCFI

Giải thuật **SEQ-GCFI** (**SEQ**uential-**G**enerators **C**losed **F**requent **I**temsets): Khai thác tuần tự *tập sinh* dựa trên cây **nLOOCTree** chứa *items cùng xuất hiện với item-hạt-nhân trong ít nhất là một giao dịch*.

Trong mục này, tác giả trình bày các bổ đề dùng để loại bỏ những *item-hạt-nhân* không có *itemset* sinh của *itemset* phổ biến đóng.

Bổ đề 1: $\forall\, i_k \prec i_j$, $i_j \in lexlooc(i_k)$:$sup(i_k \cup i_j) < sup(i_k)$.

Chứng minh: $sup(i_k \cup i_j) < sup(i_k)$, theo (1) và (2) $\pi(i_k \cup i_j) = \pi(i_k) \cap \pi(i_j) \subset \pi(i_k)$■.

Bổ đề 2: $\forall i_k \in$ I, $sup(i_k) = minsup$, nếu $lexlooc(i_k) = X_{lexlooc}$ và $sup(i_k \cup x_{sub}) < minsup$ thì $\{i_k \cup x_{sub}\} \notin$**GCFI**, $\forall\, x_{sub} \in \mathcal{P}_{\geq 1}(X_{lexlooc})$.

Chứng minh: Dựa vào bổ đề 1, $sup(i_k \cup x_{sub}) < sup(i_k) = minsup$, nên $\{i_k \cup x_{sub}\} \notin$**GCFI**, $\forall x_{sub} \in \mathcal{P}_{\geq 1}(X_{lexlooc})$■.

Bổ đề 3: $\forall i_k \in$ I, $sup(i_k) \geq minsup$ và $lexcooc(i_k) = X_{lexcooc}$ thì $\{i_k \cup x_{sub}\} \in$**GCFI**, $\forall\, x_{sub} \in \mathcal{P}_{<\lambda}(X_{lexcooc})$, $\lambda = |X_{lexcooc}|$.

Chứng minh: Dựa vào bổ đề 1, $sup(Y) < sup(\{i_k \cup \mathcal{P}_{=\lambda}(X_{lexcooc})\})$, $\forall\, Y \supset \{i_k \cup \mathcal{P}_{=\lambda}(X_{lexcooc})\}$. Khi đó, $\{i_k \cup \mathcal{P}_{=\lambda}(X_{lexcooc})\} \in$**CFI** và (2) thì $sup(\{i_k \cup \mathcal{P}_{=\lambda}(X_{lexcooc})\}) = sup(\{i_k \cup x_{sub}\})$, $\forall\, x_{sub} \in \mathcal{P}_{<\lambda}(X_{lexcooc})$■.

Tính chất 1: $\forall i_k \in I$, nếu $sup(i_k) \geq minsup$ và $lexcooc(i_k) = \{\varnothing\}$ thì $i_k \notin$**GCFI**.

Tính chất 2: (*tập sinh từ các item đồng xuất hiện hoàn toàn*) $\forall\, i_{k-1} \prec i_k \in I$, nếu $sup(i_{k-1}) = sup(i_k)$ và $i_k \in lexcooc(i_{k-1})$ thì *itemsets sinh* từ *item hạt nhân* i_{k-1}, $\forall \{i_{k-1} \cup X\} \in$**GCFI** được sinh thêm bằng cách thay bởi *item hạt nhân* i_k, $\forall \{i_k \cup X\} \in$**GCFI**.

Giải thuật khai thác *tập sinh của tập đóng phổ biến* **GCFI** từ cây **nLOOCTree**($i_k \equiv$**IndexCOOC**[k]):

