FGenHUSM: Một thuật toán hiệu quả khai thác các chuỗi sinh phổ biến lợi ích cao

Trương Chí Tín[1], Trần Ngọc Anh[1], Dương Văn Hải[1,2], Lê Hoài Bắc[2]

[1] Khoa Toán – Tin học, Trường Đại học Đà Lạt

[2] Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh

Tác giả liên hệ: Trần Ngọc Anh, anhtn@dlu.edu.vn

Ngày nhận bài: 15/07/2019, ngày sửa chữa: 09/10/2019, ngày duyệt đăng: 28/10/2019

Định danh DOI: 10.32913/mic-ict-research-vn.v2019.n2.872

Biên tập lĩnh vực điều phối phản biện và quyết định nhận đăng: PGS.TS. Lê Hoàng Sơn

Tóm tắt: Khai thác các chuỗi phổ biến và các chuỗi lợi ích cao có mức độ quan trọng khác nhau trong các ứng dụng thực tế. Gần đây, các nghiên cứu tập trung giải quyết bài toán tổng quát hơn, là khai thác tập *FHUS* chuỗi phổ biến lợi ích cao. Tuy nhiên, thời gian và bộ nhớ dùng để khai thác *FHUS* vẫn còn quá lớn. Bài báo đề xuất khái niệm tập *FGHUS* các chuỗi sinh phổ biến lợi ích cao, là một biểu diễn súc tích của *FHUS*, và một thuật toán mới hiệu quả để khai thác nó. Dựa vào hai chặn trên của độ đo lợi ích, hai chiến lược tỉa theo chiều rộng và sâu được thiết kế để loại bỏ nhanh các chuỗi ít phổ biến hoặc lợi ích thấp. Sử dụng một chặn dưới mới của lợi ích, một chiến lược tỉa địa phương mới được đề xuất để loại bỏ sớm các chuỗi không là chuỗi sinh phổ biến lợi ích cao. Dựa vào các chiến lược này, một thuật toán mới *FGenHUSM* được thiết kế để khai thác *FGHUS* mà tính hiệu quả của nó được thể hiện qua các thử nghiệm trên các cơ sở dữ liệu lớn.

Từ khóa: *Chuỗi lợi ích cao, khai thác chuỗi sinh phổ biến lợi ích cao, chặn trên và chặn dưới của độ đo lợi ích.*

Title: **FGenHUSM: An Efficient Algorithm For Mining Frequent Generator High Utility Sequences**

Abstract: Mining the set of all frequent high utility sequences (*FHUS*) in quantitative sequential databases (QSDBs) plays an important role in many real-life applications. However, for huge QSDBs and low minimum support and utility thresholds, algorithms for discovering *FHUS* often exhibit poor performance in terms of runtime and memory consumption due to the enormous cardinality of the *FHUS* set. To address this issue, this paper introduces a novel set of all frequent generator high utility sequences (*FGHUS*). This set is a concise representation of *FHUS* having a cardinality that is often much less than that of *FHUS*. Thus, it is more convenient for users to analyze the information provided by the *FGHUS* set. This paper proposes a novel algorithm named *FGenHUSM* to efficiently mine *FGHUS*. The algorithm adopts the depth and width pruning strategies to quickly eliminate infrequent or low utility sequences. In addition, it also uses a novel local pruning strategy to prune non-frequent generator high utility sequences early, which is based on a new lower bound on the utility measure. Experimental results on large QSDBs show that the proposed algorithm is efficient in terms of runtime for mining *FGHUS*, and that the pruning strategies can greatly reduce the search space.

