# KHAI PHÁ CÂY CON THƯỜNG XUYÊN TRÊN CƠ SỞ DỮ LIỆU WEBLOGS

**Hoàng Minh Quang[1], Vũ Đức Thi[2], Kiều Thu Thủy[3], Đào Văn Tuyết[4], Phan Trung Kiên[5]**

[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]Học viện Bưu chính Viễn thông
[4]Viện Cơ học và Tin học ứng dụng
[5]Trường Đại học Tây Bắc
*[1]hoangquang@ioit.ac.vn, [2]vdthi@vnu.edu.vn, [4]tuyetdv@gmail.com, [5]kienptr@gmail.com*

***TÓM TẮT*** *- Hầu hết các công ty, tổ chức hiện nay đều mong muốn thu thập và trích xuất dữ liệu về mối quan tâm của người sử dụng. Dữ liệu có cấu trúc dạng weblogs có thể biểu diễn dưới dạng đồ thị và cây. Khai phá dữ liệu cây con thường xuyên trên cơ sở dữ liệu weblogs là tìm tất cả các cây con trong rừng cây weblogs mà có số lần xuất hiện lớn hơn một ngưỡng cho trước. Đây là một bài toán có độ phức tạp tính toán hàm mũ và có rất nhiều nhà khoa học nghiên cứu về vấn đề này. Trong bài báo này, chúng tôi đề xuất một phương pháp hiệu quả khai phá cây con thường xuyên trên cơ sở dữ liệu weblogs với việc tối ưu hóa vấn đề phát hiện các cây con đẳng cấu và thuật toán tìm kiếm theo chiều sâu để giảm thời gian và giảm không gian bộ nhớ trong quá trình tính toán.*

***Từ khóa*** *- khai phá dữ liệu, cây con thường xuyên, khai phá đồ thị, weblogs, dữ liệu có cấu trúc*

## I. GIỚI THIỆU

Mục tiêu của khai phá dữ liệu là trích xuất thông tin hữu ích từ dữ liệu [5, 10]. Dữ liệu có thể có nhiều dạng: véctơ, bảng, hình ảnh, văn bản,... và dữ liệu cũng được biểu diễn khác nhau. Dữ liệu có cấu trúc và dữ liệu bán cấu trúc phù hợp cho biểu diễn đồ thị chẳng hạn cấu trúc protein-protein biểu diễn dưới dạng đồ thị với đỉnh là các gens và các cạnh là mối quan hệ các gens. Khai phá dữ liệu có cấu trúc (tree, graph, và lattices) ngày càng trở nên quan trọng trong mô hình hóa các cấu trúc tinh vi phức tạp và sự tương tác của nó, với ứng dụng rộng rãi trong nhiều lĩnh vực của cuộc sống như tin hóa học, tin sinh học, tầm nhìn máy tính, chỉ mục video, phục hồi văn bản, và phân tích Web,v.v…[10]. Đặc biệt, với xu hướng phát triển của dữ liệu lớn (Big Data), khai phá dữ liệu có cấu trúc càng trở thành một nhu cầu thiết yếu. Có thể dễ dàng biểu diễn dữ liệu có cấu trúc dưới dạng đồ thị đặc biệt hơn là cây.

Bản chất của cấu trúc sử dụng web là một dạng dữ liệu có cấu trúc dạng đồ thị. Tuy nhiên, vấn đề khai phá đồ thị con thường xuyên gặp vấn đề về độ phức tạp thời gian tính toán trong việc phát hiện đẳng cấu. Chính vì vậy, khai phá dữ liệu cấu trúc sử dụng web được đưa về khai phá cây con thường xuyên nhằm làm giảm thời gian phát hiện các đồ thị con đẳng cấu. Một trong những thuật toán hiệu quả nhất về phát hiện đồ thị con đẳng cấu được R.Shamir và D.Tsur đề xuất năm 1999 [29]. Hiện nay đã có rất nhiều các phương pháp khai phá cây con thường xuyên tuy nhiên theo tác giả khảo sát thì vẫn chưa có thuật toán nào áp dụng phương pháp phát hiện đẳng cấu của R.Shamir và D.Tsur trong quá trình khai phá dữ liệu. Trong bài báo này, chúng tôi áp dụng phương pháp phát hiện cây con đẳng cấu sử dụng phương pháp của R.Shamir và D.Tsur với điều kiện ràng buộc trên cây có gắn nhãn nhằm mục tiêu làm giảm hơn nữa thời gian phát hiện đồ thị con đẳng cấu.

Weblogs là một cấu trúc ghi lại các trang trong một phiên người dùng truy cập. Ứng với mỗi phiên truy cập, cấu trúc weblogs sẽ được lưu dưới dạng một cây có gốc là trang bắt đầu vào thăm của một website. Từ đó, mỗi nút của cây là một trang khác trong website mà người dùng đã xem trong một phiên sử dụng. Việc người dùng xem các trang tạo thành một đồ thị, tuy nhiên để giảm thời gian khai phá dữ liệu weblogs thì các log sẽ được lưu dưới dạng cây và biểu diễn bằng một mã chuỗi chỉ ra đường dẫn từ trang chủ đến các trang con mà người dùng vào xem. Ví dụ về mã chuỗi biểu diễn cho một cây trong cơ sở dữ liệu weblogs: (2 3 4 -1 1 -1 3 -1 -1) biểu diễn một cây gốc là đỉnh 2, ký hiệu -1 thể hiện việc quay lại cha của đỉnh hiện tại theo phương pháp duyệt cây theo độ sâu preorder. Các cấu trúc weblogs khác nhau đều có thể quy về cách biểu diễn theo string như trên.

