# QUICKGEN: KHAI THÁC NHANH LUẬT KẾT HỢP TRÊN DỮ LIỆU GIAO DỊCH NHỊ PHÂN

## QUICKGEN: QUICK ASSOCIATION RULES MINING IN TRANSACTIONAL DATABASES

**Phan Thành Huấn***

*Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh, Việt Nam*[1]

*Tác giả liên hệ / Corresponding author: huanphan@hcmussh.edu.vn

(Nhận bài / Received: 26/9/2023; Sửa bài / Revised: 06/01/2024; Chấp nhận đăng / Accepted: 16/01/2024)

**Tóm tắt -** Khai thác luật kết hợp (LKH) trong khai phá dữ liệu có vai trò quan trọng trong việc tìm ra các kết hợp hoặc luật xuất hiện đồng thời trong dữ liệu. Bài toán được chia thành hai giai đoạn: thứ nhất, sinh các kết hợp từ dữ liệu thỏa ngưỡng phổ biến *minsup*; thứ hai, sinh LKH từ tập chứa các kết hợp phổ biến được tìm ở trên và thỏa ngưỡng tin cậy *minconf*. Phần lớn các nghiên cứu tập trung xác định các kết hợp phổ biến ở giai đoạn thứ nhất. Ngược lại, giai đoạn sinh LKH ít được quan tâm nghiên cứu. Trong nghiên cứu, tác giả trình bày giải pháp rút gọn tập ứng viên dựa vào khái niệm lớp tương đương – giải pháp được đặt tên QuickGen. Phần thực nghiệm, tác giả xây dựng hai kịch bản: (1) so sánh tính hiệu quả của giải pháp trên giai đoạn sinh luật; (2) đánh giá hiệu quả trên toàn bộ quá trình khai thác LKH – QuickGen được dùng ở cả hai giai đoạn. Kết quả cho thấy giải pháp đề xuất mang lại hiệu suất vượt trội.

**Từ khóa -** Lớp tương đương; luật kết hợp; thuật toán QuickGen

**Abstract -** Association rules mining in data mining shows an important role in discovering combinations or rules that appear together in data. The problem is divided into two stages: first, generating combinations from data that satisfy the minimum support threshold; second, generating association rules from the set of frequent combinations found above and satisfying the minimum confidence threshold. Most research focuses on determining frequent combinations in the first stage. Conversely, the generation of association rules receives less attention in research. In this study, the author presents a solution to reduce the candidate set based on the concept of equivalence classes - a solution named QuickGen. In the experimental part, the author constructs two scenarios: (1) comparing the effectiveness of the solution in the rule generation stage; (2) evaluating the overall effectiveness of the association rule mining process - QuickGen is used in both stages. The results show that the proposed solution provides superior performance.

**Key words -** Equivalence classes; Association rules; QuickGen algorithm

### 1. Đặt vấn đề

Năm 1993, Agrawal và các đồng nghiệp đã đề xuất bài toán khai thác luật kết hợp (LKH) trên dữ liệu giao dịch (DLGD) nhị phân [1]. Mục tiêu của bài toán là tìm các LKH với ngưỡng phổ biến tối thiểu (*minsup*) và ngưỡng tin cậy tối thiểu (*minconf*). Bài toán được chia thành hai giai đoạn [1-9]:

- Giai đoạn khai thác tập phổ biến (**F**requent **I**temset **M**ining – FIM): tập hợp các tập phổ biến được tạo ra bằng cách áp dụng thuật toán Apriori. Thuật toán này sử dụng tính chất kết hợp và sàng lọc để hiệu quả tạo ra các tập phổ biến đáp ứng hoặc vượt qua ngưỡng *minsup*. Kết quả của giai đoạn này là một tập hợp các tập phổ biến;

- Giai đoạn khai thác LKH (**A**ssociation **R**ule **M**ining - ARM): Sau khi có được các tập phổ biến, giai đoạn này tập trung vào tìm ra các LKH từ các tập phổ biến đó. Một LKH được biểu diễn dưới dạng "X→Y", trong đó X và Y là các tập con của một tập phổ biến. Một LKH được coi là ý nghĩa nếu nó đáp ứng các tiêu chí như độ tin cậy (confidence) và độ phổ biến (support).

Phương pháp của Agrawal cung cấp một cách tiếp cận hệ thống để khám phá các mối liên hệ ý nghĩa giữa các thuộc tính trong DLGDnhị phân. Phương pháp này đã có ảnh hưởng đáng kể trong lĩnh vực khai thác LKH và đã mở ra con đường cho nghiên cứu và phát triển tiếp theo.

