Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.00057

MỘT SỐ VẤN ĐỀ VỀ KHAI PHÁ ĐỒ THỊ CON THƯỜNG XUYÊN ĐÓNG

Hoàng Minh Quang[1], Vũ Đức Thi[2], Phạm Quốc Hùng[3]
[1]Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam.
[2]Viện Công nghệ thông tin, Đại học Quốc gia Hà Nội.
[3]Khoa Công nghệ thông tin - Đại học Sư phạm kỹ thuật Hưng Yên.
[1]hoangquang@ioit.ac.vn, [2]vdthi@vnu.edu.vn, [3]quochungvnu@gmail.com

TÓM TẮT— *Khai phá các mẫu thường xuyên là bài toán quan trọng có nhiều khả năng ứng dụng vào thực tiễn. Các ứng dụng trong thực tiễn rất đa dạng và phong phú nên phương pháp khai phá tập mục thường xuyên bị giới hạn bởi cấu trúc dữ liệu dạng tập hợp không phản ánh được hết bản chất của dữ liệu chẳng hạn như cấu trúc thành phần hóa học của các viên thuốc tân dược, cấu trúc gen tế bào, cấu trúc protein động vật và nhiều cấu trúc khác. Các cấu trúc dữ liệu này hầu hết đều có thể biểu diễn dưới một dạng dữ liệu có cấu trúc đã biết như đồ thị, cây hoặc lattice. Do vậy, các nghiên cứu về khai phá đồ thị con thường xuyên có ý nghĩa rất lớn đặc biệt hữu ích trong lĩnh vực y tế. Trong bài báo này, chúng tôi đưa ra một số nhận xét, đánh giá về các thuật toán khai phá đồ thị con thường xuyên hiện nay đồng thời cũng đề xuất một vài điểm thay đổi trong việc thực hiện khai phá đồ thị con thường xuyên nhằm tăng hiệu quả khai phá đồ thị con thường xuyên nhất là đồ thị con thường xuyên đóng.*

Từ khóa— *Khai phá dữ liệu, đồ thị con thường xuyên, khai phá đồ thị, dữ liệu có cấu trúc, đồ thị con thường xuyên đóng, độ phức tạp tính toán.*

I. GIỚI THIỆU

Khai phá dữ liệu là lĩnh vực rất quan trọng. Một trong các phương pháp khai phá dữ liệu có nhiều ứng dụng nhất là khai phá các mẫu thường xuyên. Vấn đề khai phá mẫu thường xuyên là từ một tập dữ liệu các đối tượng, với một ngưỡng độ hỗ trợ tối thiểu minsup cho trước, ta đi tìm các đối tượng có độ hỗ trợ lớn hơn hoặc ít nhất là bằng với độ hỗ trợ tối thiểu minsup. Dữ liệu có thể rất đa dạng từ dữ liệu nhị phân, dữ liệu số nguyên, số thực hoặc các dữ liệu có cấu trúc phức tạp hơn như cây, đồ thị, lattice v.v... Hầu hết các phương pháp khai phá mẫu thường xuyên đều sử dụng một nguyên lý chung là tính chất "Downward Closure Property" (DCP) hay còn gọi là tính chất phản đơn điệu. Các tập dữ liệu bảo toàn tính chất DCP đều có thể áp dụng thuật toán tựa Apriori để khai phá mẫu thường xuyên. Về mặt cơ bản, thuật toán Apriori gồm hai bước: thứ nhất là bước sinh tập ứng viên và thứ hai là tỉa các tập ứng viên dựa trên tính chất DCP. Ví dụ trong khai phá tập mục thường xuyên, một đối tượng dữ liệu là một giao tác và tập dữ liệu là tập giao tác. Trong mỗi giao tác sẽ chứa một số mục dữ liệu là có xuất hiện hay không xuất hiện trong giao tác đó. Khai phá tập mục thường xuyên là tìm ra tất cả các tập mục mà có tần suất xuất hiện trong một số giao tác lớn hơn một ngưỡng cho trước nào đó. Công việc này rất đơn giản ta chỉ việc đếm một tập các mục mà đồng thời tất cả các mục trong tập đó đều xuất hiện trên một số giao tác sao cho số lần xuất hiện đủ lớn hơn một ngưỡng thì tập đó là thường xuyên. Và ta thấy rằng, một tập là thường xuyên thì tập con của nó cũng là thường xuyên và ngược lại một tập là không thường xuyên thì tập cha của nó cũng là không thường xuyên. Đây chính là tính chất DCP trong khai phá tập mục thường xuyên. Từ tính chất này, vấn đề sinh tập ứng viên lần lượt tập có k-mục là thường xuyên ta xây dựng tập (k+1)-mục và đi tìm xem các tập (k+1)-mục nào là thường xuyên với k thực hiện từ 1 đến hết số lượng mục có trong cơ sở dữ liệu giao tác.