Một số tác giả sử dụng cơ sở dữ liệu cây weblogs CS-LOG [25, 26, 28]. Các thuật toán của các nhóm tác giả này đều có những điểm mạnh và điểm yếu vì vậy việc so sánh hiệu suất thực hiện chỉ mang tính chất tương đối. Các thuật toán đưa ra các mẫu cây con thường xuyên cũng khác nhau do sử dụng các kiểu cấu trúc cây khác nhau mặc dù cùng dùng chung cơ sở dữ liệu rừng cây CS-LOG được biểu diễn dưới dạng mã chuỗi theo trật tự duyệt trước theo độ sâu. Trong bài báo này, chúng tôi lấy CS-LOG làm cơ sở dữ liệu thử nghiệm thuật toán, các cơ sở dữ liệu weblogs khác có thể sử dụng trong thuật toán với điều kiện đưa về dạng mã chuỗi theo trật tự duyệt trước theo độ sâu (preorder depth first search) giống với cơ sở dữ liệu weblogs CS-LOG được cung cấp, thuật toán tìm ra các cây con thường xuyên chính xác (induced subtree frequent mining).

## II. MỘT SỐ ĐỊNH NGHĨA

Chúng ta biểu diễn dữ liệu weblogs bằng một dạng cây gắn nhãn không có thứ tự. Dựa theo biểu diễn cây của T.Asai và cộng sự. Một số định nghĩa được phát biểu lại trong thuật toán của T.Asai [26].

Cây gắn nhãn không có thứ tự là một đồ thị không có chu trình T = (V, E, r, label) với một đỉnh riêng biệt r gọi là đỉnh gốc thỏa mãn các điều kiện: V là tập hợp các đỉnh, E ⊆ V x V là tập hợp các cạnh, label là một ánh xạ label: V→L được gọi là hàm gắn nhãn. Với mọi đỉnh v ∈ V có duy nhất một đường UP(v) = ($v_0$ = r, $v_1$, ..., $v_d$) (d ≥ 0) từ đỉnh gốc đến đỉnh v. Độ sâu của đỉnh v ký hiệu là deg(v) = d.

Cho hai đỉnh u và v. Nếu (u, v) ∈ E ta nói rằng u là đỉnh cha của v, v là đỉnh con của u. Nếu có đường từ u đến v thì u được gọi là tiền bối của v, v gọi là hậu bối của u. Một lá là một đỉnh không có đỉnh con. Với mỗi đỉnh v ta gọi T(v) là một cây con của cây T có gốc là v. Kích thước của T là số đỉnh trong nó |T| = |V|.

Ta sử dụng một cây gắn nhãn có thứ tự để biểu diễn một cây gắn nhãn không có thứ tự. Cây gắn nhãn có thứ tự là cây gắn nhãn không có thứ tự được với thứ tự từ trái qua phải theo con của mỗi đỉnh. Một cây gắn nhãn có thứ tự là một bộ 5 T = (V, E, r, label, B) với V, E, r, label là tập các đỉnh, tập cạnh, đỉnh gốc, hàm gắn nhãn của tập đỉnh V. Một quan hệ hai ngôi B⊆VxV là thứ tự từ trái qua phải của các con của một đỉnh. RMB(T) = ($r_0$, ..., $r_c$) (c ≥ 0) là đường từ đỉnh gốc r tới lá bên phải nhất của cây T. Đường RMB(T) gọi là nhánh phải nhất của cây T, rml(T) = k là lá bên trái nhất của cây T.

Cho hai cây không có thứ tự T = ($V_T$, $E_T$, $r_T$, $label_T$) và D = ($V_D$, $E_D$, $r_D$, $label_D$). Ta gọi cây T là cây mẫu đầu vào, cây D là cây dữ liệu, mẫu T xảy ra trong D nếu có một ánh xạ $\mathcal{I}$: $V_T$→$V_D$thỏa mãn 3 điều kiện với mọi đỉnh x, y ∈ $V_T$.

(1) $\mathcal{I}$ là ánh xạ một – một, với mọi x ≠ y thì $\mathcal{I}(x) \neq \mathcal{I}(y)$

(2) $\mathcal{I}$ bảo toàn quan hệ đỉnh cha, (x, y)∈ $E_T$ nếu và chỉ nếu ($\mathcal{I}(x)$,$\mathcal{I}(y)$) ∈ $E_D$.

(3) $\mathcal{I}$ bảo toàn nhãn, $label_T(x) = label_D(\mathcal{I}(x))$.

Ánh xạ $\mathcal{I}$ gọi là phù hợp từ cây T vào cây D.

**Định nghĩa (2.1)**. Cho số nguyên dương k ≥ 1, cây T = (V, E, r, label) là một k-mẫu không có thứ tự. Giả sử ta có sự phù hợp$\mathcal{I}$: $V_T$→$V_D$∈ $M^D(T)$ từ cây T vào D = {$D_i$}. Có bốn kiểu xảy ra của T trong D:

(1) Xảy ra toàn thể của T (total occurrence) là một k-bộ $T_{OC}(\mathcal{I}) = \langle \mathcal{I}(1), ..., \mathcal{I}(k) \rangle \in (V_D)^k$.

(2) Xảy ra nhúng của T (embedding occurrence) là tập $E_{OC}(\mathcal{I}) = \{\mathcal{I}(1), ..., \mathcal{I}(k)\} \subseteq V_D$.

(3) Xảy ra gốc của T (rooted occurrence) $R_{OC}(\mathcal{I}) = \mathcal{I}(1) \in V_D$.

(4) Xảy ra tài liệu của T (document occurrence) là một chỉ số $D_{OC}(\mathcal{I}) = i$ sao cho $E_{OC}(\mathcal{I}) \subseteq \boldsymbol{V_{D_i}}$ với 1 ≤ i ≤ |D|.

**Định lý (2.1)**. $T_{OC} \geq E_{OC} \geq R_{OC} \geq D_{OC}$.

**Định lý (2.2)**. D là một cơ sở dữ liệu cây và T là một mẫu cây, $\frac{|\boldsymbol{TO^D(T)}|}{|\boldsymbol{EO^D(T)}|} = \boldsymbol{k^{O(k)}}$ và $\frac{|\boldsymbol{EO^D(T)}|}{|\boldsymbol{RO^D(T)}|} = \boldsymbol{O(n^k)}$

**Khai phá cây con thường xuyên**

Cho một ngưỡng minfreq, một lớp C các cây, một quan hệ thứ tự các cây con P ≼ T giữa hai cây P, T ∈ C, cho một tập hữu hạn dữ liệu cây D ⊆ C, vấn đề khai phá cây thường xuyên là tìm tất cả các cây P ⊆ C mà không có hai cây bất kỳ nào trong P đẳng cấu với nhau và cho tất cả p ∈ P: freq(P,D) = $\sum_{T \in D} d(P,T) \geq$ minfreq với d là một hàm phản đơn điệu ∀T ∈ C: d(P',T)≥d(P,T) nếu P' ≼ P.