Hai hướng tiếp cận chính trong khai thác LKH:

- *Khai thác đầy đủ LKH*: Đây là hướng tiếp cận tập trung cải tiến hiệu suất của giai đoạn sinh tập phổ biến, chẳng hạn như các thuật toán tựa Apriori [1], Eclat [2], FP-Growth [3], NOV-FI [4]…

- *Khai thác không đầy đủ LKH*: Đây là hướng tiếp cận tìm các tập phổ biến có số lượng nhỏ – tập phổ biến đóng [5, 8]; tập phổ biến tối đại [9]. Mặt khác, hướng tiếp cận này còn đề xuất rút gọn tập LKH bằng cách bổ sung các ràng buộc trong khai thác LKH – giảm số lượng LKH dư thừa [6-7].

Ngoài ra, nhóm tác giả ở công trình [10] đã phân tích chi tiết không gian sinh giai đoạn khai thác tập phổ biến là $(2^m – 1)$ với m là số lượng thuộc tính trên DLGD; tương ứng, không gian sinh LKH là $(3^m – 2^{m+1}+1)$. Công trình đã phân tích và cho thấy thực sự giai đoạn khai thác LKH là giai đoạn có không gian sinh lớn hơn nhiều so với giai đoạn sinh tập phổ biến khi số lượng thuộc tính lớn. Tuy nhiên, giai đoạn khai thác LKH ít được quan tâm nghiên cứu cải tiến. Chính vì lẽ đó, tác giả đã nghiên cứu và tìm kiếm giải pháp tối ưu quá trình khai thác LKH.

Trong bài viết, tác giả tập trung trình bày đề xuất giải thuật sinh nhanh và đầy đủ LKH dựa vào khái niệm lớp tương đương được giới thiệu bởi Zaki và đồng sự [2].

[1] Vietnam National University Ho Chi Minh City - University of Science, Hochiminh, Vietnam (Huan Phan)

## 2. Các vấn đề liên quan

### *2.1. Khai thác tập phổ biến*

Cho I = $\{i_1, i_2,..., i_m\}$ là tập chứa *m* thuộc tính, mỗi thuộc tính gọi là *item*. Với $X \subseteq I$, $X = \{i_1, i_2,..., i_k\}$, $\forall i_j \in I$ $(1 \le j \le k)$ gọi là *itemset*, *itemset* có *k item* gọi là *k-itemset*. DLGDgồm *n* giao dịch gọi là tập các giao dịch T = $\{t_1, t_2,..., t_n\}$, mỗi giao dịch $t_k = \{i_{k1}, i_{k2},..., i_{km}\}$, $i_{kj} \in I$ $(1 \le k_j \le m)$.

**Định nghĩa 1 [1]**: Cho X, Y ∈ I với X ∩ Y = ∅, LKH là một mệnh đề kéo theo có dạng X → Y, thỏa hai ngưỡng độ đo cho trước (*minsup* – độ phổ biến tối thiểu; *minconf* – độ tin cậy tối thiểu), trong đó X gọi là "tiền đề" và Y là "kết luận".

**Định nghĩa 2 [1]:** Độ phổ biến (*support*) dùng để đánh giá mức độ xuất hiện của một *itemset* trong tập dữ liệu. Độ phổ biến được đo bằng tỷ lệ giữa số lần xuất hiện của *itemset* trong tập dữ liệu và tổng số giao dịch.

$$sup(X) = \frac{\left|\{t \in T \mid X \subseteq t\}\right|}{n}$$

**Định nghĩa 3 [1]:** Cho $X \subseteq I$, *X* là *itemset* phổ biến nếu $sup(X) \ge minsup$, *minsup* là ngưỡng phổ biến tối thiểu (được người dùng cho trước). Tập chứa các *itemset* phổ biến được ký hiệu là **FI**.

**Định nghĩa 4 [1]**: Độ phổ biến của LKH X→Y, ký hiệu *sup*(X ∪ Y) – độ phổ biến của {X→Y}.

$$sup(X \to Y) = sup(X \cup Y)$$

**Định nghĩa 5 [1]**: Độ tin cậy (confidence) của LKH X→Y, ký hiệu conf(X→Y) - tỷ lệ giữa số giao dịch chứa {X ∪ Y} và số giao dịch chứa X.

$$conf(X \to Y) = \frac{sup(X \cup Y)}{sup(X)}$$

Cho DLGDT sử dụng cho các minh họa trong khai thác LKH như ở Bảng 1, có 8 thuộc tính *I* = {*A, B, C, D, E, F, G, H*} và 10 dòng giao dich.

Xét DLGD T với *minsup* = 0,50 – sinh tập phổ biến FI gồm 11 *itemset* {(G; 0,50), (E; 0,70), (A; 0,80), (C; 0,80), (GA; 0,50), (GC; 0,50), (EA; 0,50), (EC; 0,50), (AC; 0,80), (GAC; 0,50), (EAC; 0,50)}. Theo công trình [10], tập FI có *itemset* chiều dài lớn nhất là GAC và EAC – chiều dài bằng 3, nghĩa là không gian sinh LKH tối thiểu tương ứng $3^3 - 2^{3+1} + 1 = 12$ LKH thỏa *minsup*. Tuy nhiên, có 2 *itemset* phổ biến GAC, EAC, do có chung AC nên số lượng LKH là 2×12 – 2 = 22 (2 LKH sinh từ AC).