Nhiều lĩnh vực hiện nay đòi hỏi khai phá mẫu thường xuyên trên tập dữ liệu có cấu trúc phức tạp hơn chẳng hạn như cấu trúc hóa học các hợp chất, cấu trúc gen tế bào, cấu trúc các thành phần thuốc, v.v... Hầu hết các cấu trúc phức tạp đều có thể được biểu diễn dưới dạng cây hoặc đồ thị. Khai phá mẫu thường xuyên trên tập dữ liệu có cấu trúc phức tạp chẳng hạn như cây hoặc đồ thị phức tạp hơn rất nhiều lần so với khai phá tập mục thường xuyên. Tính chất DCP vẫn được đảm bảo cho dữ liệu cây hoặc đồ thị nghĩa là nếu một đồ thị/cây là thường xuyên thì đồ thị con/cây con cũng là thường xuyên và ngược lại nếu một đồ thị/cây là không thường xuyên thì đồ thị cha/cây cha của nó cũng là đồ thị/cây không thường xuyên. Mặc dù tính chất DCP được đảm bảo nhưng vấn đề sinh ứng viên lại gặp nhiều khó khăn vì với một tập đỉnh và tập cạnh cho trước, việc tìm đồ thị con/cây con với tập đỉnh và tập cạnh đó có phải là đồ thị con/cây con của một đồ thị/cây đã cho hay không là một vấn đề không dễ giải quyết. Vấn đề này được gọi là tìm đồ thị con đẳng cấu (subgraph isomorphism). Nhiều công trình nghiên cứu đã chứng minh rằng việc xác định chính xác một đồ thị có phải là đồ thị con đẳng cấu của một đồ thị hay không có độ phức tạp tính toán thuộc lớp NP-complete [Garey và Johnson 1979]. Nếu cấu trúc dữ liệu là cây thì việc xác định đồ thị con đẳng cấu đã có thể giải quyết trong thời gian đa thức [Chi 2004; Tsur và Shamir 1999]. Những thách thức này dẫn đến nhiều công trình nghiên cứu làm tăng hiệu quả vấn đề xác định đồ thị con đẳng cấu như các thuật toán gSpan [Yan và Han 2002], FFSM [Huan 2003], FSG [Kuramochi 2001]. Tuy nhiên các công trình này vẫn phải giải quyết vấn đề tìm đồ thị con đẳng cấu trong thời gian không đa thức.

Khai phá đồ thị con thường xuyên là một phương pháp khai phá dữ liệu hiệu quả. Tuy nhiên, các ứng dụng thực tiễn hiện nay với các tập dữ liệu vừa có cấu trúc phức tạp lại vừa có kích thước rất lớn đã dẫn đến việc tìm tập tất cả các đồ thị con thường xuyên cũng là rất lớn. Hơn hết, có một số đồ thị thường xuyên lại có độ hỗ trợ bằng với đồ thị thường xuyên cha của nó. Vì thế, việc tìm tập tất cả các đồ thị con thường xuyên đóng có hiệu quả trong các ứng dụng thực tiễn hơn. Bởi từ đồ thị thường xuyên đóng ta có thể tìm ra tất cả các đồ thị là con của đồ thị đó nên việc liệt kê hết

các đồ thị con thường xuyên của một đồ thị thường xuyên đóng làm tốn thêm bộ nhớ lưu trữ. Tuy lúc cần có thể tìm các đồ thị con thường xuyên nhanh hơn nhưng nếu số lượng đồ thị đầu vào lớn và số lượng đồ thị con thường xuyên là lớn thì việc liệt kê hết không thể hiệu quả bằng chỉ liệt kê các đồ thị con thường xuyên đóng.

Trong bài báo này, chúng tôi đề xuất một kết quả có thể làm tăng hiệu quả khai phá đồ thị con thường xuyên nhất là đồ thị con thường xuyên đóng. Với một cách nhìn khác với các thuật toán của gSpan, FFSM, FSG và các công trình nghiên cứu liên quan khác chúng tôi đã giảm được thời gian tính toán trong việc khai phá đồ thị con và thuật toán của chúng tôi hiệu quả hơn nữa khi áp dụng vào khai phá đồ thị con thường xuyên đóng.

II. MỘT SỐ ĐỊNH NGHĨA

Một đồ thị gắn nhãn G là một bộ G = (V,E,$\sum_V$,$\sum_E$,l) với V là tập đỉnh, E ⊂ V × V là tập cạnh. $\sum_V$ và $\sum_E$ là nhãn của đỉnh và cạnh tương ứng. Hàm gắn nhãn l là ánh xạ V → $\sum_V$ và E → $\sum_E$. Không mất tính tổng quát, ta giả sử có một thứ tự toàn thể ≼ trên tập nhãn→ $\sum_V$ và → $\sum_E$.

Cho một cặp đồ thị G = (V,E,$\sum_V$,$\sum_E$,l) và G' = (V',E',$\sum_{V'}$,$\sum_{E'}$,l'), G là đồ thị con của G' nếu và chỉ nếu:

(1.) V ⊆ V'

(2.) ∀ u ∈ V, (l(u) = l'(u))

(3.) E ⊆ E'

(4.) ∀ (u,v) ∈ E, (l(u,v) = l'(u,v))

G' được gọi là đồ thị cha của G

Hai đồ thị G = (V,E,$\sum_V$,$\sum_E$,l) và G' = (V',E',$\sum_{V'}$,$\sum_{E'}$,l') là đẳng cấu nếu và chỉ nếu tồn tại một song ánh f:V → V' thỏa mãn:

(1.) ∀ u ∈ V, (l(u) = l'(f(u)))

(2.) ∀ u,v ∈ V, ((u,v) ∈ E) ↔ (f(u),f(v)) ∈ E'

(3.) ∀ (u,v) ∈ E, (l(u,v) = l'(f(u),f(v))

Đồ thị G là đồ thị con đẳng cấu của G', ký hiệu G ⊆ G', nếu và chỉ nếu tồn tại một đồ thị con G'' của G' mà G đẳng cấu với G''

Cho một tập dữ liệu đồ thị GD và một ngưỡng σ (0 <σ< 1), độ hỗ trợ của G, ký hiệu sup_G được xác định như một phân số các đồ thị trong GD với G là một đồ thị con đẳng cấu của G':

$$sup_G = \frac{\{G' \in GD | G \subseteq G'\}}{|GD|}$$

G là đồ thị thường xuyên nếu và chỉ nếu $sup_G \geq \sigma$. Vấn đề khai phá đồ thị con thường xuyên là cho một ngưỡng σ và một cơ sở dữ liệu đồ thị GD phải tìm tất cả các đồ thị con thường xuyên trong GD.