Giải thuật 3: **Sinh tập GCFI**

Đầu vào: **IndexCOOC**, *minsup*

Đầu ra: Tập sinh **GCFI**

1. Với mỗi $sup(i_k) \geq minsup$
2. Nếu ($i_k.cooc \neq \{\varnothing\}$)*// tính chất 1*
3. **GCFI**[k]=$\cup${$i_k \cup \mathcal{P}_{<\lambda}(i_k.cooc)$}
4. Nếu ($sup(i_k) > minsup$)$\wedge$($i_k.looc \neq \{\varnothing\}$)*//bổ đề 2*
5. **nLOOCTree**(i_k)
6. ***SSP***← GenPath(i_k)*//sinh sub-sp*
7. Với mỗi $ssp_j \in$***SSP***
8. Nếu ($sup(ssp_j) \geq minsup$) $\wedge$ ($sup(ssp_j \backslash \{i_\ell\}) \geq minsup$)
9. **GCFI** [*k*] = $\cup${($ssp_j \backslash \{i_\ell\}$)}
10. **GCFI** [*k*] = $\cup${($ssp_j \backslash \{i_\ell\}$)$\cup \mathcal{P}_{<\lambda}(i_k.cooc)$}*//bổ đề 3*
11. Nếu ($sup(i_{k-1}) = sup(i_k) \wedge (i_{k-1} \in i_k.cooc)$
12. **GCFI**[k]=Replace(**GCFI**[k-1], i_k)*//tính chất 2*
13. Trả về tập sinh **GCFI**

Thí dụ1: Cho Ɗ được mô tả ở Bảng 1 với *minsup* = 3. Kết thúc giải thuật 1, cho ta mảng **IndexCOOC** được trình bày ở Bảng 3.

Xét dòng 1-2: tập item {V, W, U, O, R} là *item sinh* theo **tính chất 1**;

Xây dựng các cây **nLOOCTree** cho các *item* cần khai phá: V, W, U, O và R;

Xét *item*V, *lexcooc*(V) = {O, R} sinh tập **GCFI**$_{[V]}$ = {(**V**, 3), (**V**O, 3), (**V**R, 3)} (*dòng 3*) và *sup*(V) = *minsup*, nên không sinh **GCFI** từ **nLOOCTree**(V) (*dòng 4*).

Xét *item*W, *lexcooc*(W) = {O, R} sinh tập **GCFI**$_{[W]}$ = {(**W**, 5), (**W**O, 5), (**W**R, 5)} (*dòng 3*) xem cây **nLOOCTree**(W): có đường đi đơn {**W**→U} với

sup(**W**U) = 3 ≥*minsup*. Vì vậy, **GCFI[W]** = ∪{(**W**U, 3), (**W**UO, 3), (**W**UR, 3)} (*dòng 4 đến 10*).

Xét *item*U, *lexcooc*(U) = {∅} nên **GCFI[U]** = {∅}, xem cây **nLOOCTree**(U): có một đường đi đơn {**U**→O→R} với *sup*(**U**OR) = 5 ≥*minsup*. Vì vậy, **GCFI[U]** = {(**U**O, 5), (**U**R, 5)}.

Xét *item*O, *lexcooc*(O) = {R} sinh tập **GCFI[O]** = {(**O**, 8)} (*dòng 3*) và *lexlooc*(O) = {∅}, nên không sinh **GCFI** từ **nLOOCTree**(O) (*dòng 4*).

Xét *item*R, đồng xuất hiện hoàn toàn với *item*O: **GCFI[R]** = {(**R**, 8)} (*dòng 11*).

Tập sinh **GCFI** từ dữ liệu $\mathcal{D}$ ở Bảng 1 với *minsup* = 3:

Bảng 4. *Tập sinh* ***GCFI*** *trên* $\mathcal{D}$ *với minsup = 3*

item	Tập sinh GCFI (#GCFI = 13)		
V	(V, 3)	(VO, 3)	(VR, 3)
W	(W, 5)	(WO, 5)	(WR, 5)
	(WU, 3)	(WUO, 3)	(WUR, 3)
U	(UO, 5)	(UR, 5)	
O	(O, 8)		
R	(R, 8)		

4. Giải thuật song song MCP-GCFI

Ngày nay, nhiều máy tính cá nhân và máy trạm có *trên hai nhân* cho phép nhiều luồng xử lý được thực hiện đồng thời - điều này làm cho máy tính có được tốc độ xử lý nhanh và khả năng đa nhiệm tốt hơn. Để tận dụng hiệu năng của BXLĐN, cần phân phối xử lý đồng thời trên nhiều nhân cho nhiều pha/ bài toán để tiết kiệm thời gian và nâng cao hiệu năng.