***Bảng 1.*** *Dữ liệu giao dịch T*

| TID | Items |
|---|---|
| *t1* | A, C, E, F |
| *t2* | A, C, G |
| *t3* | E, H |
| *t4* | A, C, D, G |
| *t5* | A, C, E, G |
| *t6* | E, H |
| *t7* | A, B, C, E, F |
| *t8* | A, C, D |
| *t9* | A, C, E, G |
| *t10* | A, C, E, G |

***Bảng 2.*** *LKH trên T thỏa minsup = 0,50 và minconf =1*

| k-*itemset* | Luật kết hợp (LKH; sup; conf) |
|---|---|
| 2 | (G→A; 0,50; 1), (G→C; 0,50; 1), (A→C; 0,80; 1), (C→A; 0,80; 1) |
| 3 | (GC→A; 0,50; 1), (GA→C; 0,50; 1), (G→AC; 0,50; 1), (EC→A; 0,50; 1), (EA→C; 0,50; 1) |

Bảng 2, liệt kê 9 LKH thỏa ngưỡng *minconf* = 1 gồm có 4 LKH được tạo thành từ 2-*itemset* phổ biến và 5 LKH được tạo thành từ 3-*itemset* phổ biến.

### *2.2. Thuật toán khai thác LKH*

#### *2.2.1. Thuật toán sinh tập phổ biến*

Thuật toán Apriori [1] là thuật toán quan trọng và được sử dụng rộng rãi trong khai thác luật kết hợp, được đề xuất bởi Agrawal và cộng sự; đã có nhiều trích dẫn và cải tiến từ đó. Ngoài ra, thuật toán cũng có thể được áp dụng trên nhiều định dạng dữ liệu khác nhau.

Một số ký hiệu trong thuật toán Apriori:

- $L_k$ : tập thành viên *k-itemset* thỏa *minsup*, mỗi thành viên có 2 trường thông tin là *itemset* và *sup*;
- $C_k$ : tập ứng viên chứa *k-itemset* tiềm năng, mỗi ứng viên có 2 trường thông tin là *itemset* và *sup*;

**Mã giả thuật toán sinh tập phổ biến Apriori**

**Đầu vào:** DLGD **T**, *minsup*

**Đầu ra:** Tập FI

```
1. L1 = {1-itemset};// tập các item
2. For (k=2; Lk-1 ≠ ∅; k++) do
3.   Ck = AprioriGen(Lk-1) //sinh các kết hợp k-itemset
4.   Forall t ∈ T do
5.     Ct = subset(Ck , t)
6.     Forall c ∈ Ct do
7.       c.sup += 1/n //n là tổng số giao dịch
8.   Lk = {c ∈ Ck| c.sup ≥ minsup}
9. Trả về FI= ∪k Lk
```

**Mô tả thuật toán Apriori**:

Dòng 1, sinh tập $L_1$ chứa các *item* thỏa *minsup*. Dòng 3, sinh tập ứng viên $C_k$ tiềm năng từ tập phổ biến $L_{k-1}$. Từ dòng 4 đến 7, mỗi giao dịch t sinh tập con $C_t$ theo tập $C_k$ chứa ứng viên tiềm năng; với mỗi ứng viên c thuộc $C_k$ có trong $C_t$ thì được tính tần suất xuất hiện trên một giao dịch. Dòng 8, lọc ứng viên c từ $C_k$ thỏa *minsup* và đưa vào tập $L_k$ chứa k-*itemset* phổ biến.

Thủ tục AprioriGen - phát sinh $C_k$ chứa các ứng viên tiềm năng k-*itemset* từ $L_{k-1}$:

**Mã giả thủ tục AprioriGen**

**Đầu vào**: Tập $L_{k-1}$

**Đầu ra**: Tập $C_k$

```
1. Ck = {X ∪ X'| X, X' ∈ Lk-1, |X ∩ X'| = k – 2}
2. Forall itemset c ∈ Ck do
3.   Forall (k-1)-subset s of c do
4.     If (s ∉ Lk-1) then
5.       Ck = Ck - c
6. Trả về Ck
```

Thủ tục AprioriGen hoạt động dựa trên tính chất Apriori, tức là nếu một *itemset không phổ biến*, thì tất cả các *tập con* của nó cũng *không phổ biến*. Điều này giúp giảm không gian tìm kiếm bằng cách loại bỏ các ứng viên *itemset* không cần thiết.

Các bước thực hiện của thuật tục AprioriGen:

Từ tập $L_{k-1}$ chứa (k-1)-*itemset* phổ biến;

Dòng 1, tạo tập ứng viên k-*itemset* tiềm năng $C_k$: trong $L_{k-1}$ tìm các cặp (k-1)-*itemset* là X và X' có số *item* giống nhau và bằng (k-2) thì đưa vào $C_k$;

Dòng 2 đến 5: loại bỏ các ứng viên c trong $C_k$ khi tồn tại (k-1)-*itemset* con của c không là *itemset* phổ biến.