III. PHƯƠNG PHÁP TIẾP CẬN KHAI PHÁ ĐỒ THỊ CON THƯỜNG XUYÊN

Vấn đề khai phá đồ thị con thường xuyên là một phương pháp khai phá đồ thị cùng với một số các phương pháp cải tiến trong nhiều công việc như phân loại các hợp chất hóa học [Huan 2004c; Deshpande 2005], phân cụm tài liệu ảnh [Barbu 2005], đánh chỉ số đồ thị [Shasha 2002, Yan 2004], tìm kiếm đồ thị [Yan 2005; 2006; Chen 2007] v.v... Có hai phương pháp tiếp cận khai phá đồ thị con thường xuyên là phương pháp tiếp cận theo Apriori và phương pháp tiếp cận phát triển mẫu.

Algorithm 2: Pattern growth approach

Input : g = a frequent subgraph, σ = minimum support, $\mathbb{GD}$ = a graph dataset
Output: $\mathbb{F}$, a set of frequent subgraphs

```
1 F ← ∅;
2 F1 ← detect all frequent 1-subgraphs in GD;
3 k ← 1;
4 foreach g ∈ F1 do
5 |   Pattern-growth(g, F1, σ, F);
6 end
```

Hai thuật toán tương ứng với hai tiếp cận khác nhau. Thuật toán tiếp cận theo Apriori áp dụng chiến thuật tìm kiếm theo bề rộng, thuật toán tiếp cận theo phát triển mẫu áp dụng chiến thuật tìm kiếm theo độ sâu. Cả hai phương pháp đều có những ưu nhược điểm riêng nhưng về cơ bản vẫn là sinh ra một tập đồ thị con và kiểm tra nó có phải là đồ thị con đẳng cấu với một đồ thị nằm trong tập dữ liệu đồ thị hay không từ đó xác định độ hỗ trợ của đồ thị con ứng viên đó và kết luận đồ thị con ứng viên có thuộc tập đồ thị con thường xuyên hay không.

Function Pattern-growth$(g, \mathbb{GD}, \sigma, \mathbb{F})$

$k \leftarrow k + 1$;
$C_k \leftarrow \emptyset$;
if $g \in \mathbb{F}$ **then**
  return;
else
  $\mathbb{F} \leftarrow \mathbb{F} \cup g$;
end
scan $\mathbb{GD}$, find all the edges e such that g can be extended to $g \cup e, g \leftarrow g \cup e$, and insert g into C_k;
foreach $g_k \in C_k$ **do**
  if $g_k.count \geq \sigma|\mathbb{GD}|$ **then**
    Pattern-growth$(g_k, \mathbb{GD}, \sigma, \mathbb{F})$;
  else
    return;
  end
end

Algorithm 1: Apriori-based approach

Input : $\mathbb{GD}$ = a graph dataset, σ = minimum support
Output: $\mathbb{F}_1, \mathbb{F}_2, ..., \mathbb{F}_k$, a set of frequent subgraphs sets

$\mathbb{F}_1 \leftarrow$ detect all frequent 1-subgraph in $\mathbb{GD}$;
$k \leftarrow 2$ **while** $\mathbb{F}_k \neq \emptyset$ **do**
  $\mathbb{F}_k \leftarrow \emptyset$;
  $C_k \leftarrow candidate - gen(\mathbb{F}_{k-1})$;
  foreach *candidate* $g_k \in C_k$ **do**
    $g_k.count \leftarrow 0$ **foreach** $G_i \in \mathbb{GD}$ **do**
      if *subgraph-isomorphism*(g_k, G_i) **then**
        $g_k.count \leftarrow g_k.count + 1$;
      end
    end
    if $g_k.count \geq \sigma|\mathbb{GD}| \wedge g_k \notin \mathbb{F}_k$ **then**
      $\mathbb{F}_k = \mathbb{F}_k \cup g_k$;
    end
  end
  $k \leftarrow k + 1$;
end

Ta có thể nhận thấy rằng hầu hết các thuật toán trên vẫn vướng vào việc xác định đồ thị con đẳng cấu cho mỗi đồ thị con ứng viên được sinh ra. Vấn đề ở chỗ chúng ta không biết có bao nhiêu ứng viên được sinh ra khi kết hợp hai đồ thị con mức k hoặc mở rộng một đồ thị con mức k thành đồ thị ứng viên mức (k+1). Do đó quá trình này sẽ là, với mỗi đồ thị con ứng viên mức (k+1) được sinh ra, và với mỗi đồ thị trong tập dữ liệu đối tượng đồ thị ta sẽ phải xác định đồ thị con ứng viên mức (k+1) này có phải là đồ thị con đẳng cấu của đồ thị trong tập dữ liệu đồ thị hay không. Việc xác định này có độ phức tạp tính toán thuộc lớp NP. Nếu đồ thị con ứng viên mức (k+1) này là đồ thị con đẳng cấu với các đồ thị trong tập dữ liệu thì độ hỗ trợ của nó cũng tăng lên tương ứng và nếu đồ thị con ứng viên mức (k+1) này có độ hỗ trợ lớn hơn một ngưỡng minsup σ cho trước thì đồ thị con ứng viên này sẽ là một đồ thị con thường xuyên. Rõ ràng rằng ở đây ta thấy có thể có một cải tiến cho phương pháp xác định đồ thị con đẳng cấu cho một đồ thị và xác định độ hỗ trợ của nó.

IV. PHÂN TÍCH GSPAN VÀ FFSM

Đồ thị đẳng cấu là một dạng đối sánh một tương ứng một giữa các đỉnh của một đồ thị và các đỉnh của đồ thị khác mà bảo toàn được các đỉnh kề. Phát hiện đồ thị đẳng cấu là bước quan trọng trong sinh đồ thị con ứng viên trong khai phá đồ thị thường xuyên. Đồ thị đẳng cấu thì chưa biết có thể giải quyết trong thời gian đa thức hay không đa thức. Tuy nhiên phát hiện đồ thị con đẳng cấu được biết là thuộc lớp NP-complete [Garey và Johnson 1979]. Khi giới hạn đồ thị thành cây thì độ phức tạp giảm đi có thể phát hiện cây con đẳng cấu trong thời gian đa thức [Hopcroft và Tarjan 1972; Matula 1978; Chung 1987; Shamir và Tsur 1999]. Vấn đề xác định đồ thị con đẳng cấu còn được áp dụng trong rất nhiều lĩnh vực như nhận dạng mẫu [Lu 1991], phân tích hình dạng [Pearce 1994], học máy [Cook và Holder 1994]. Nhiều thuật toán tìm mọi cách để né tránh vấn đề phát hiện đồ thị con đẳng cấu như các thuật toán [Ullman 1976; Schmidt 1976; McKay 1981 v.v...].