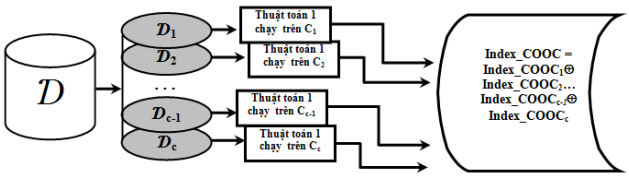
Tác giả xây dựng giải thuật song song **MCP-GCFI** (***M**ulti **C**ore **P**rocessor – **G**enerators of **C**losed **F**requent **I**temsets*) khai thác *tập sinh* trên BXLĐN dựa trên giải thuật tuần tự **SEQ-GCFI**.

Giải thuật tuần tự **SEQ-GCFI**, gồm *2 pha* chính:

Pha 1: Xây dựng mảng **IndexCOOC**;

Pha 2: Giải thuật 3 khai thác *tập sinh* từ mảng **IndexCOOC**.

Bước thứ nhất, song song hóa **Pha 1** theo sơ đồ Hình 2.



Hình 2. *Sơ đồ song song hóa Pha 1*

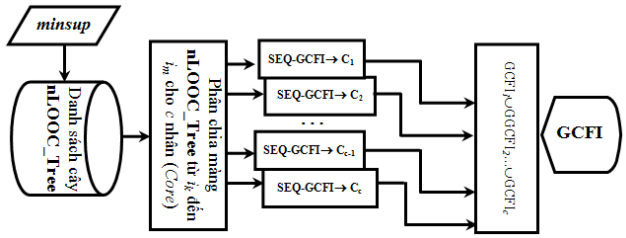
Hình 2, phân chia dữ liệu $\mathcal{D}$ thành *c* tập dữ liệu con $\mathcal{D}_1$, $\mathcal{D}_2$,…, $\mathcal{D}_{c-1}$, $\mathcal{D}_c$ ứng với từng nhân C_i thực hiện giải thuật 1 với đầu vào là dữ liệu $\mathcal{D}_i$ và đầu ra là mảng $IndexCOOC_i$ tương ứng. Để tính mảng **IndexCOOC** cho $\mathcal{D}$, thực hiện phép tính:

$$\textbf{IndexCOOC}=IndexCOOC_1 \oplus IndexCOOC_2 \oplus \ldots \quad (3)$$

Sau đó, mảng **IndexCOOC** được xếp tăng theo *sup* và chuẩn hóa theo dòng 13, 14 và 15 của giải thuật 1.

Thí dụ 2: Giả sử $\mathcal{D}$ được chia thành 2 tập - $\mathcal{D}_1$ gồm có {t1, t2, t3, t4, t5} và $\mathcal{D}_2$ là {t6, t7, t8, t9, t10}.

$\mathcal{D}_1$ chạy giải thuật 1 trên nhân C_1 và trả về $IndexCOOC_1$:



Hình 3. *Sơ đồ song song hóa Pha 2*

Hình 3, phân chia mảng **IndexCOOC** từ i_1 đến i_m thành *c* phần ứng với từng nhân C_j thực hiện giải thuật **SEQ-GCFI** với đầu vào là mảng **IndexCOOC** từ phần tử thứ *[(j-1)*(m div c)+1]* đến phần tử thứ *[j*(m div c)]* và đầu ra là tập sinh **GCFI**$_j$ tương ứng. Tập sinh cho tập dữ liệu giao dịch $\mathcal{D}$, ta thực hiện theo phương trình:

$$\textbf{GCFI}_{\mathcal{D}} = GCFI_1 \cup GCFI_2 \ldots \cup GCFI_c \quad (4)$$

Thí dụ 3: Cho $\mathcal{D}$ được mô tả ở Bảng 1, *minsup* = 3. Kết thúc song song hóa **Pha 1**, có như Bảng 3.

Song song hóa **Pha 2**: Nhân C_1 chạy giải thuật **SEQ-GCFI** từ *item*V đến U khai thác *itemset sinh* $GCFI_1$:

GCFI$_{1[V]}$	(V, 3)	(VO, 3)	(VR, 3)
GCFI$_{1[W]}$	(W, 5)	(WO, 5)	(WR, 5)
	(WU, 3)	(WUO, 3)	(WUR, 3)
GCFI$_{1[U]}$	(UO, 5)	(UR, 5)	

Nhân C_2 chạy giải thuật **SEQ-GCFI** từ *item*O đến R khai thác *itemset sinh* $GCFI_2$:

GCFI$_{2[O]}$	(O, 8)		
GCFI$_{2[R]}$	(R, 8)		

Tập *sinh* **GCFI** trên $\mathcal{D}$ với *minsup* = 3 được tính **GCFI**$_{\mathcal{D}}$ = $GCFI_1 \cup GCFI_2$, ta có kết quả như Bảng 4.