- Ưu điểm: thuật toán dễ hiểu và cài đặt;
- Nhược điểm: độ phức tạp tính toán tương đối cao – dòng 1 có độ phức tạp $O(n^2)$ và từ dòng 2 đến dòng 5 có độ phức tạp $O(n^3)$.

*2.2.2. Thuật toán sinh LKH*

Đầu tiên, Agrawal cùng đồng sự [1] đề xuất thuật toán SimplGenRules sinh LKH từ $l_k$ là k-*itemset* phổ biến (k>1). Thuật toán sinh LKH bắt đầu bằng cách chọn $a_{k-1}$ làm "tiền đề" là tập con (k-1)-*itemset* của k-*itemset*, từ đó sinh LKH $\{a_{k-1} \rightarrow (l_k - a_{k-1})\}$. Nếu LKH $\{a_{k-1} \rightarrow (l_k - a_{k-1})\}$ thỏa *minconf* thì gọi đệ quy sinh các LKH có "tiền đề" là tập con của $a_{k-1}$ (theo tính chất Apriori).

**Mã giả thủ tục SimpleGenRules**

**Đầu vào**: Tập $L_{k-1}$

**Đầu ra**: Tập luật kết hợp AR

1. **Forall** $l_k \in$ FI, k ≥ 2 **do**//sinh LKH từ 2-*itemset*
2. GenRules($l_k$, $l_k$)

**Mã giả thủ tục GenRules**

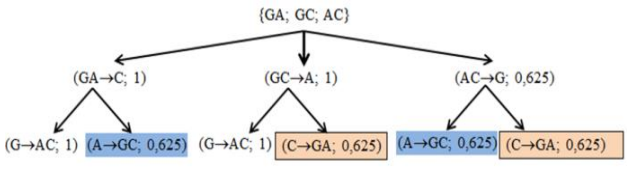
**Đầu vào**: k-*itemset* $l_k$, $a_m$ là m-*itemset* ở vế trái

**Đầu ra**: Tập luật kết hợp **AR** sinh từ $l_k$

1. A = {(m-1)-*itemsets* $a_{m-1}$| $a_{m-1} \subset a_m$}
2. **Forall** $a_{m-1} \in$ A **do**
3. *conf* = sup($l_k$)/sup($a_{m-1}$)//tính độ tin cậy
4. **If** (*conf* ≥ *minconf*) **then**
5. AR = AR + {$a_{m-1} \rightarrow (l_k - a_{m-1})$}
6. **If** (m – 1 > 1) **then**
7. GenRules($l_k$, $a_{m-1}$)
8. Trả về AR

**Minh họa thuật toán SimpleGenRules**:

Xét *itemset* (GAC; 0,50) thỏa *minsup* = 0,50 và sinh LKH thỏa *minconf* = 0,60:




***Hình 1.*** *Minh họa thuật toán SimpleGenRules sinh LKH*

Hình 1, minh họa thuật toán SimpleGenRules sinh LKH. Quá trình đệ quy sinh LKH từ 2-*itemset* từ (GA→C;1) và (AC→G;0,625) có trùng LKH (A→GC; 0,625); tương tự, sinh LKH từ (GC→A;1) và (AC→G; 0,625) có trùng LKH (C→GA; 0,625).

- Ưu điểm: thuật toán dễ hiểu và cài đặt;
- Nhược điểm: với mỗi k-*itemset* phổ biến thì thuật toán cần xem xét ($2^k$ – 2) LKH có thỏa *minconf* hay không trong trường hợp xấu nhất. Ngoài ra, trong quá trình đệ quy sinh các LKH có "tiền đề" là tập con, làm phát sinh nhiều LKH trùng nhau.

Từ nhược điểm trên, Agrawal và đồng sự [1] đã đề xuất cải tiến thuật toán sinh LKH - với từng $l_k$ là k-*itemset* phổ biến (k>1), sinh *k* LKH có "kết luận" là 1-*item*; tương ứng với các LKH thỏa *minconf*, các LKH tiềm năng có "kết luận" được tạo thành từ thủ tục AprioriGen (được trình bày ở trên) nhằm khắc phục sinh LKH trùng lặp.