Các phương pháp xác định đồ thị con đẳng cấu trong vấn đề khai phá đồ thị con thường xuyên hiện nay hiệu quả nhất vẫn là gSpan và FFSM. Cả hai thuật toán này đều sử dụng một thứ tự cho đồ thị xây dựng nên một bộ mã cho mỗi đồ thị, đồng thời xác định một mã bé nhất hoặc lớn nhất trong bộ mã của một đồ thị và coi là mã chuẩn. Do đó một đồ thị mặc dù được biểu diễn bởi nhiều bộ mã khác nhau nhưng chỉ có duy nhất một mã chuẩn. Với mỗi đồ thị ứng viên, việc xác định xem nó có phải là đồ thị con đẳng cấu của một đồ thị hay không phụ thuộc vào một số đỉnh và số cạnh của đồ thị con và số đỉnh và số cạnh của đồ thị trong tập dữ liệu đồ thị được đem ra so sánh. Sau đây ta sẽ phân tích hai thuật toán được coi là có hiệu quả tốt nhất trong thời điểm hiện nay là gSpan và FFSM.

Algorithm 3: gSpan-Miner(c, σ, $\mathbb{GD}$, F)

Input : c = a subgraph represented by a DFS code, σ = minimum support, $\mathbb{GD}$ = a graph dataset
Output: $\mathbb{F}$ a set of frequent subgraphs

```
1  sort labels of the vertexes and edges in GD by their frequency;
2  remove infrequent vertexes and edges;
3  relabel the remaining vertexes and edges in descending frequency;
4  F_1 ← { all frequent 1-edge subgraphs in GD};
5  sort F_1 in DFS lexicographic order;
6  F ← ∅;
7  foreach c ∈ F_1 do
8  |  subgSpan(c, GD, σ, F);
9  |  GD ← GD − c;
10 |  if |GD| < σ then
11 |  |  break;
12 |  end
13 end
```

Procedure subgSpan(c, $\mathbb{GD}$, σ, $\mathbb{F}$)

Input :
Output:

```
1  if c ≠ min(c) then
2  |  return;
3  end
4  F ← F ∪ {c};
5  C ← ∅;
6  Scan GD once, find every edge e such that c can be right-most extended to c∪e, C ← c∪e;
7  Sort C in DFS lexicographic order;
8  foreach g_k ∈ C do
9  |  if support(g_k) ≥ σ then
10 |  |  subgSpan(g_k, GD, σ, F);
11 |  end
12 end
```

Trong thuật toán gSpan-Miner, tác giả sử dụng mã DFS với thứ tự lexicographic order để xác định một mã chuẩn duy nhất cho một đồ thị. Xem xét dòng số 8 trong gSpan-Miner sẽ gọi thủ tục subgSpan. Trong thủ tục subgSpan, nếu một đồ thị mà không có mã DFS nhỏ nhất theo thứ tự lexicographic order thì loại đi, nếu đồ thị con *c* thỏa mãn mã DFS thì được thêm vào tập đồ thị con thường xuyên. Sau đó quét toàn bộ cơ sở dữ liệu và tìm các cạnh *e* có thể gắn vào đồ thị con thường xuyên *c* để sinh ra đồ thị con mới đưa vào tập đồ thị con ứng viên *C* và sắp xếp tập đồ thị con ứng viên *C* theo thứ tự của lexicographic và với mỗi đồ thị con ứng viên trong *C* thì sẽ được gọi đệ quy để xác định các đồ thị con mức (k+1) tiếp theo của đồ thị của đồ thị con *c* có được đưa vào tập đồ thị con thường xuyên hay không. Rõ ràng với phương pháp duyệt theo độ sâu của gSpan với mã DFS và thứ tự lexicographic order thì bài toán sinh ứng viên và kiểm tra đồ thị con đẳng cấu thể hiện ở dòng 6 của thủ tục subgSpan. Mặc dù sử dụng right-most extended để search toàn bộ tập dữ liệu đồ thị GD để thêm các cạnh vào một đồ thị con thường xuyên *c* thì tập ứng viên được tìm thấy vẫn nằm trong độ phức tạp tính toán là NP vì với mọi đồ thị con thường xuyên *c* ở mức k thì thủ tục đệ quy subgSpan đều được gọi để sinh ra các đồ thị con mức (k+1) của *c*.

Algorithm 4: FFSM-Miner($C, \sigma, \mathbb{GD}, \mathbb{F}$)

Input : C = a suboptimal CAM list, σ = minimum support, $\mathbb{GD}$ = graph dataset
Output: $\mathbb{F}$, a set of frequent subgraphs

```
1  F ← { the CAMs of the frequent vertexes and edges };
2  F_1 ← { the CAMs of the frequent edges };
3  FFSM-Search(F_1, F);
```