Cho một cơ sở dữ liệu rừng cây D và T là một cây trong cơ sở dữ liệu rừng cây D. Một cây con P có quan hệ P ≼ T được gọi là P xảy ra trong T hay P là cây con đẳng cấu của T và hàm d được xác định:

$$\boldsymbol{d(P,T)} = \begin{cases} \boldsymbol{1\ nếu\ P \leq T} \\ \boldsymbol{0\ các\ trường\ hợp\ khác} \end{cases}$$

Mức độ thường xuyên của cây con P là số lần xuất hiện của cây con P trong cơ sở dữ liệu rừng cây D chứa các cây T. Chúng ta mượn thuật ngữ giao tác trong khai phá dữ liệu luật kết hợp gọi là thường xuyên dựa trên giao tác. Mỗi giao tác là một cây T, cơ sở dữ liệu giao tác là cơ sở dữ liệu rừng cây D. Gọi độ hỗ trợ của cây P trong D là: sup(P, D) = freq(P, D)/|D|. Do tính chất phức tạp của đồ thị và cây nên một cây con P có thể xuất hiện nhiều lần trong một cây T trong khi với khai phá dữ liệu tập mục thường xuyên thì một tập con k-tập mục chỉ xuất hiện một lần trong một giao tác. Do đó hàm freq(P, D) chỉ được tính khi P là cây con của cây T trong cơ sở dữ liệu rừng cây D.

## III. KIỂM TRA ĐẲNG CẤU

Sử dụng phương pháp phát hiện đẳng cấu trong [29] với thay đổi một chút về nhãn của đỉnh và định nghĩa một thứ tự cho tất cả các đỉnh. Từ đó làm tăng hiệu suất phát hiện đẳng cấu trong cây có gắn nhãn. Các thành phần sau được phát biểu lại công trình nghiên cứu của R.Shamir và D.Tsur 1999 [29].

Cây có gốc là một bộ ba G = (V, E, r). Định nghĩa $G_v^r$ là cây con có gốc cha là r và v là một con của r và chỉ xem xét cây có gốc v.$G^r$ và $H^s$ là đẳng cấu nếu có một đẳng cấu giữa G và H dưới ánh xạ từ r tới s.$H^s$ là cây con đẳng cấu với $G^r$ nếu có một cây con $J^r$ của $G^r$ mà đẳng cấu với $H^s$.Tập láng giềng mở là tập tất cả láng giềng của v ký hiệu N(v).Tập láng giềng đóng là tập tất cả láng giềng của v bao gồm cả đỉnh v ký hiệu N[v].Bậc của v trong đồ thị G được định nghĩa là $d_G(v)$. Thuật toán có độ phức tạp là $O(k^{1.5}n)$.

Cho G = (V, E) và H = ($V_H$, $E_H$) với $|V_H| > 1$. Chọn một đỉnh r là gốc của G. Cần tìm một cây con đẳng cấu $H_w^u$ của $G_v^r$ với u là đỉnh của H và v là đỉnh của G và w là láng giềng của u.

**Bổ đề (1)** Cho mọi đỉnh v của $G^r$, đỉnh u trong H, và một đỉnh w thuộc tập láng giềng đóng của u, ta có $H_u^w$ là cây con đẳng cấu của $G_v^r$ nếu và chỉ nếu mọi đỉnh con u' của u trong $H_u^w$ có một đỉnh con phân biệt v' của v mà $H_{u'}^{u}$ là cây con đẳng cấu của $G_{v'}^{r}$.

Chúng ta lưu giữ thông tin này trong các tập S(v, u) = {đỉnh w thuộc láng giềng đóng của u sao cho: $H_u^w$ là cây con đẳng cấu của $G_v^r$} với mọi đỉnh v của G và cho mọi đỉnh u của H.

**Định lý (3.1).** Thuật toán Subtree-Isomorphism giải quyết vấn đề cây con đẳng cấu trong thời gian $O(k^{1.5}n)$ và không gian O(kn).

Cho B = (X, Y, E) là một đồ thị song phương với X = ($x_1$, …, $x_s$), Y = ($y_1$, …, $y_t$), $s \le t + 1$. Cho $X_0 = X$ và $X_i = X - \{x_i\}$ với $1 \le i \le s$, $m_i$ là sự phù hợp tối đa giữa $X_i$ và Y. M là sự phù hợp tối đa của B, xác định đồ thị có hướng $B_M = (X \cup Y, E_M)$ với

$E_M = \{(x, y): xy \in E - M, x \in X, y \in Y\} \cup \{(y, x): xy \in M, x \in X, y \in Y\}$

**Bổ đề (3.1)**. Với mọi sự phù hợp tối đa M của B và mọi đỉnh $x_i \in X$, $x_i$ là điểm quan trọng ($m_i = m_0 - 1$) nếu và chỉ nếu $x_i$ được phù hợp trong M, không có đường trực tiếp nào trong $B_M$ từ một đỉnh thuộc $X_M$ tới $x_i$.

**Hệ quả (3.1)**. Cho một phù hợp tối đa M của B, ta có thể tính giá trị của $m_i$ với $0 \le i \le s$ với độ phức tạp thời gian O(st).

**Định lý (3.2)**. Vấn đề tìm một phù hợp tối đa trong B có thể giảm độ tính toán từ O(st) tới tìm một phù hợp tối đa trong đồ thị con của B với nhiều nhất $s^2$ đỉnh và cạnh với bậc cao nhất là s.

**Định lý (3.3)**. Thuật toán Improved-Subtree-Isomorphism tìm một phù hợp tối đa trong B(v,u) với độ phức tạp thời gian $O(st + ts^{0.5}D(u))$.