5. Kết quả thực nghiệm

Giải thuật đề xuất được thực nghiệm cài đặt trên máy tính cấu hình: CPU Core Duo 2.0 GHz, bộ nhớ 4GB; ngôn ngữ lập trình C#(VS 2010).

Giải thuật được thực nghiệm trên 2 nhóm dữ liệu:

- Dữ liệu thu thập thực tế với mật độ dày: 2 tập **Chess** và **Mushroom** từ kho lưu trữ UCI.

- Dữ liệu chạy giả lập với mật độ thưa: 2 tập dữ liệu giả lập **T10I4D100K** và **T40I10D100K** từ trung tâm Almaden của IBM.

Bảng 5. *Dữ liệu thực nghiệm*

$\mathcal{D}$	Số item	Số item nhỏ-nhất	Số item lớn-nhất	Số item trung-bình
Chess	75	37	37	37
Mushroom	119	23	23	23
T10I4D100K	870	1	29	10
T40I10D100K	942	4	77	40

Trong nghiên cứu, tác giả đề xuất giải thuật *khai thác song song tập sinh*. Đây là nghiên cứu đề xuất theo hướng

tiếp cận song song, nên chưa có giải thuật có cùng hướng tiếp cận nhằm đánh giá hiệu năng giải thuật. Vì vậy, tác giả đề xuất đánh giá hiệu năng giải thuật song song như sau: So sánh giải thuật tuần tự **SEQ-GCFI** với giải thuật **GrGrowth** [6] và giải thuật **Zart** [8]. Sau đó, so sánh hiệu suất của giải thuật song song **MCP-GCFI** với giải thuật tuần tự **SEQ-GCFI**. Trong thực nghiệm, tác giả tiến hành đánh giá theo từng ngưỡng tối thiểu *minsup* và tất cả 4 giải thuật trên đều cho kết quả như nhau.

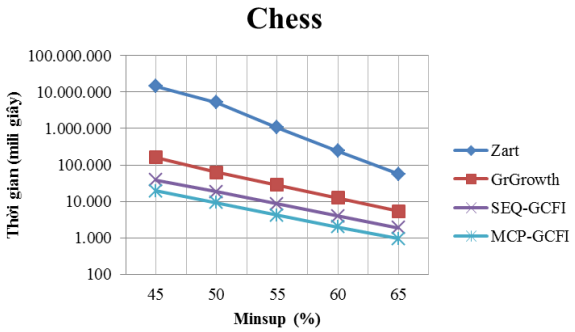
Hiệu suất chạy giải thuật song song trên BXLĐN:

$$HS = 1 - \left(T_M - \frac{T_S}{c}\right) \Big/ \left(\frac{T_S}{c}\right) \quad (5)$$

Trong đó:

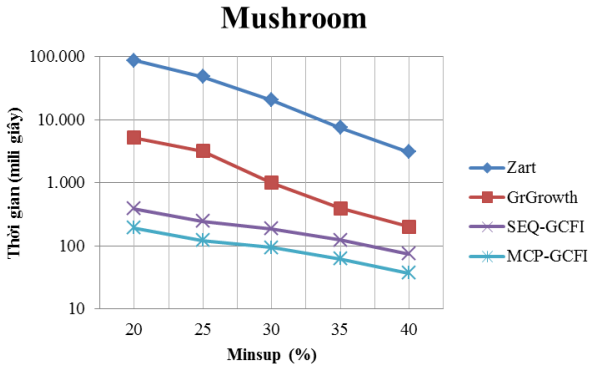
- T_S: thời gian thực hiện tuần tự;
- T_M: thời gian thực hiện song song trên BXLĐN;
- c: số lượng nhân của CPU (số *core*).

Phương trình (5): Đánh giá hiệu suất giải thuật song song **MCP-GCFI** so với giải thuật tuần tự **SEQ-GCFI**.