Trong thuật toán FFSM-Miner ta thấy dòng 1 là khởi tạo tập đồ thị con thường xuyên F ban đầu, dòng 2 khởi tạo đồ thị con thường xuyên F_1 chỉ chứa cạnh ban đầu, dòng 3 thực sự là nơi thực hiện thuật toán và nó gọi thủ tục đệ quy FFSM-Search. Trong thủ tục đệ quy FFSM-Search, đầu vào là một tập các đồ thị con tối ưu theo nghĩa CAM (canonical adjacency matrix), dạng chuẩn của ma trận kề của đồ thị. Thủ tục FFSM-Search thực hiện như sau: với mỗi đồ thị con tối ưu *P* trong tập đồ thị con tối ưu *W*, nếu đồ thị con tối ưu *P* là một CAM thì nó thuộc tập đồ thị con thường xuyên. Tại dòng 4 của FFSM-Search gán tập đồ thị con ứng viên *C* trạng thái rỗng. Từ dòng 5 đến dòng 7 trong thủ tục FFSM-Search sẽ sử dụng toàn bộ các đồ thị con tối ưu trong tập đồ thị con tối ưu *W* để kết hợp lại với nhau sinh ra các đồ thị con ứng viên *C*, dòng 8 sẽ mở rộng các đồ thị con tối ưu *P* để sinh đồ thị con ứng viên vào trong *C*. Và quá trình tiếp tục khi dòng 10 gọi đệ quy thủ tục FFSM-Search. Dòng 9 sẽ kiểm tra các đồ thị con ứng viên trong *C* có thỏa mãn tính chất thường xuyên hay không và xóa nó khỏi tập ứng viên nếu nó không thường xuyên hoặc không tối ưu. Như vậy ta có thể thấy rằng tập ứng viên được sinh ra vẫn nằm trong độ phức tạp thời gian tính toán thuộc lớp NP vì vẫn phải sinh ra hết các đồ thị con ứng viên và kiểm tra xem nó có thường xuyên hay không.

```
Procedure FFSM-Search(W, 𝔽)
  Input  : W = a suboptimal CAM list
  Output: 𝔽, a set of frequent subgraphs
1 foreach P ∈ W do
2   if P.isCAM then
3     𝔽 ← 𝔽 ∪ {P};
4     C ← ∅;
5     foreach Q ∈ W do
6       C ← C ∪ FFSM − Join(P, Q);
7     end
8     C ← C ∪ FFSM − Extension(P);
9     remove CAM(s) from C that is either infrequent or not suboptimal;
10    FFSM-Search(C, 𝔽);
11  end
12 end
```

V. THUẬT TOÁN TÌM ĐỒ THỊ CON THƯỜNG XUYÊN ĐÓNG

Yan và Han sử dụng thuật toán gSpan và đưa ra thuật toán khai phá đồ thị con thường xuyên đóng. Đồ thị con thường xuyên đóng là đồ thị con đảm bảo hai tiêu chí: một là đồ thị thường xuyên, hai là đóng dưới quan hệ độ hỗ trợ. Nếu g là một đồ thị con của g' thì g' là đồ thị cha của g, ký hiệu $g \subseteq g'$ và là cha thực sự (proper subgraph) nếu $g \subset g'$ tức là g' phải có số đỉnh hoặc số cạnh nhiều hơn g. Cho một tập dữ liệu đồ thị được gắn nhãn, $D = \{ G_1, ..., G_n\}$, tập FS chứa tất cả các đồ thị con thường xuyên tức là độ hỗ trợ mọi đồ thị g trong FS là $sup_g \geq \sigma$ (với σ ngưỡng độ hỗ trợ tối thiểu hoặc gọi là min_sup). Tập tất cả các đồ thị con thường xuyên đóng, ký hiệu CS = { g | g ∈ FS \wedge \not \exists g' ∈ FS: g \subset g' \wedge $sup_g = sup_{g'}$ } tập tất cả các đồ thị con thường xuyên mà trong đó không có đồ thị nào là đồ thị cha của đồ thị khác đồng thời lại có cùng độ hỗ trợ với nó. Như vậy, $CS \subseteq FS$ và vấn đề khai phá đồ thị con thường xuyên đóng là tìm tập đầy đủ của CS trong tập dữ liệu đồ thị D với ngưỡng độ hỗ trợ tối thiểu min_sup (σ).

```
Algorithm 5: CloseMining(D, min_sup, S)
  Input  : A graph dataset D và min_sup
  Output: The closed frequent graph set S
1 remove infrequent vertices and edges;
2 S^0 ← code of frequent graphs with single vertex;
3 S ← S^0;
4 foreach code s ∈ S^0 do
5   CloseGraph(s, NULL, D, min_sup, S)
6 end
```

Tương tự như thuật toán gSpan-Miner, thuật toán CloseMining sử dụng cùng một phương pháp chỉ khác ở chỗ kiểm tra điều kiện trong dòng 10 đến 12 của thủ tục đệ quy CloseGraph xem có tồn tại một đồ thị cha nào mà có cùng độ hỗ trợ với nó hay không. Về mặt bản chất, việc sinh tập ứng viên và vấn đề xác định đồ thị con đẳng cấu không có gì thay đổi vẫn thuộc lớp NP.

Procedure CloseGraph(s, p, D, min_sup, S)

Input : A DFS code s, its parent p, a graph dataset D, and min_sup
Output: The closed frquent graph set S

1 **if** $s \neq min(s)$ **then**
2 | return;
3 **end**
4 **if** $\exists e', g' = g_p \cup e' \wedge g' < g_s \wedge I(g_p, D) = L(g_p, g', D) \wedge g_p$ *not a failure case of early termination* **then**
5 | return;
6 **end**
7 set C to $\emptyset$;
8 scan D once, find every edge e such that s can be extended to frequent $s \cup e$; insert $s \cup e$ into C;
9 detect any possible failure of early termination in s;
10 **if** $\nexists s \cup e \in C, sup_s = sup_{s \cup e}$ **then**
11 | insert s into S;
12 **end**
13 remove $s \cup e$ from C which cannot be right-most extended from s;
14 sort C in DFS lexicographic order;
15 **foreach** $s \cup e \in C$ **do**
16 | CloseGraph($s \cup e, e, s, D$, min_sup, S);
17 **end**

VI. THUẬT TOÁN PSI-CFSM

Trong bài báo này, chúng tôi đưa ra một phương pháp về mặt tính toán là tối ưu hơn so với gSpan và FFSM. Chúng tôi cũng sử dụng thứ tự cho nhãn của đỉnh và cạnh giống gSpan và FFSM. gSpan sử dụng mã DFS để xác định mỗi đồ thị con chỉ có một biểu diễn duy nhất, FFSM sử dụng CAM để biểu diễn sự duy nhất của đồ thị con. Chúng tôi sử dụng CAM để biểu diễn sự duy nhất của đồ thị con và áp dụng một phương pháp khai phá đồ thị con thường xuyên mới và gọi tên là Polynomial Subgraph Isomorphism Closed Frequent Subgraph Mining (PSI-CFSM).