**Định lý (3.4)**. $g(n) = 2^{\ominus(n)}$

**Định lý (3.5)**. $f(n) = \ominus(n/\log n)$

**Định lý (3.6)**. Với mọi đỉnh khó $y_j$, $d(y_j) \le k^{\epsilon} + l_j$ và $D(y_j) \le k^{\epsilon} + f(l_j)$

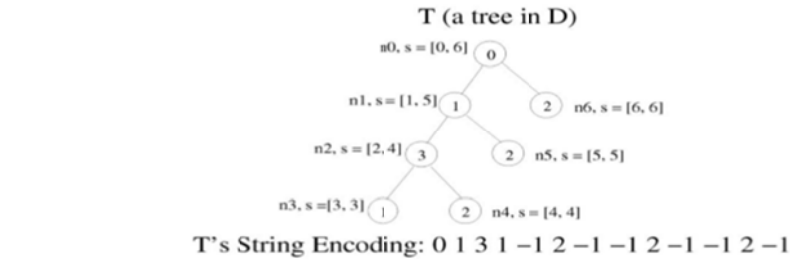
**Định lý (3.7)**. $\sum_{j=1}^{k} d(y_j)^{0.5}D(y_j) = O(k^{1.5}/\log k)$.

**Định lý (3.8)**. Thuật toán Improved-Subtree-Isomorphism giải quyết vấn đề cây con đẳng cấu với độ phức tạp thời gian $O((k^{1.5}/\log k)n)$.

Thuật toán phát hiện cây con đẳng cấu của R.Shamir và D.Tsur 1999 [29] xác định cây con đẳng cấu trên cây không gắn nhãn và không có thứ tự. Bài toán tìm cây con thường xuyên trên cơ sở dữ liệu weblogs có điểm khác biệt là các cây trong cơ sở dữ liệu weblogs đều có thể gắn nhãn và hơn nữa còn có thể biểu diễn dưới dạng cây gắn nhãn có thứ tự. Chính vì vậy khi áp dụng thuật toán Improved-Subtree-Isomorphism vào cây gắn nhãn có gốc có thứ tự sẽ làm giảm hơn nữa độ phức tạp tính toán tùy thuộc vàocách thức gắn nhãn trong cây. Nếu cây có các nhãn riêng biệt không trùng lặp thì việc đánh thứ tự chỉ việc gắn cho các nhãn, nếu cây có các nhãn mà có trùng lặp thì ngoài thứ tự các nhãn còn có thứ tự từ trái sang phải trong cây.

## IV. SINH TẬP ỨNG VIÊN

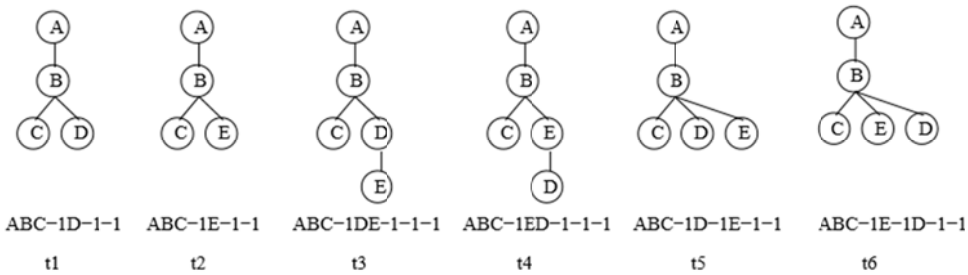
Có hai bước quan trọng trong việc tìm tất cả các cây con thường xuyên trong cơ sở dữ liệu các cây D. Bước đầu tiên là sinh tập cây con ứng viên để kiểm tra mức độ thường xuyên. Tập ứng viên được sinh ra phải không dư thừa, nghĩa là mỗi cây con ứng viên chỉ được sinh ra nhiều nhất một lần. Cây được biểu diễn dưới dạng mã chuỗi ký tự, cây T được sinh ra theo cách: thêm một đỉnh được gắn nhãn vào cây T theo phương pháp duyệt trước cây theo độ sâu (preorder depth first search) và thêm một ký tự duy nhất -1 không thuộc vào tập nhãn của cây (hoặc # hoặc $, …) khi xảy ra một quay lui từ đỉnh con lên đỉnh cha trong cây T [26].




**Hình 1**. Biểu diễn mã chuỗi của cây duyệt trước theo độ sâu.

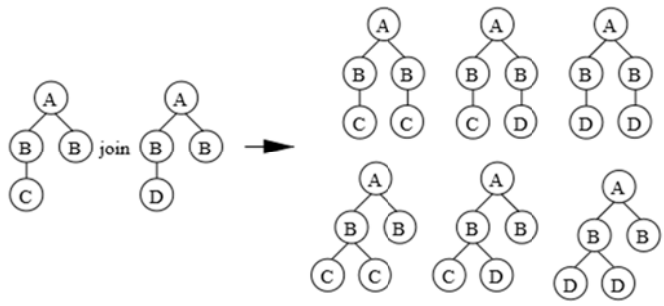
Định dạng này có ưu điểm có thể biểu diễn các cây với tùy ý số lượng đỉnh con của mỗi đỉnh. Theo đó, mỗi nhánh của cây phải được duyệt theo cả hai hướng tiến lên và quay lui. Bộ nhớ để lưu trữ cây theo biểu diễn mã chuỗi là 2m + 1 = 2n – 1 với m là số lượng cạnh và n là số lượng đỉnh có trong cây T. Ký hiệu l(T) mô tả chuỗi các nhãn của cây T theo phương pháp duyệt trước theo độ sâu nhưng không chứa ký tự biểu diễn quay lui (với cây 0 1 3 1 -1 2 -1 -1 2 -1 -1 2 -1 thì l(T) = 0131222).