Keywords: *High utility sequence, frequent generator high utility sequence, upper and lower bounds on utility measure.*

I. GIỚI THIỆU

Khi khai thác tập các chuỗi phổ biến (*FSM*: Frequent Sequence Mining) trên các cơ sở dữ liệu chuỗi tuần tự (SDB: Sequential Datadase), ta có thể đánh mất nhiều chuỗi lợi ích cao (HU: High Utility) quan trọng trong nhiều ứng dụng thực tế khi chúng ít phổ biến. Chẳng hạn, lợi ích của các mặt hàng bán được là thông tin rất hữu ích khi ra các quyết định trong kinh doanh. Vì vậy, bài toán khai thác các chuỗi HU (*HUSM*: High Utility Sequence Mining) trên các SDB lượng hóa (QSDB: Quantitative SDB) ra đời sau đó như một nhu cầu bức thiết.

HUSM tổng quát hơn *FSM* và có nhiều ứng dụng như phân tích hành vi duyệt web [1], dữ liệu thương mại di động [2], hiệu chỉnh gen [3], v.v. Ta xét một ứng dụng điển hình của *HUSM* là phân tích dữ liệu mua hàng. Xét cơ sở dữ liệu chuỗi biểu diễn các đơn mua hàng của khách hàng trong một cửa hàng bán lẻ. Khi đó một chuỗi chứa các mặt hàng được mua bởi một khách hàng ở các thời điểm khác nhau. Chi tiết hơn, nó là một danh sách có thứ tự của các tập mặt hàng, mỗi tập mặt hàng chứa các mặt hàng được mua cùng nhau. Lấy ví dụ, ta có chuỗi ⟨{kem đánh răng, bàn chải đánh răng}, {bánh Kinh đô, phô mai}, {nhang}⟩. Chuỗi mặt hàng có thể được mua bởi một phụ nữ. Người

này mua kem đánh răng và bàn chải đánh răng cùng nhau, sau đó mua bánh Kinh đô với phô mai, cuối cùng mua nhang. Mỗi mặt hàng có một giá bán ra. Khi đó, ta có thể xác định được giá trị của một tập các mặt hàng được mua cùng nhau cũng như giá trị của một danh sách con các tập mặt hàng trong chuỗi mua hàng của một khách hàng nào đó. Khi khai thác trên một tập rất lớn các đơn mua hàng, ta có thể biết được các thông tin như: (i) các mặt hàng nào thường được mua cùng nhau, (ii) lợi ích đem lại của một chuỗi các tập mặt hàng nào đó, v.v. Các thông tin này rất có ích cho việc ra các quyết định kinh doanh của cửa hàng.

Cho Ψ' là một QSDB chứa các chuỗi đầu vào, trong mỗi chuỗi có các sự kiện, trong mỗi sự kiện có các thuộc tính, mỗi thuộc tính gắn với một số lượng và một giá trị lợi ích. Do mỗi chuỗi α có thể xuất hiện ở các vị trí khác nhau (với các lợi ích khác nhau) trong Ψ', nên lợi ích của α trong Ψ' có thể được tính dưới dạng tổng [4], giá trị lớn nhất $u_{\max}(\alpha)$ [5, 6] hoặc giá trị nhỏ nhất $u_{\min}(\alpha)$ [7, 8] (theo nghĩa an toàn và ít rủi ro cho việc phát triển các chiến lược kinh doanh hay ra quyết định). Khi đó, lợi ích của α là tổng lợi ích trên tất cả các chuỗi Ψ' chứa α. Nếu lợi ích của α lớn hơn hoặc bằng một ngưỡng lợi ích tối thiểu mu, nó được gọi là chuỗi HU, ngược lại α được gọi là chuỗi lợi ích thấp (LU: Low Utility). Mục đích của *HUSM* là khai thác tập chuỗi lợi ích cao (*HUS*) chứa các chuỗi HU trên một QSDB.