**Mã giả thuật toán sinh LKH (AR-Ori)**

**Đầu vào**: Tập giao dịch T, ngưỡng *minconf*

**Đầu ra**: Tập luật kết hợp **AR**

1. AR = ∅
2. **Forall** $l_k \in$ FI, k ≥ 2 **do**//sinh LKH từ 2-*itemset*
3. $H_1$ = {chứa vế phải có 1-*item* của LKH sinh từ $l_k$}
4. AR =AR ∪ **Ap-GenRules**($l_k$, $H_1$)
5. Trả về AR

**Mã giả thủ tục Ap-GenRules**

**Đầu vào**: k-*itemset* $l_k$, tập m-*item* ở vế phải $H_m$

**Đầu ra**: Tập luật kết hợp **AR** sinh từ $l_k$

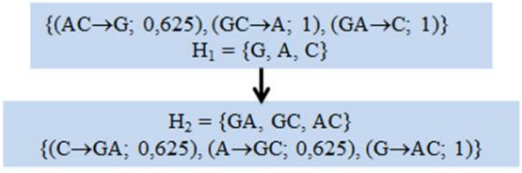
1. **If** (k > m+1) **then**
2. $H_{m+1}$ = **AprioriGen**($H_m$)//tạo vế phải (m+1)-*item*
3. **Forall** $h_{m+1} \in H_{m+1}$ **do**
4. *conf* = sup($l_k$)/sup($l_k - h_{m+1}$)//tính độ tin cậy
5. **If** (*conf* ≥ *minconf*) **then**
6. AR = AR ∪ {$l_k - h_{m+1} \rightarrow h_{m+1}$}
7. **Else**
   $H_{m+1} = H_{m+1} - h_{m+1}$//xóa $h_{m+1}$
8. Ap-GenRules($l_k$, $H_{m+1}$)
9. Trả về AR

**Minh họa thuật toán Ap-GenRules**:

Xét *itemset* (GAC; 0,50) thỏa *minsup* = 0,50 và sinh LKH thỏa *minconf* = 0,60:

Hình 2, minh họa thuật toán Ap-GenRules sinh LKH. Sinh đầy đủ ($2^3$ – 2) LKH từ *itemset* (GAC; 0,50) và không trùng lặp.

- Ưu điểm: khắc phục nhược điểm của thuật toán SimpleGenRules và dễ dàng cài đặt;
- Nhược điểm: như đã phân tích về thuật toán AprioriGen cho thấy độ phức tạp chung của thuật toán thuộc dạng đa thức $O(n^3)$.




***Hình 2.*** *Minh họa thuật toán Ap-GenRulessinh LKH*

## 3. Thuật toán đề xuất

Trong phần này, tác giả trình bày thuật toán QuickGen là một thuật toán được đề xuất để cải thiện hiệu suất và hiệu quả so với thuật toán phát sinh tập ứng viên AprioriGen được sử dụng ở cả hai giai đoạn.

### *3.1. Thuật toán đề xuất*

Để tối ưu hóa quá trình khai thác LKH: Năm 1997, Zaki và đồng sự đã đưa ra khái niệm về lớp tương đương (Equivalence Classes) bằng cách chỉ xem xét các cặp k-*itemset* có cùng (k-1) *items* đầu tiên có thứ tự để sinh ứng viên tiềm năng (k+1)-*itemset* [2].

**Ý tưởng đề xuất thuật toán QuickGen**:

Tác giả nhận thấy, quá trình khai thác LKH theo hướng tiếp cận của Agrawal sử dụng thủ tục AprioriGen trong cả hai giai đoạn và được lặp lại nhiều lần. Điều này, dẫn đến thời gian thực hiện khai thác LKH của thuật toán kém hiệu quả. Dưới đây là thuật toán QuickGen được đề xuất thay thế AprioriGen trong quá trình khai thác LKH:

*Thứ nhất*, sắp xếp các *item* theo thứ tự tăng dần của độ phổ biến để rút gọn các kết hợp theo tính chất Apriori;

*Thứ hai*, bổ sung 3 trường thông tin cho từng k-*itemset* trong $L_k$ là *min*, *max* và *sub*. Trường *min*, *max* là thứ tự nhỏ nhất và lớn nhất của *item* có trong k-*itemset*; trường *sub* là thứ tự của *item* thứ (k-1) – ban đầu được khởi tạo bằng *min*. Khi đó, ta cần xác định cặp k-*itemset* có cùng lớp tương đương thì chỉ thực hiện so sánh trường *min* và *sub* của cặp k-*itemset* trên là bằng nhau từng đôi một.

Cặp k-*itemset* $X_i$ và $X_j$ cùng lớp tương đương sẽ sinh ứng viên (k+1)-*itemset* là $X_\eta$ có trường *min* được giữ nguyên, trường *sub* và *max* được cập nhật:

$X_\eta.sub = MIN(X_i.max, X_j.max)$;

$X_\eta.max = MAX(X_i.max, X_j.max)$;

**Mã giả thuật toán QuickGen**

**Đầu vào**: Tập $L_{k-1}$

**Đầu ra**: Tập $C_k$

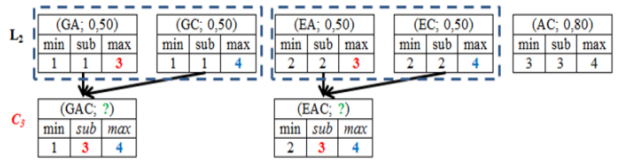
1. $C_k = \varnothing$
2. i = 1
3. **While** (i < |$L_k$-1|) **do**
4. j = i + 1
5. **Do** //$X_i$, $X_j \in L_{k-1}$
6. **If** (($X_i.min == X_j.min$) $\wedge$ ($X_i.sub == X_j.sub$)) **then**
7. $C_k = C_k \cup \{X_i \cup X_j\}$//$X_i$, $X_j$ cùng lớp tương đương
8. j++
9. **Else**
10. i = j
11. **While** (i $\neq$ j)
12. Trả về $C_k$