Trong vấn đề sinh tập ứng viên trong bài toán tìm mẫu thường xuyên dựa vào tính chất phản đơn điệu để tăng hiệu suất sinh tập ứng viên. Trong khai phá tập mục thường xuyên của khai phá luật kết hợp thì tính chất phản đơn điệu là tập con của tập thường xuyên là tập thường xuyên, tập cha của tập không thường xuyên là tập không thường xuyên. Đối với cây ta cũng có tính chất tương tự là cây con của cây thường xuyên là cây thường xuyên, cây cha của cây không thường xuyên là cây không thường xuyên. Dựa vào tính chất này ta chỉ cần sinh các cây con ứng viên mức (k+1) có (k+1) đỉnh từ cây con k đỉnh nếu cây con k đỉnh là thường xuyên, nếu cây con k đỉnh là không thường xuyên thì không cần sinh cây con (k+1) đỉnh. Từ đó giảm số lượng cây con ứng viên trong quá trình tìm tất cả các cây con thường xuyên trong cơ sở dữ liệu cây D.




**Hình 2**. Kết hợp hai cây t1 và t2 được các cây t3, t4, t5 t6.

Chi [49] đề xuất một thuật toán được gọi là HybridTreeMiner sử dụng kết hợp cả phương pháp duyệt theo chiều rộng (breadth-first traversal) và theo chiều sâu (depth-first traversal) để sinh tập ứng viên dùng hai phương pháp là kết hợp (join) và mở rộng (extension). Để đảm bảo mỗi ứng viên chỉ được sinh ra một lần, Chi [25] sử dụng một dạng biểu diễn chuẩn theo chiều rộng (breadth-first canonical form) dựa trên việc duyệt theo mức và thứ tự trong cây. Cho một cây có gốc không có thứ tự biểu diễn dạng chuẩn mức (k+1). Nếu có hai đỉnh lá trong cùng mức của cây T thì mã theo chiều rộng có thể lấy bằng cách bỏ đi một trong hai đỉnh lá và chia sẻ tiền tố chung mức (k-1), ví dụ cây t1 và t2 trong hình 2 chia sẻ tiền tố chung bằng cách bỏ đi đỉnh cuối. Trong trường hợp đỉnh cuối của cây con P là con của hai đỉnh cuối của P (ví dụ t3 và t4 trong hình 2) thì ta bỏ đi 2 đỉnh cuối và chia sẻ tiền tố chung.




**Hình 3**. Cây con tự đẳng cấu.

Phương pháp sinh tập cây con ứng viên từ hai cây con thường xuyên mức k chia sẻ tiền tố chung mức (k-1). Mỗi cặp cây con sẽ được kết hợp với nhau để sinh các cây con ứng viên mức (k+1). Các cây không thể thực hiện kết hợp thì sẽ được thực hiện mở rộng. Khi kết hợp hai cây con thường xuyên mức k chia sẻ tiền tố chung mức (k-1) để được cây con mức (k+1) sẽ phải thêm vào một vấn đề về tự đẳng cấu cây con (tree automorphism), nếu không có tự

đẳng cấu thì việc sinh ứng viên sẽ phức tạp hơn nhiều. Ví dụ hình 3 thể hiện khi kết hợp hai cây thường xuyên mức k chia sẻ tiền tố chung mức (k-1) mà không có tự đẳng cấu sẽ có nhiều cây con mức (k+1).

## V. THUẬT TOÁN

Thuật toán khai phá cây con thường xuyên tương tự như thuật toán khai phá tập mục thường xuyên trong khai phá dữ liệu luật kết hợp. Thông thường có hai phương pháp tiếp cận: phương pháp tiếp cận dựa trên thuật toán Apriori do Argawal đề xuất 1994 [1], phương pháp tiếp cận dựa trên phát triển mẫu dùng FP-Growth do Han đề xuất [30].

Thuật toán tìm cây con thường xuyên theo phương pháp tiếp cận Apriori:

**Thuật toán khai phá cây con thường xuyên theo cách tiếp cận Apriori**.

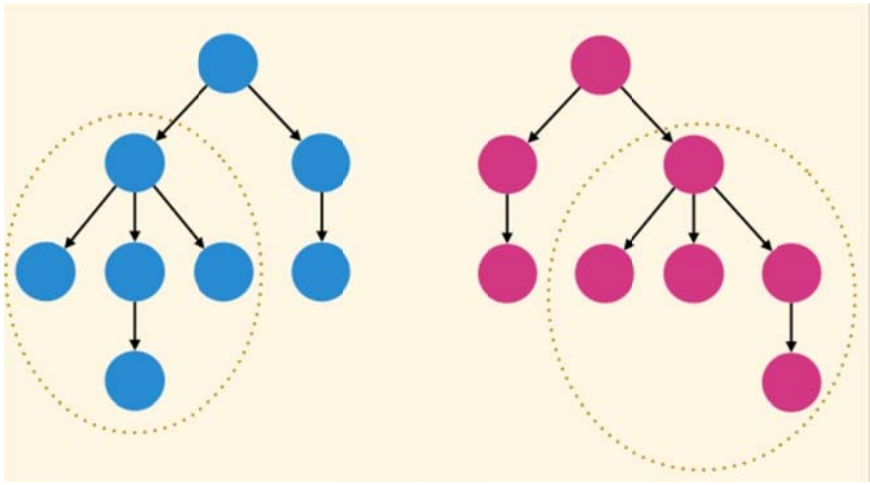
*Input: Cơ sở dữ liệu cây trong rừng cây D, ngưỡng độ hỗ trợ tối thiểu σ*
*Output: $F_1$, …, $F_k$ là các tập cây con thường xuyên với đỉnh tương ứng từ 1 đến k.*
*1. $F_1$ ← phát hiện tất cả các cây con thường xuyên có 1 đỉnh từ rừng cây D*
*2. k ← 2*
*3. while $F_{k-1} \neq \phi$ do*
*4. $F_k \leftarrow \phi$*
*5. $C_k \leftarrow$ candidate_gen($F_{k-1}$)*
*6. foreach t ∈ $C_k$ do*
*7. t.count ← 0*
*8. foreach $D_i \in D$ do*
*9. if Subtree-Isomorphism(t, $D_i$) then*
*10. t.count ← t.count + 1*
*11. end*
*12. end*
*13. if ((t.count ≥ σ) and (t không thuộc $F_k$)) then*
*14. $F_k \leftarrow F_k \cup t$*
*15. end*
*16. end*
*17. k ← k + 1*
*18.end*

Trong bài báo này, chúng tôi cải tiến thuật toán tìm cây con theo phương pháp tiếp cận Apriori sử dụng một danh sách các cây con thường xuyên cùng với sử dụng phương pháp phát hiện đẳng cấu của R.Shamir và Tsur 1999 [29] đồng thời thêm vào điều kiện gắn nhãn cho đỉnh và đưa thứ tự các đỉnh vào trong cây để tăng hiệu suất tìm kiếm tất cả cây con thường xuyên.