Thuận lợi chính khi giải *FSM* là tính 'a priori', còn gọi là tính đơn điệu giảm (anti-monotonic) hay DCP (Downward-Closure Property), của độ đo hỗ trợ: Mọi chuỗi cha của một chuỗi ít phổ biến (LF: Low Frequent) cũng LF [5, 9]. Đặc tính này cho phép rút gọn đáng kể không gian tìm kiếm khi tiến hành khai thác các chuỗi lớn dần trên cây tiền tố. Đáng tiếc là trong *HUSM*, độ đo lợi ích không có tính DCP. Để khắc phục, các chặn trên (UB: Upper Bound) lợi ích được thiết kế để thu hẹp phạm vi tìm kiếm.

Chặn trên SWU [5] (thỏa DCP nhưng giá trị lớn) và các chặn trên khác chặt hơn (có giá trị nhỏ và gần với lợi ích hơn) lần lượt được thiết kế và sử dụng (mặc dù chỉ thỏa mãn các tính chất tựa DCP). Với $u_{\max}(\alpha)$, có thể kể ra SPU và SRU (2013), PEU và RSU (2016), gần đây là MEU [6] và SEU [10]. Với $u_{\min}(\alpha)$, các tác giả trong [7] đã đề xuất hai chặn trên, RBU và LRU, thiết kế hai chiến lược tỉa theo chiều sâu (DPS: Depth Pruning Strategy) và rộng (WPS: Width Pruning Strategy), và tích hợp chúng vào thuật toán *EHUSM* để khai thác hiệu quả *HUS*. Mặc dù *EHUSM* có thể khai thác nhanh *HUS*, lực lượng tập kết quả thường rất lớn, việc quản lý và phân tích chúng gây khó khăn đối với người sử dụng. Một tiếp cận thường được dùng trong *FSM* là khai thác các biểu diễn súc tích của chúng, chẳng hạn như các chuỗi tối đại, đóng và sinh [11, 12]. Chú ý rằng, bằng việc tích hợp *FSM* và *HUSM*, ta xét bài toán tổng quát hơn, là *FHUSM*: Khai thác tập *FHUS* các chuỗi phổ biến lợi ích cao (FHU: Frequent High Utility), đối với độ đo lợi ích $u_{\min}$. Theo cách tiếp cận trên, nhóm tác giả trong [13] đã đề xuất các thuật toán hiệu quả nhằm khai thác hai biểu diễn súc tích của *FHUS*, bao gồm các chuỗi phổ biến tối đại lợi ích cao (FMaxHU) và tập *FCHUS* các chuỗi phổ biến đóng lợi ích cao (FCHU). Tuy nhiên, chiều dài các chuỗi FMaxHU và FCHU thường khá lớn. Vì vậy, bài báo đề xuất một biểu diễn súc tích khác *FGHUS* của *FHUS*.

Trước đây, đã xuất hiện nhiều công trình nhằm khai thác các tập (tập thuộc tính) sinh phổ biến có/không có lợi ích cao [14–16]. Các chuỗi (danh sách có thứ tự các tập thuộc tính) đóng/tối đại/sinh phổ biến cũng đã là đối tượng của nhiều nghiên cứu gần đây [12, 17, 18]. Tập chứa các chuỗi sinh phổ biến lợi ích cao (*FGHUS*) là mở rộng tự nhiên của chuỗi sinh phổ biến truyền thống. Một chuỗi FHU được gọi là chuỗi sinh phổ biến lợi ích cao (FGHU) nếu không tồn tại chuỗi con FHU nào có cùng độ hỗ trợ.