**Phân tích thuật toán**:

Thuật toán QuickGen được trình bày đơn giản và dễ cài đặt. Ngoài ra, độ phức tạp của thuật toán QuickGen dạng đa thức bậc hai $O(n^2)$ – làm tăng hiệu suất đáng kể trong quá trình khai thác LKH.

### *3.2. Minh họa thuật toán QuickGen*

Theo DLGD T ở Bảng 1, các *item* thỏa *minsup* = 0,50 là {G, E, A, C} và thứ tự của các *item* được biểu diễn từ 1 đến 4. Hình 3, minh họa bước sinh tập ứng viên $C_3$ từ tập phổ biến $L_2$ và các trường *sub* và *max* được cập nhật.




***Hình 3.*** *Minh họa thuật toán QuickGen sinh ứng viên*

## 4. Kết quả thực nghiệm

Để đánh giá hiệu suất của thuật toán đề xuất và thuật toán khác – các thuật toán được cài đặt bằng ngôn ngữ C#; so sánh thời gian thực hiện trên máy trính cấu hình Core i7-3540M 3.0 GHz và RAM 4Gb.

### *4.1. Mô tả dữ liệu thực nghiệm*

Trong cả 2 thực nghiệm, tác giả sử dụng 3 tập dữ liệu **Accidents**, **Connect** và **USCensus** từ UCI Machine Learning Repository - kho dữ liệu phổ biến trong lĩnh vực học máy và khai thác dữ liệu.

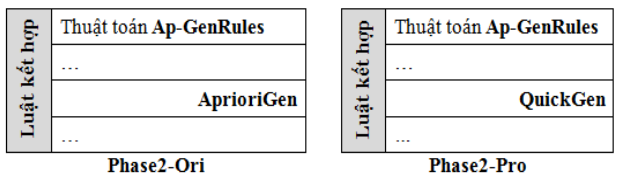
***Bảng 3.*** *Mô tả dữ liệu thực nghiệm*

| Dữ liệu giao dịch | Số items | Số giao dịch | Mật độ (%) |
|---|---|---|---|
| **Accidents** | 368 | 340.183 | 7,22 |
| **Connect** | 129 | 67.557 | 33,33 |
| **USCensus** | 392 | 13.369 | 17,34 |

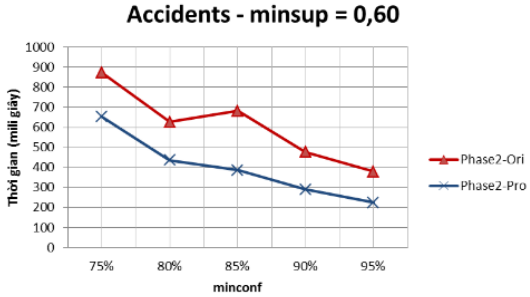
Bảng 3, mô tả 3 tập dữ liệu sử dụng trong cả 2 thực nghiệm, gồm các thông số như số lượng các *item*, số lượng giao dịch, mật độ của từng tập dữ liệu.

### *4.2. Thực nghiệm 1*

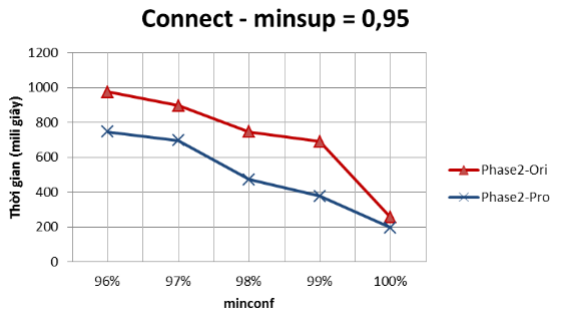
Trong phần thực nghiệm này, tác giả so sánh hiệu suất của 2 thuật toán ở giai đoạn sinh LKH: thuật toán sử dụng thủ tục AprioriGen nguyên bản (gọi là Phase2-Ori) và thuật toán QuickGen (gọi là Phase2-Pro). Hai thuật toán trên đều cho cùng số lượng LKH ở các ngưỡng *minconf* khác nhau (cố định *minsup* và thay đổi *minconf*).