Thuật toán kiểm tra cây con đẳng cấu

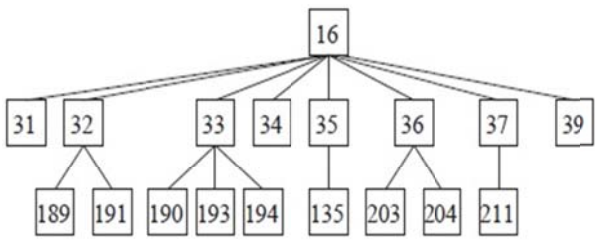
**Thuật toán kiểm tra cây con đẳng cấu**

*Input: cây có gốc G với đỉnh gốc là r, cây có gốc H*
*Output: H có phải là cây con đẳng cấu của G hay không?*
*1. Chọn một đỉnh r của G làm gốc G.*
*2. Với mọi u ∈ H,v ∈ G đặt S(v,u)← φ.*
*3. For all đỉnh lá v của $G^r$ do*
*4. For all đỉnh lá u của H do S(v,u)← N(u).*
*5. For all đỉnh trong v của $G^r$ theo thứ tự postorder do*
*6. Cho $v_1,...,v_t$ là các con của đỉnh v.*
*7. For all đỉnh u = $u_0$ của H với bậc cao nhất t + 1 do*
*8. Cho $u_1,...,u_s$ là các láng giềng của đỉnh u.*
*9. Xây dựng đồ thị song phương B(v,u) = (X,Y,$E_{vu}$),*
*với X ={$u_1,...,u_s$}, Y = {$v_1,...,v_t$},*
*và $E_{vu}$ ={$u_i v_j$ : u ∈ S($v_j$,$u_i$)}.*
*Xác định $X_0$ = X và $X_i$ = X −{$u_i$}.*
*10. For all 0≤ i ≤ s do tính $m_i$ là phù hợp tối đa giữa $X_i$ và Y.*
*11. S(v,u)←{$u_i$ : $m_i$ = |$X_i$|, 0≤ i ≤ s}.*
*12. If u ∈ S(v,u) then trả lời YES và dừng.*
*13. End for*
*14.End for*
*15.Trả lời NO.*



**Hình 4**. Hình dạng cây con đẳng cấu.

Dữ liệu web logs được biểu diễn dưới dạng đồ thị. Việc tìm đồ thị con đẳng cấu là có độ phức tạp tính toán thuộc lớp NP-đầy đủ [8] nên chúng ta biểu diễn web logs dưới dạng cây để giảm bớt độ phức tạp tính toán. Chúng ta chấp nhận mất một số thông tin có thể coi là chấp nhận được khi biểu diễn web logs dưới dạng cây thay cho đồ thị.



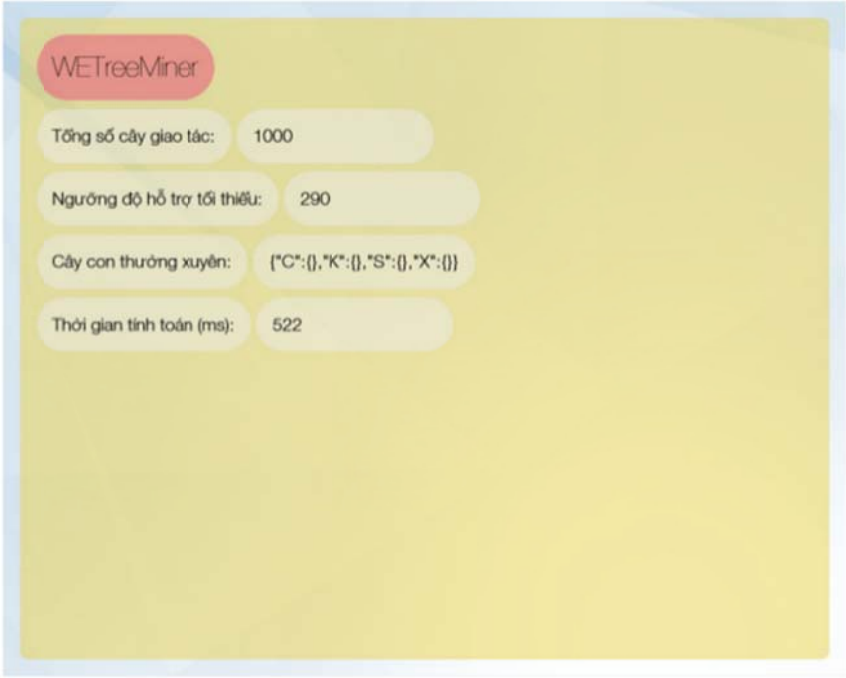

| | |
|---|---|
| 16 | http://dml.cs.ucla.edu/projects/dml_esip/ |
| 31 | http://dml.cs.ucla.edu/projects/dml_esip/Objectives/objectives.html |
| 32 | http://dml.cs.ucla.edu/projects/dml_esip/Status/status.html |
| 33 | http://dml.cs.ucla.edu/projects/dml_esip/Technology/technology.html |
| 34 | http://dml.cs.ucla.edu/projects/dml_esip/Science/science.html |
| 35 | http://dml.cs.ucla.edu/projects/dml_esip/Contacts/contacts.html |
| 36 | http://dml.cs.ucla.edu/projects/dml_esip/Clustering/clustering.html |
| 37 | http://dml.cs.ucla.edu/projects/dml_esip/Links/links.html |
| 39 | http://dml.cs.ucla.edu/projects/dml_esip/index.html |
| 189 | http://dml.cs.ucla.edu/oasis/OASIS_Overview/oasis_overview.html |
| 191 | http://dml.cs.ucla.edu/~weiwang/myresearch.shtml |
| 190 | http://dml.cs.ucla.edu/projects/dml_esip/Technology/Conquest/conquest.html |
| 193 | http://dml.cs.ucla.edu/projects/dml_esip/Technology/VPN/vpn.html |
| 194 | http://dml.cs.ucla.edu/projects/dml_esip/Technology/SEML/seml.html |
| 135 | http://dml.cs.ucla.edu/people/index.html |
| 203 | http://dml.cs.ucla.edu/projects/dml_esip/Clustering/ESP2Net-CaLSIP/esp2net-calsip.html |