Vì các chuỗi FGHU có chiều dài thường bé hơn nhiều so với các chuỗi FMaxHU và FCHU, nên chúng có các ưu điểm sau. Thứ nhất, ta có thể xem nó như một biểu diễn nén của *FHUS*. Điều này cũng rất phù hợp với nguyên lý chiều dài mô tả bé nhất (MDL: Minimum Description Length) [19]. Thứ hai, nó cho độ chính xác cao trong các nhiệm vụ chọn mô hình (so với *FHUS* hay tập *FCHUS*). Ngoài ra, khai thác các mẫu sinh (với chiều dài tối thiểu) còn là một bước quan trọng trong việc tìm các luật tuần tự quan trọng, chẳng hạn, các luật với ít giả thiết (vế trái) nhưng dẫn ra nhiều kết luận (vế phải), hoặc các luật không dư thừa [20]. Khi đó, các mẫu sinh chính là vế trái, vế phải có thể là các mẫu đóng hoặc không. Do đó, các mẫu sinh trong *FGHUS* thường được ưa thích hơn đối với người dùng khi cần phân tích tập kết quả, vì số lượng và chiều dài của chúng khá bé so với *FHUS*. Tập *FHUS* thường được sử dụng khi lực lượng của chúng khá bé, chẳng hạn khi ngưỡng hỗ trợ (ms) và ngưỡng lợi ích tối thiểu (mu) khá cao. Ngược lại, khi các ngưỡng này khá thấp và đặc biệt trên các tập dữ liệu lớn, để vượt qua khó khăn trong việc sử dụng cũng như phân tích tập kết quả *FHUS* với kích thước quá lớn, tập *FGHUS* sẽ là một lựa chọn phù hợp hơn với người dùng.

Mục tiêu của bài báo này là khai thác tập *FGHUS*. Để khai thác hiệu quả nó, do $FGHUS \subseteq FHUS$, nên một chuỗi LU hoặc LF sẽ không là FGHU. Do đó, tính chất DCP của độ hỗ trợ và hai chiến lược WPS và DPS dựa vào RBU và LRU [7, 8] có thể được sử dụng để tỉa hiệu quả các chuỗi LF hoặc LU không những từ *FHUS* mà cả *FGHUS*. Tuy nhiên, vì *FGHUS* không là tập con của tập tất cả các chuỗi sinh phổ biến (*FGS*), nên ta không thể áp dụng trực tiếp các điều kiện tỉa 3E trong [12] để loại bỏ các chuỗi không là chuỗi sinh lợi ích cao (GHU).

Bài báo có một số đóng góp sau đây: (i) đề xuất chặn dưới SF của $u_{\min}$; (ii) dựa vào điều kiện tỉa sớm tổng quát *GEP* [13] và SF, chiến lược tỉa địa phương (*LPG*) được thiết kế để loại bỏ sớm các ứng viên (và các mở rộng của chúng) không là GHU; (iii) tích hợp ba chiến lược *DPS*, *WPS* và *LPG* vào thuật toán *FGenHUSM* để khai thác các chuỗi sinh phổ biến lợi ích cao (FGHU); (iv) các thử nghiệm trên hai cơ sở dữ liệu lớn, thực tế và tổng hợp, đã chỉ ra tính hiệu quả của *FGenHUSM* (so với thuật toán cơ sở không áp dụng các chiến lược tỉa) về mặt thời gian chạy và lực lượng của *FGHUS* thường rất bé so với *FHUS*. Đây là thuật toán đầu tiên khai thác biểu diễn súc tích *FGHUS* của *FHUS* với độ đo lợi ích $u_{\min}$.

Phần còn lại của bài báo được tổ chức như sau. Phần II trình bày các khái niệm và kết quả cơ bản. Phần xây dựng chiến lược tỉa địa phương *LPG*. Phần IV đưa ra thuật toán *FGenHUSM* và các kết quả thử nghiệm. Phần V đưa ra các kết luận của bài báo.

II. CÁC KHÁI NIỆM VÀ KẾT QUẢ CƠ BẢN

1. Định nghĩa bài toán

Mục này giới thiệu vài khái niệm cơ sở liên quan đến bài toán *HUSM* với $u_{\min}$ trong [7].