Như trên đã đề cập, thuật toán gSpan sử dụng mã DFS để xác định biểu diễn duy nhất cho một đồ thị. Mã này được gọi là Minimum DFS Code (M-DFSC). Cho một đồ thị, giả sử coi mỗi đỉnh là một gốc thì từ gốc đó theo phương pháp duyệt theo độ sâu sẽ có rất nhiều cây bao trùm của đồ thị đó. Như vậy ở đây không có tính duy nhất về mặt biểu diễn. Vì vậy, [Yan và Han 2002] đã đề xuất một phương pháp mã chuẩn dựa trên nhãn của đồ thị bằng cách gán mỗi định danh duy nhất cho mỗi đỉnh và mỗi cạnh của đồ thị trong mã DFS sẽ được biểu diễn bởi một bộ 5 thành phần (i, j, l_i, l_e, l_j) với i và j là định danh của đỉnh, l_i và l_j là các nhãn tương ứng, l_e là nhãn của cạnh nối giữa hai đỉnh tương ứng. Dựa trên một thứ tự gọi là DFS lexicographic order, M-DFSC của đồ thị g được gọi là mà chuẩn của g. Mã chuẩn này chính là mã nhỏ nhất theo phương pháp duyệt cây theo độ sâu mà mỗi bước sẽ thêm vào một cạnh trong mã DFS.

Thuật toán FFSM thì sử dụng CAM để biểu diễn sự duy nhất của một đồ thị. Với một đồ thị g sẽ có (n!) ma trận kề của đồ thị g với n là số đỉnh của đồ thị. Cho một ma trận kề M của một đồ thị g, mã của ma trận M là một chuỗi các phần tử của phần tam giác dưới hoặc trên của một ma trận M bao gồm cả các phần tử nằm trên đường chéo chính. Với mỗi hoán vị của ma trận M ta sẽ thu được một mã của ma trận M. Tuân theo một thứ tự được xác định trên tập nhãn của đỉnh và tập nhãnh của cạnh của đồ thị g ta sẽ tìm được mã lớn nhất hoặc nhỏ nhất trong tất cả các hoán vị của ma trận M. Mã lớn nhất hoặc nhỏ nhất này sẽ được coi là mã chuẩn để biểu diễn duy nhất cho một đồ thị g thay cho (n!) biểu diễn của một đồ thị g. Ma trận kề mà lớn nhất hoặc nhỏ nhất được gọi là dạng chuẩn ma trận kề (Canonical Adjacency Matrix, hoặc gọi tắt là CAM) [Inokuchi 2000, 2002; Kuramuchi 2001; Huan 2003]. Theo đó, một CAM biểu diễn duy nhất cho một đồ thị g và một đồ thị g chỉ có một CAM. Với các phương pháp vét cạn việc xác định CAM của một đồ thị cũng có độ phức tạp tính toán là O(n!) thuộc lớp NP. Tuy nhiên, với việc sử dụng thứ tự nhãn của đỉnh và cạnh của đồ thị thì việc xác định CAM sẽ có độ phức tạp tính toán thời gian đa thức.

Định lý. Cho tập đồ thị gắn nhãn cho cả đỉnh và cạnh GD với một thứ tự toàn phần trên nhãn của đỉnh và cạnh. Tồn tại một thuật toán tìm tập chứa tất cả các đồ thị con thường xuyên đóng của GD mà vấn đề xác định đồ thị con đẳng cấu trong quá trình sinh tập ứng viên và đếm độ hỗ trợ của đồ thị con được thực hiện trong thời gian đa thức.

Algorithm 6: PSI-CFSM(D, σ = min_sup)

Input : Tập dữ liệu đồ thị $\mathbb{GD}$ và σ = min_sup
Output: $CS_2, CS_3, ..., CS_k$, tập các đồ thị con thường xuyên đóng tương ứng với số đỉnh các tập đó chứa

1 xây dựng một danh sách liên kết được sắp thứ tự theo CAM C_2^i chỉ chứa các 2-subgraph;
2 **foreach** $u \in C_2^i$ **do**
3 | tìm kiếm nhị phân u trong C_2^j và xác định $sup_u \geq \sigma$ đưa vào FS_2^i và FS_2^D;
4 **end**
5 $k \leftarrow 3$;
6 **while** *sinh ra ứng viên mức k trong* $g_i \in \mathbb{GD}$ *từ* FS_2^i *và* CS_{k-1}^i **do**
7 | xây dựng danh sách liên kết được sắp thứ tự theo CAM C_k^i;
8 | **foreach** $u \in C_k^i$ **do**
9 | | thực hiện tìm kiếm nhị phân để tìm u trong C_k^j và xác định $sup_u \geq \sigma$ để đưa vào CS_k^i và CS_k;
10 | | kiểm tra $v \in CS_{k-1}^i$ nếu $sup_v = sup_u$ thì loại v ra khỏi CS_{k-1};
11 | **end**
12 | $k \leftarrow k + 1$;
13 **end**