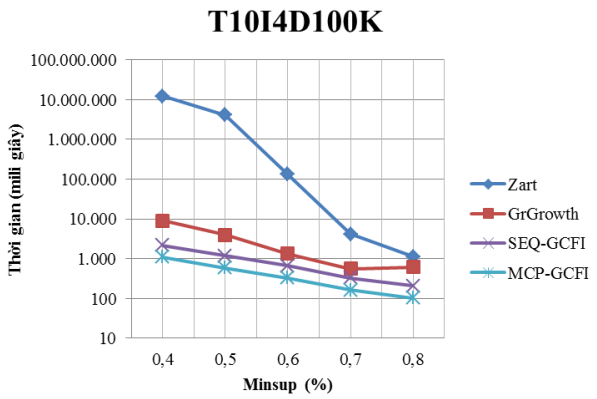
Hình 4. *Thời gian thực hiện khai thác **GCFI** trên **Chess***

Hình 4 - kết quả được thực nghiệm từ tập dữ liệu thực **Chess** mật độ dày đặc (49,3%) cho thấy, giải thuật tuần tự **SEQ-GCFI** nhanh hơn giải thuật **GrGrowth**, **Zart** và giải thuật **MCP-GCFI** có thời gian chạy nhanh hơn giải thuật **SEQ-GCFI.** Hiệu suất trung bình của giải thuật **MCP-GCFI** là 76,1% và độ lệch chuẩn 1,8%.



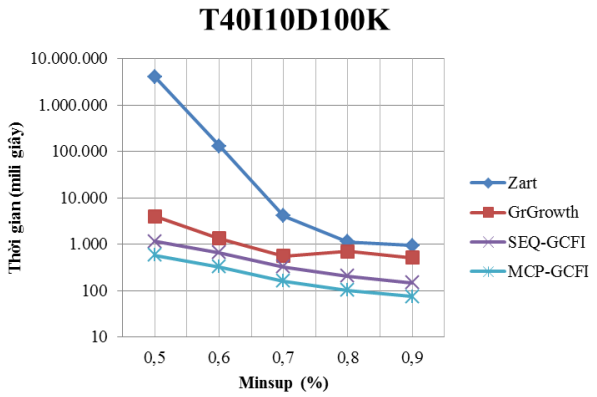
Hình 5. *Thời gian thực hiện khai thác **GCFI** trên **Mushroom***

Hình 5 - kết quả được thực nghiệm từ tập dữ liệu thực **Mushroom** mật độ dày (19,3%) cho thấy, giải thuật tuần tự **SEQ-GCFI** nhanh hơn giải thuật **GrGrowth, Zart** và giải thuật **MCP-GCFI** có thời gian chạy nhanh hơn giải thuật **SEQ-GCFI.** Hiệu suất trung bình của giải thuật **MCP-GCFI** là 74,3% và độ lệch 1,5%.



Hình 6. *Thời gian thực hiện khai thác **GCFI** trên **T10I4D100K***

Hình 6 - kết quả được thực nghiệm từ tập dữ liệu giả lập **T10I4D100K** mật độ rất thưa (1,1%) cho thấy, giải thuật tuần tự **SEQ-GCFI** nhanh hơn giải thuật **GrGrowth, Zart** và giải thuật **MCP-GCFI** có thời gian chạy nhanh hơn giải thuật tuần tự **SEQ-GCFI**. Hiệu suất trung bình của **MCP-GCFI** là 80,1% và độ lệch 1,4%.



Hình 7. *Thời gian thực hiện khai thác **GCFI** trên **T40I10D100K***

Hình 7 - kết quả được thực nghiệm từ tập dữ liệu giả lập **T40I10D100K** mật độ thưa (4,2%) cho thấy, giải thuật tuần tự **SEQ-GCFI** nhanh hơn giải thuật **GrGrowth, Zart** và giải thuật chạy trên BXLĐN là **MCP-GCFI** có thời gian chạy nhanh hơn giải thuật tuần tự **SEQ-GCFI**. Hiệu suất trung bình của giải thuật **MCP-GCFI** là 81,5% và độ lệch chuẩn 1,6%.