**Hình 5**. Cấu trúc dữ liệu cây web logs thực tế.

Từ cấu trúc dữ liệu web logs thực tế, ta chuẩn hóa web logs thành dạng cây mã chuỗi có thứ tự theo phương pháp duyệt trước theo độ sâu (preorder depth first search) như biểu diễn trong Hình 1. Với mỗi nút chúng ta gán một giá trị thuộc vào miền ký hiệu thuộc từ điển. Với mỗi ký hiệu trong từ điển ta gán một thứ tự chẳng hạn $A < B < C < D < \ldots$, rồi từ đó áp dụng thuật toán phát hiện cây con đẳng cấu theo [29]. Phương pháp phát hiện cây con đẳng cấu theo [29] là cây không có thứ tự và không được gán nhãn có độ phức tạp tính toán là $O((k^{1.5}/\log k)n)$ nên khi gán nhãn và thứ tự các nhãn thì độ phức tạp tính toán trong phát hiện cây con đẳng cấu được giảm đi đáng kể. Trong bài báo này, chúng tôi vẫn chưa chứng minh được độ phức tạp tính toán phát hiện cây con đẳng cấu dựa trên [29] mà gán nhãn và thứ tự cho nhãn, chúng tôi vẫn tiếp tục nghiên cứu vấn đề này.

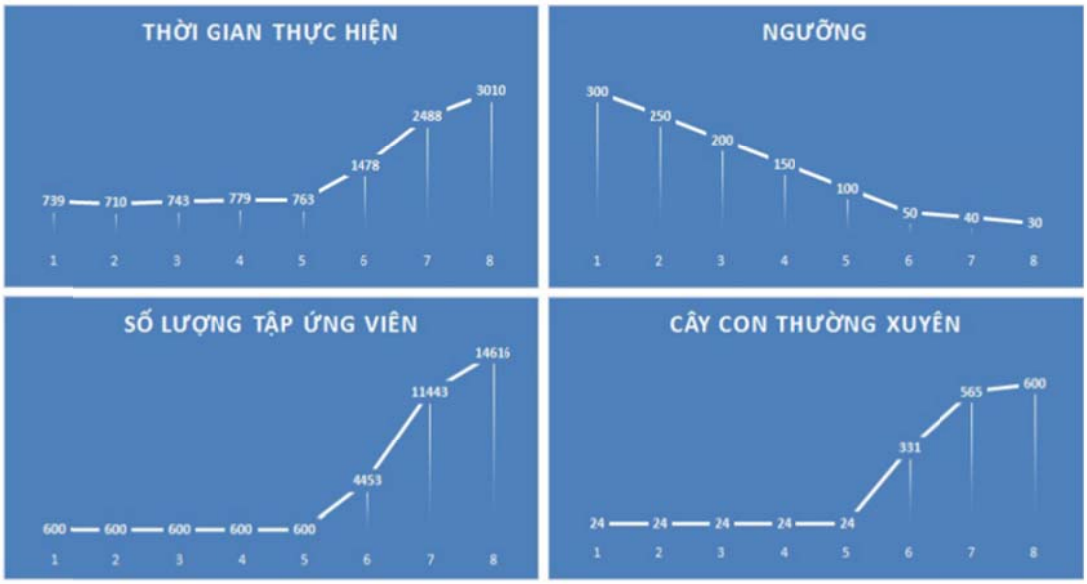
Thuật toán WETreeMiner được xây dựng dựa trên phương pháp tiếp cận Apriori, phương pháp xây dựng cây con ứng viên theo cách tiếp cận của Chi [25], cải tiến phương pháp phát hiện cây con đẳng cấu của Shamir và Tsur

[29]. Chương trình được thực hiện trên máy tính Core 2 Duo E7500, 2GB RAM sử dụng ngôn ngữ lập trình C thực hiện thuật toán và trích xuất kết quả ra file hiển thị trên nền Web.




**Hình 6**. Giao diện thuật toán WETreeMiner.

Kết quả thực hiện thử nghiệm thuật toán WETreeMiner so với thuật toán TreeMiner của Zaki [27] với bộ dữ liệu CS-LOGS cắt gọn còn 1000 cây giao tác.




**Hình 7**. Thực nghiệm WETreeMiner.

## VI. KẾT LUẬN

Qua những khảo sát và nghiên cứu xây dựng thuật toán khai phá cây con thường xuyên, chúng tôi nhận thấy các thuật toán khai phá cây con thường xuyên hiện nay đều gặp hai vấn đề lớn mà các nhà khoa học đều đang nỗ lực nghiên cứu, cải tiến để nâng cao hiệu suất khai phá nhất là với các bộ dữ liệu lớn. Hai vấn đề đó là kiểm tra đẳng cấu và tính toán độ hỗ trợ của các cây con để xác định mức độ thường xuyên của cây con có thỏa mãn tính chất về độ hỗ trợ cây con được sinh ra phải ≥ một ngưỡng độ hỗ trợ tối thiểu cho trước là σ. Thuật toán tìm tất cả các cây con thường xuyên trong một cơ sở dữ liệu rừng cây cho trước nhìn chung là có độ phức tạp hàm mũ. Vì vậy khi áp dụng vào thực tế, chẳng hạn với cơ sở dữ liệu weblogs CS-LOG như trình bày trong bài báo gặp nhiều khó khăn khi khai phá với ngưỡng thấp do thời gian tính toán quá lớn dẫn đến việc khó triển khai vào thực tế. Trên cơ sở bài báo này, các tác giả đang nghiên cứu thêm một số ràng buộc có ý nghĩa trong việc liệt kê tất cả các mẫu cây con thường xuyên của cơ sở dữ liệu.