Chúng tôi đưa ra phương pháp xác định đồ thị con đẳng cấu bằng cách sử dụng CAM để biểu diễn duy nhất cho một đồ thị g trong tập dữ liệu đồ thị đầu vào GD. Cùng với CAM chúng tôi chia tách vấn đề khai phá đồ thị con thường xuyên thành khai phá đồ thị con thường xuyên theo từng mức. Bắt đầu từ mức đồ thị chỉ có 1 cạnh tức là chỉ có 2 đỉnh và chúng tôi gọi các đồ thị này là 2-subgraph. Đầu tiên sẽ khai phá tất cả các đồ thị 2-subgraph, tìm tất cả các đồ thị 2-subgraph là thường xuyên và lưu lại. Ở đây chúng tôi sử dụng nhiều tập lưu trữ các 2-subgraph thường xuyên. Tập CS_2 sẽ lưu toàn bộ các 2-subgraph thường xuyên đóng của toàn bộ tập dữ liệu đồ thị GD và các tập FS^i_2 sẽ lưu các 2-subgraph thường xuyên của các đồ thị $g_i \in$ GD. Cùng với đó chúng tôi cũng sẽ dùng C^i_2 để lưu toàn bộ ứng viên 2-subgraph của đồ thị $g_i \in$ GD. Tiếp theo từ FS^i_3 trở đi chúng ta chỉ lưu các tập đồ thị con thường xuyên đóng nên FS^i_3 bị thay thế bằng CS^i_3. Tại mỗi bước khai phá đồ thị con thường xuyên đóng mức k (k >= 2) ta sẽ kết hợp các đồ thị nằm trong FS^i_2 và CS^i_{k-1}để sinh ra các ứng viên nằm trong C^i_k, mỗi thành viên nằm trong C^i_k khi sinh ra sẽ được đưa vào một danh sách liên kết được sắp xếp sẵn tạo thành một danh sách liên kết được sắp thứ tự theo CAM. Khi cần xác định một ứng viên trong C^i_k có phải là đồ thị con thường xuyên hay không thì chúng ta áp dụng chiến thuật tìm kiếm nhị phân được coi là tối ưu nhất hiện nay để tìm sự xuất hiện của nó trong các C^j_k khác. Rõ ràng, ta thấy ở đây mỗi đồ thị con ứng viên sẽ được tìm bằng tìm kiếm nhị phân. Giả sử mỗi đồ thị g_i có $m = 2^n$ đồ thị con ứng viên mức k thì việc tìm kiếm một đồ thị con trong nó là $O(\log_2 2^n)$ nên dù m có lớn đến đâu thì thuật toán xác định đồ thị con đẳng cấu và đếm độ hỗ trợ của một đồ thị con ứng viên vẫn thuộc lớp P tức là giải quyết trong thời gian đa thức.

Đánh giá độ phức tạp thuật toán PSI-CFSM. Dòng 1 xây dựng danh sách liên kết được sắp thứ tự theo CAM các đồ thị 2-subgraph tạo thành C^i_2 cho mỗi đồ thị g_i. 2-subgraph là 2 đỉnh nên chỉ có duy nhất 1 cạnh bởi chúng ta chỉ quan tâm đến các đồ thị liên thông. Do vậy bước này sẽ chỉ là xác định tất cả các cạnh của tất cả các đồ thị g_i trong GD. Dòng 2 đến 4 là đối với mỗi đồ thị con u trong C^i_2 của một đồ thị g_i ta sẽ so sánh với các đồ thị v của C^j_2 theo tìm kiếm nhị phân thì thời gian tính toán tồi nhất là $O(\log_2 |C^j_2|)$ và nếu tìm thấy thì ta sẽ tăng độ hỗ trợ của u và v lên 1 và đánh dấu là đã xét để lần sau chúng ta không tìm đến nó nữa. Như vậy bước 2 đến 4 sẽ tính toán trong $O(n \log_2 \max(|C^i_2|))$ với n là số lượng các đồ thị trong GD. Từ dòng 6 đến dòng 13 là một vòng lặp mà nếu vẫn sinh ra ứng viên mức k thì vẫn tiếp tục. Nghĩa là vòng lặp này sẽ chạy tối đa là $\max(|V_{gi}|)$ bước, mỗi bước của vòng lặp chính là tìm tập đồ thị con thường xuyên đóng ở mức k. Trong vòng lặp ta thực hiện công việc tương tự như với 2-subgraph là xây dựngC^i_k là một danh sách liên kết được sắp xếp theo thứ tự của CAM, việc xây dựng danh sách liên kết này cũng chỉ tính toán trong thời gian tồi nhất là $O(\log_2 C^k_{|Vgi|})$ và vẫn là đa thức. Các công việc tiếp theo ở mức k là tìm kiếm các đồ thị ứng viên nào thỏa mãn là đồ thị con thường xuyên theo tìm kiếm nhị phân cũng chỉ thực hiện trong thời gian tồi nhất là $O(n \log_2 \max(|V_{gi}|))$ và cũng thực hiện trong thời gian đa thức. Dòng thứ 10, kiểm tra $v \in CS^i_{k-1}$ có tồn tại đồ thị nào có độ hỗ trợ bằng với độ hỗ trợ của đồ thị con đang xét u hay không thì từ đồ thị u ta phải xây dựng các đồ thị con mức (k-1) của u và xác định tất cả các con này có con nào trùng với v thì ta loại v khỏi CS^i_{k-1}. Vậy độ phức tạp tính toán ở dòng 10 là $O(m \log_2 |CS^i_{k-1}|)$ với m là số đồ thị con mức (k-1) được sinh ra của u. Mà số đồ thị con mức (k-1) của u chính là lần lượt bỏ đi các đỉnh của u cùng với cạnh gắn với nó ta sẽ được tập S^u_{k-1} và lực lượng lớn nhất của $\max(|S^u_{k-1}|) = (k-1)*(k-2)$ nên số đồ thị con mức (k-1) của u là $NS_u = |V_u|*(k-1)*(k-2)$. Vậy tổng hợp lại ta sẽ có độ phức tạp tính toán của PSI-CFSM là

$$O((max(|V_{g_i}| \times (log_2 C^k_{|V_{g_i}|} \times n \times log_2 max(|V_{g_i}|) \times \sum_{i=1}^{n} \sum_{k=3}^{n} (k-1) \times (k-2) \times log_2 |CS^i_{k-1}|)).$$

VII. KẾT LUẬN

Với kết quả xây dựng được thuật toán khai phá đồ thị con thường xuyên đóng với xác định đồ thị con đẳng cấu thực hiện trong thời gian đa thức mang lại một ý nghĩa lớn trong việc khai phá dữ liệu nói chung và khai phá đồ thị nói riêng. Tiếp theo bài báo này, chúng tôi sẽ thực hiện thử nghiệm thuật toán để chứng minh tính hiệu quả của thuật toán mới được đề xuất.