Qua thực nghiệm cho thấy, giải thuật khai thác song song tập sinh **MCP-GCFI** trên BXLĐN tốt hơn rất nhiều so với giải thuật **GrGrowth, Zart**. Giải thuật **MCP-GCFI** cần được thực nghiệm mở rộng trên các dữ liệu giao dịch có kích cỡ lớn và so sánh thêm với các giải thuật chạy trên hệ thống phân tán như Hadoop, Spark.

6. Kết luận

Bài viết đã trình bày giải thuật tuần tự **SEQ-GCFI** khai thác *tập sinh của tập đóng phổ biến* gồm ba bước: *Đầu tiên* là phát sinh nhanh cấu trúc mảng **IndexCOOC** có chứa *items* xuất hiện đồng thời với *item-hạt-nhân* và *items* xuất hiện ít nhất với *item-hạt-nhân* trong một giao dịch; *Bước 2*: Xây dựng **nLOOCTree** dựa vào mảng **IndexCOOC**; *Bước 3*: Khai thác nhanh *tập sinh của tập đóng phổ biến* dựa trên cây **nLOOCTree**. Từ giải thuật tuần tự **SEQ-**

GCFI, tác giả mở rộng và song song hóa thực hiện trên BXLĐN gọi là giải thuật **MCP-GCFI**. Hiệu suất trung bình khi song song hóa trên 4 tập dữ liệu thực nghiệm là 78% và độ lệch chuẩn 3,3%.

Trong các nghiên cứu tiếp theo, tác giả hướng đến việc mở rộng giải thuật **MCP-GCFI** để khai thác nhanh *tập sinh* trên hệ thống điện thoại thông minh đa lõi có tài nguyên hạn chế, cũng như thực nghiệm mở rộng trên hệ thống phân tán phổ biến hiện nay như Hadoop, Spark.

TÀI LIỆU THAM KHẢO

[1] R. Agrawal, T. Imilienski andA. Swami, *Mining association rules between sets of large database*s, Proc. of the ACM SIGMOD Int Conf on Management of Data., 1993, pp. 207-216.

[2] P. Huan and L. Bac, *A Novel Algorithm for Frequent Itemsets Mining in Transactional Databases*, In: Trends and Applications in Knowledge Discovery and Data Mining. PAKDD 2018. LNCS, 11154, Springer Cham, 2018, pp. 243–255.

[3] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal, *Mining Minimal Non-Redundant Association Rules using Closed Frequent Itemssets*, In: 1st International Conference on Computational Logic, 2000, pp.972 – 986.

[4] M. J. Zaki, "Mining Non-Redundant Association Rules", *Data Mining and Knowledge Discovery*, 9, Springer,2004, pp. 223–248.

[5] P. Huan, *NOV-CFI: A Novel Algorithm for Closed Frequent Itemsets Mining in Transactional Databases*, ICNCC 2018, ACM, NY, USA, 2018, pp. 58-63.

[6] G. Liu, J. Li and L. Wong, "A new concise representation of frequent itemsets using generators and a positive border", *Knowl. Inf. Syst*, 17(1), Springer, 2008, pp. 35-56.

[7] T. Hamrouni, "Key roles of closed sets and minimal generators in concise representations of frequent patterns", *Intell. Data Anal*, 16(4), 2012, pp.581-631.

[8] L. Szathmary, A. Napoliand S. O. Kuznetsov, *ZART: AMultifunctional Itemset Mining Algorithm*, The 6th Intl Conf on Concept Lattices and Their Applications, 2008, pp. 47–58.

[9] A. Soulet and F. Rioult, *Efficiently Depth-First Minimal Pattern Mining*, In: Tseng V.S., Ho T.B., Zhou ZH., Chen A.L.P., Kao HY. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2014. LNCS. 8443, Springer Cham, 2014, pp. 28-39.

[10] A. Soulet and F. Rioult, *Exact and Approximate Minimal Pattern Mining*, In: Guillet F., Pinaud B., Venturini G. (eds) Advances in Knowledge Discovery and Management. SCI. 665, Springer Cham,2017, pp. 61-81.

[11] Phan Thành Huấn, Lê Hoài bắc, Thuật toán hiệu quả khai thác tập hiếm tối thiểu, FAIR – Nghiên cứu cơ bản và ứng dụng, 2018, pp. 497-505.

(BBT nhận bài: 21/10/2019, hoàn tất thủ tục phản biện: 08/5/2020)