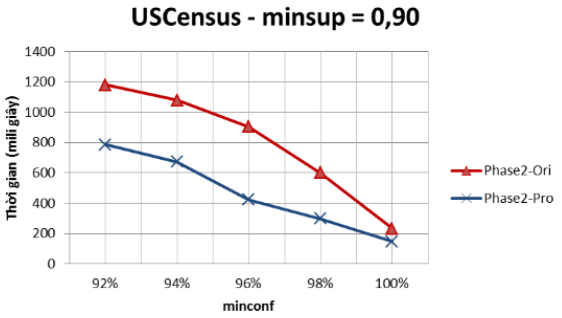

***Hình 4.*** *Minh họa so sánh hiệu suất theo thực nghiệm 1*




***Hình 5.*** *So sánh thời gian trên Accidents theo TN1*




***Hình 6.*** *So sánh thời gian trên Connect theo TN1*



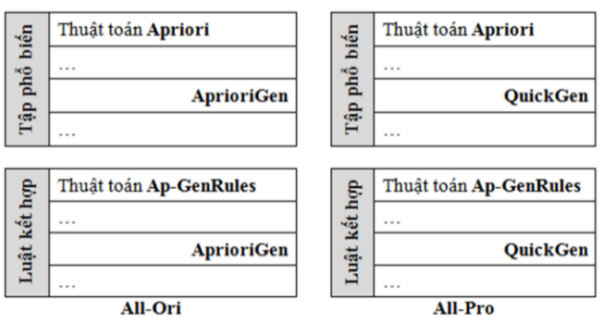

***Hình 7.*** *So sánh thời gian trên USCensus theo TN1*

Hình 5, 6 và 7 so sánh thời gian thực hiện của 2 thuật toán Phase2-Ori và Phase2-Pro lần lượt trên 3 tập dữ liệu Accidents, Connect và USCensus. Thuật toán đề xuất Phase2-Pro trung bình nhanh hơn Phase2-Ori trên các tập dữ liệu {35,83%; 30,37%; 42,32%}.

Kết quả thực nghiệm trên nhóm dữ liệu thực cho thấy rằng thuật toán Phase2-Pro có thời gian thực hiện nhanh hơn so với thuật toán Phase2-Ori trên các ngưỡng *minconf* khác nhau. Điều này có nghĩa là khi giảm ngưỡng *minsup*, thuật toán Phase2-Pro thường có hiệu suất cao hơn trong việc tìm kiếm các tập phổ biến và LKH. Tuy nhiên, kết quả có thể thay đổi tùy thuộc vào cấu trúc và kích thước của dữ liệu, cũng như cài đặt cụ thể của thuật toán.

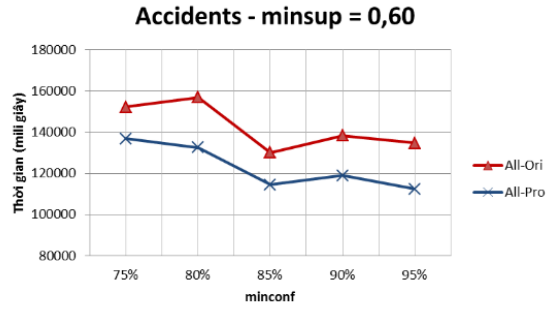
### *4.3. Thực nghiệm 2*

Phần thực nghiệm này, tác giả so sánh hiệu suất của 2 thuật toán gồm cả hai giai đoạn sinh tập phổ biến và LKH: thuật toán sử dụng 2 lần thủ tục AprioriGen (gọi là All-Ori) và thuật toán QuickGen (gọi là All-Pro). Hai thuật toán trên đều cho cùng số lượng LKH ở các ngưỡng *minconf* khác nhau.

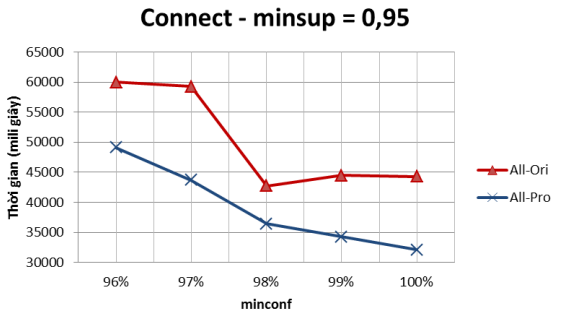



***Hình 8.*** *Minh họa so sánh hiệu suất theo thực nghiệm 2*

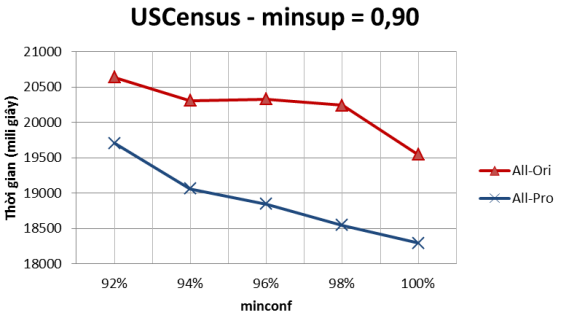
Hình 9, 10 và 11 so sánh thời gian thực hiện của 2 thuật toán All-Ori và All-Pro lần lượt trên 3 tập dữ liệu Accidents, Connect và USCensus. Thuật toán đề xuất All-Pro trung bình nhanh hơn All-Ori trên các tập dữ liệu {13,64%; 21,93%; 6,55%}. Ngoài ra, dựa vào thực nghiệm 1 và thực nghiệm 2, cho thấy: thời gian của giai đoạn sinh tập phổ biến chiếm nhiều thời gian hơn so với giai đoạn sinh LKH khi thực hiện trên 3 tập dữ liệu trên.