VIII. LỜI CẢM ƠN

Chúng tôi xin gửi lời cảm ơn tới đề tài CS16.17 "Nghiên cứu một số phương pháp khai phá quan điểm và ứng dụng vào bài toán tổng hợp ý kiến theo tính năng sản phẩm của người tiêu dùng Việt Nam" - Viện Công nghệ Thông tin - Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã tạo điều kiện, trợ giúp một phần kinh phí để hoàn thiện bài báo này.

TÀI LIỆU THAM KHẢO

1. E. Barbu, P. Héroux, S. Adam, and E. Trupin. *Clustering document image using graph summaries*. In Proceedings of the 5th International Conference on Learning and Data Mining, pages 194-202, 2005.
2. C. Chen, X. Yan, P.S. Yu, J. Han, D. Zhang, and X. Gu. *Towards graph containment search and indexing*. In Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07), pages 926-937, 2007.
3. Y. Chi, Y. Yang, Y. Xia, and R.R. Muntz. 2004. *HybridTreeMiner: An Efficient Algorithm for Mining Frequent Rooted Trees and Trees using Canonical Forms*, In Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 11–20.
4. M.J. Chung. *$O(n^{2.5})$ time algorithm for the subgraph homomorphism problem on trees*. Journal of Algorithms, 13:106-112, 1987.
5. D.J. Cook and L.B. Holder. *Substructure discovery using minimum description length and background knowledge*. Journal of Artificial Intelligence Research, 1:231-255, 1994.
6. M. Deshpande, M. Kuramochi, and G. Karypis. *Frequent substructure based approaches for classifying chemical compounds*. IEEE Transactions on Knowledge and Data Engineering, 17(8):1036-1050, 2005.
7. M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
8. J.E. Hopcroft and R.E. Tarjan. *Isomorphism of planar graphs*. In R.E. Miller and J.W. Thatcher, editors, Complexity of Computer Computations, pages 131-152. Plenum Press, 1972.
9. J. Huan, W. Wang and J. Prins. *Effcient mining of frequent subgraph in the presence of isomorphism*. In Proceedings of the 2003 International Conference on Data Mining (ICDM'03), pages 549-552, 2003.
10. J. Huan, W. Wang, A. Washington, J. Prins, R. Shah, and A. Tropsha. *Accurate classification of protein structural families based on coherent subgraph analysis*. In Proceedings of Pacific Symposium on Biocomputing, pages 411-422, 2004.
11. Inokuchi, T.Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructure from graph data. In Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, pages 13-23, 2000.
12. Inokuchi, T. Washio, K. Nishimura, and H. Motoda. A fast algorithm for mining frequent connected subgraphs. Research Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.
13. S.W. Lu, Y. Ren, and C.Y. Suen. *Hierarchical attributed graph representation and recognition of handwritten chinese characters*. Pattern Recognition, 24:617-632, 1991.
14. D.W. Matula. *Subtree isomorphism in $O(n^{5/2})$*. Annals of Discrete Mathematics, 2: 91-106, 1978.
15. B.D. McKay. *Practical graph isomorphism*. Congress Numerantium, 30:45-87, 1981.
16. M. Kuramochi and G. Karypis. 2001. *Frequent Subgraph Discovery*, In Proceedings of International Conference on Data Mining, 313–320.
17. Pearce, T. Caelli, and W.F. Bischof. Rule-graphs for graph matching in pattern recognition. Pattern Recognition, 27(9):1231-1246, 1994.
18. D.C. Schmidt and L.E. Druffel. *A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices*. Journal of the ACM, 23(3):433-445, 1976.
19. R. Shamir and D. Tsur. 1999. *Faster Subtree Isomorphism*, Journal of Algorithms 33(2), 267–280.
20. D. Shasha, J.T.L. Wang, and R. Giugno. *Algorithms and applications of tree and graph searching*. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles on Database Systems (PODS'02), pages 39-52, 2002.
21. J.R. Ullmann. *An algorithm for subgraph isomorphism*. Journal of the ACM, 23(1): 31-42, 1976.
22. X. Yan and J.W. Han. *gSpan: Graph-based substructure pattern mining*. In Proceedings of the 2002 International Conference on Data Mining, page 721, 2002.
23. X. Yan and J. Han. *Close graph: Mining closed frequent graph patterns*. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 286-295, 2003.
24. X. Yan, P.S. Yu, and J. Han. *Graph indexing: A frequent structure-based approach*. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04), pages 335-346, 2004.
25. X. Yan, P.S. Yu, and J. Han. *Sub-structure similarity search in graph databases*. In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD'05), pages 766-777, 2005.
26. X. Yan, F. Zhu, J. Han, and P.S. Yu. *Searching substructures with superimposed distance*. In Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), page 88, 2006.

SOME PROBLEMS ON CLOSED FREQUENT SUBGRAPH MINING

Hoang Minh Quang, Vu Duc Thi, Pham Quoc Hung

ABSTRACT— *Frequent patterns mining is an important problem is more likely into practical applications. The diversity of data structures such as chemical structures of the drug ingredients, cell gene structures, animal protein structures and other structures, causes many difficulties in frequent itemsets mining. The good news is that these data can be represented by many structured data formats known as graphs, trees or lattices. Therefore, the study of frequent subgraph mining is significantly useful in the medical field. In this paper, we review and evaluate some current frequent subgraph mining algorithms and recommend some improvememt in the implementation of frequent subgraph mining in order to increase the efficiency of frequent subgraphs mining algorithms, especially closed frequent subgraphs mining algorithms.*

Keywords— *Data mining, frequent subgraph, graph mining, structured data, closed frequent subgraph, complexity computation.*