**LỜI CẢM ƠN**

Nhóm tác giả cảm ơn Phòng Khoa học dữ liệu và ứng dụng - Viện Công nghệ thông tin – Viện Hàn lâm Khoa học và công nghệ Việt Nam đã cung cấp một số tài liệu nghiên cứu, dữ liệu web logs và trang thiết bị phục vụ thực nghiệm cho bài báo này được hoàn thiện.

**TÀI LIỆU THAM KHẢO**

[1]. Agrawal, R. and Srikant, R. 1994. Fast Algorithm for Mining Association Rules, In Proceedings of the 20th International Conference on Theoretical Informatics, 598-612.

[2]. Avis, D. and Fukuda, K. 1996. Reverse Search for enumeration, Discrete Applied Mathematics, Volume 65, 21-46

[3]. Brin, S. and Page, L. 1998. The Anatomy of a Large-scale Hyper-textual Web Search Engine. In Proceedings of the 7th International World Wide Web Conference, 107-117.

[4]. Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopaln, S. and Tomkins, A. 1999. Mining the Link Structure of the World Wide Web, IEEE Computer 32(8), 60-76.

[5]. Chen, M.S., Han, J. and Yu, P.S. 1996. Data Mining: An Overview from Database Perspective, IEEE Transaction on Knowledge and Data Engineering 8, 866–883.

[6]. Cook, D.J. and Holder, L.B. 2000. Graph-based Data Mining, IEEE Intelligent Systems 15(2), 32–41.

[7]. Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. 2005. Frequent Sub-Structure-based Approach for Classifying Chemical Compounds, IEEE Transactions on Knowledge and Data Engineering 17(8), 1036-1050.

[8]. Fortin, S. 1996. The Graph Isomorphism Problem, Technical Report, no. TR06-20, The University of Alberta.

[9]. Han, J., Cheng, H., Xin, D. and Yan, X. 2007. Frequent Pattern Mining: Current Status and Future Directions, Journal of Data Mining and Knowledge Discovery 15(1), 55–86.

[10]. Han, J. and Kamber, M. 2006. Data Mining Concepts and Techniques, 2nd Edition, San Francisco: Morgan Kaufmann.

[11]. Huan, J., Wang, W., Prins, J. and Yang, J. 2004a. SPIN: Mining Maximal Frequent Subgraphs from Graph Databases, In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 581–586.

[12]. Inokuchi, A., Washio, T. and Motoda, H. 2000. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data, In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, 13–23.

[13]. Jiang, C., Coenen, F. and Zito, M. 2010. Frequent Sub-graph Mining on Edge Weighted Graphs, Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science Volume 6263, 77-88.

[14]. Kashima, H., Tsuda, K. and Inokuchi, A. 2003. Marginalized Kernels Between labelled Graphs, In Proceedings of the 20th International Conference on Machine Learning (ICML'03), 321-328.

[15]. Kleinberg, J.M. 1998. Authoritative Sources in a Hyper-linked Environment, In Proceedings of ACM-SIAM Symposium Discrete Algorithms, 668-677.

[16]. Kosala, R. and Blockeel, H. 2000. Web Mining Reasearch: A Survey, ACM, SIGKDD Explorations Newsletter 2(1), 1-15.

[17]. Liu, B. 2008. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer.

[18]. Newman, M.E.J. 2004. Detecting Commuity Structure in Networks, The European Physical Journal B-Condensed Matter and Complex Systems 38(2), 321-330.

[19]. Shasha, D., Wang, J.T.L. and Giugno, R. 2002. Algorithms and Applications of Tree and Graph Searching, In Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles on Database Systems, 39-52.

[20]. Washo, T. and Motoda, H. 2003. State of the Art of Graph-based Data Mining, SIGKDD Explorations 5, 59-68.

[21]. Yan, X. and Han, J.W. 2002. gSpan: Graph-based Substructure pattern mining, In Proceedings of International Conference on Data Mining, 721–724.

[22]. Yan, X., Yu, P.S. and Han, J. 2005b. Sub-Structure Similarity Search in Graph Databases, In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, 766-777.

[23]. Yan, X., Zhu, F., Han, J. and Yu, P.S. 2006. Searching Substructures with Superimposed Distance, In Proceedings of the 22nd International Conference on Data Engineering, 88-97.

[24]. Yan, X., Cheng, H., Han, J. and Yu, P.S. 2008. Mining Significant graph patterns by leap search, In Proceedings of the 2008 ACM SIGMOD International Conference on Management Data, Vancouver, Canada, 433-444.

[25]. Chi, Y., Yang, Y., Muntz, R. R.: HybridTreeMiner: An Efficient Algorithm for Mining Frequent Rooted Trees and Free Trees Using Canonical Forms, The 16th International Conference on Scientific and Statistical Database Management (SSDBM'04), June 2004.

[26]. Asai, T., Arimura, H., Uno, T., Nakano, S.: Discovering Frequent Substructures in Large Unordered Trees, The 6th International Conference on Discovery Science, October 2003.

[27]. Zaki, M. J.: Efficiently Mining Frequent Trees in a Forest, Proc. of the 2002 Int. Conf. Knowledge Discovery and Data Mining (SIGKDD'02), July 2002.

[28]. Tan, H., Dillon, T.S., Hadzic, F., Chang, E. and Feng, L. 2006. IMB3-Miner: Mining Induced/Embedded subtrees by Constraining the Level of Embedding, In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 450–461.

[29].Shamir, R. and Tsur, D. 1999. Faster Subtree Isomorphism, Journal of Algorithms 33(2), 267–280.

[30]. Han, J., Pei, J. and Yin, Y. 2000. Mining Frequent Patterns without Candidate Generation, In Proceedings of ACM SIGMOD International Conference on Management of Data, 1–12.