***Hình 9.*** *So sánh thời gian trên Accidents theo TN2*




***Hình 10.*** *So sánh thời gian trên Connect theo TN2*




***Hình 11.*** *So sánh thời gian trên USCensus theo TN2*

Qua hai kịch bản thực nghiệm trên, thuật toán đề xuất QuickGen được sử dụng trong Phase2-Pro và All-Pro đã mang lại hiệu suất ổn định cho quá trình khai thác LKH. Tuy nhiên, trên đây chỉ là thực nghiệm minh họa trên 3 tập dữ liệu từ kho dữ liệu UCI – để đánh giá mang tính khách quan hơn, nghiên cứu cần thực nghiệm trên nhiều dữ liệu có đặc trưng khác nhau: có mật độ cao hoặc thưa.

## 5. Kết luận và hướng phát triển

Trong nghiên cứu này, tác giả đề xuất giải pháp phát sinh tập ứng viên QuickGen dựa vào khái niệm cặp k-*itemset* cùng lớp tương đương với độ phức tạp bậc hai; đánh giá hiệu suất của QuickGen dựa trên hai kịch bản: thực nghiệm thứ nhất, so sánh hiệu suất áp dụng QuickGen ở giai đoạn sinh LKH; thực nghiệm thứ hai, so sánh hiệu suất trong cả quá trình khai thác LKH bao gồm giai đoạn

sinh tập phổ biến và sinh LKH từ tập phổ biến. Kết quả thực nghiệm cho thấy, sử dụng thuật toán QuickGen mang lại hiệu quả cao với cả hai kịch bản.

Qua nghiên cứu đề xuất thuật toán QuickGen trong khai thác LKH, tác giả nhận thấy AprioriGen cũng như QuickGen được sử dụng ở cả hai giai đoạn trong khai thác LKH; độ phức tạp thuật toán AprioriGen và QuickGen lần lượt là $O(n^3)$, $O(n^2)$ – cả hai thuật toán khó có thể đáp ứng trên dữ liệu lớn. Tuy nhiên, thuật toán QuickGen được xây dựng dựa trên dữ liệu có thứ tự và xác định nhanh cặp k-*itemset* cùng lớp tương đương, đây là cơ sở mở rộng QuickGen trên dữ liệu lớn theo hướng tiếp cận chia-để trị trong nghiên cứu tương lai.

## TÀI LIỆU THAM KHẢO


[1] R. Agrawal, R. Srikant, *Fast Algorithms for Mining Association Rules*, VLDB 1994, 1994.

[2] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, *New Algorithms for Fast Discovery of Association Rules*, KDD'97, 1997.

[3] J. Han, J. Pei, and Y. Yin, *Mining Frequent Patterns without Candidate Generation*, SIGMOD2000, 2000.

[4] P. Huan and L. Bac, *A Novel Algorithm for Frequent Itemsets Mining in Transactional Databases*, PAKDD 2018. LNCS, 11154, Springer Cham, 2018.

[5] M. Kryszkiewicz, Representative Association Rules and Minimum Condition Maximum Consequence Association Rules, *PKDD, European Symposium on Principles of Data Mining and Knowledge Discovery, France*, vol. 1510, pp. 361-369, 1998. DOI:10.1007/BFb0094839

[6] Y. Xin, W. Na, and W. Chunyu, A New Method for Eliminating Redundant Association Rules, *International Conference on Intelligent Computation Technology and Automation*, 2010. DOI: 10.1109/ICICTA.2010.129

[7] M. Belaid, C. Bessiere, and N. Lazaar, Constraint Programming for Association Rules, *Proceedings of the 2019 SIAM International Conference on Data Mining*, 2019, pp. 127-125. https://doi.org/10.1137/1.9781611975673.1

[8] P. Huan, An Efficient Mining Algorithm of Closed Frequent Itemsets on Multi-core Processor, *Advanced Data Mining and Applications*, 2019, pp. 107-118. DOI:10.1007/978-3-030-35231-8_8.

[9] A. Bai *et al*., "An efficient approach based on selective partitioning for maximal frequent itemsets mining", *Sādhanā,* vol. 44, no. 183, pp. 1-22, 2019.

[10] T.H. Phan and H.B. Le, "A Comprehensive Survey of Frequent Itemsets Mining on Transactional Database with Weighted Items", *Journal of Research and Development on Information and Communication Technology*, vol. 2021, no. 1, pp.18-27, 2021.