Định nghĩa 1 ([7]): Gọi $\mathcal{A} \stackrel{\text{def}}{=} \{a_1, a_2, \ldots, a_M\}$ là tập các thuộc tính phân biệt. Mỗi thuộc tính a gắn liền với một số dương $\mathrm{P}(a)$ thể hiện giá trị lợi ích đơn vị của nó. Khi đó, ta có véctơ $\mathrm{P}(A) \stackrel{\text{def}}{=} \langle \mathrm{P}(a_1), \mathrm{P}(a_2), \ldots, \mathrm{P}(a_M) \rangle$. Một thuộc tính số lượng/định lượng (q-thuộc tính) là một cặp (a, q), với $a \in \mathcal{A}$ và $q \in R_+$ là một số lượng dương. Tập con A của $\mathcal{A}$, $A \subseteq \mathcal{A}$, được gọi là một sự kiện. Không mất tổng quát, giả sử rằng các thuộc tính trong một sự kiện được sắp tăng theo thứ tự từ điển $\prec$. Một sự kiện số lượng (q-sự kiện) ứng với A được định nghĩa là

$$A' \stackrel{\text{def}}{=} \{(a_i, q_i) \mid a_i \in A, q_i \in R_+\}.$$

A được gọi là sự kiện chiếu của A', ký hiệu là $A = \mathrm{proj}(A')$. Danh sách các q-sự kiện $\langle A'_k, k = 1, 2, \ldots, p \rangle$ ký hiệu là $\alpha' = A'_1 \to A'_2 \cdots \to A'_p$, được gọi là một q-chuỗi. Chuỗi chiếu α của q-chuỗi α' được định nghĩa là

$$\alpha = \mathrm{proj}(\alpha') \stackrel{\text{def}}{=} \mathrm{proj}(A'_1) \to \mathrm{proj}(A'_2) \to \cdots \to \mathrm{proj}(A'_p).$$

Để thuận tiện, ta ký hiệu $\alpha'[k] \stackrel{\text{def}}{=} A'_k$ và $\alpha[k] \stackrel{\text{def}}{=} \mathrm{proj}(A'_k)$. Một q-chuỗi là rỗng (<>) nếu tất cả các sự kiện của nó là rỗng. Một cơ sở dữ liệu chuỗi lượng hóa (QSDB) $\mathcal{D}'$ chứa hữu hạn các q-chuỗi đầu vào, $\mathcal{D}' = \{\Psi'_i, i = 1, 2, \ldots, N\}$ và $P(A)$. Mỗi q-chuỗi Ψ'_i gắn với một định danh (SID) i. Cơ sở dữ liệu chuỗi (không lượng hóa SDB) $\mathcal{D}$ ứng với $\mathcal{D}'$ được định nghĩa là

$$\mathcal{D} = \mathrm{proj}(\mathcal{D}') \stackrel{\text{def}}{=} \{\mathrm{proj}(\Psi'_i), i = 1, 2, \ldots, N\}.$$

Kích thước của q-chuỗi α', ký hiệu là $\mathrm{size}(\alpha')$, là số các q-sự kiện (p) của nó.

Từ đây về sau, ta xét hai q-chuỗi bất kỳ $\alpha' = A'_1 \to A'_2 \to \cdots \to A'_p$ và $\beta = B'_1 \to B'_2 \to \cdots \to B'_q$ cùng hai chuỗi tương ứng $\alpha = A_1 \to A_2 \to \cdots \to A_p$ và $\beta = B_1 \to B_2 \to \cdots \to B_q$.

Định nghĩa 2 ([7]): Xét hai q-sự kiện A' và B' sau:

$$A' = \{(a_{i_1}, q_{i_1}), (a_{i_2}, q_{i_2}), \ldots, (a_{i_m}, q_{i_m})\},$$
$$B' = \{(a_{j_1}, q_{j_1}), (a_{j_2}), q_{j_2}), \ldots, (a_{j_n}, q_{j_n})\},$$

với $m \le n$. A' được gọi là chứa trong B', ký hiệu là $A' \sqsubseteq B'$, nếu tồn tại các số tự nhiên $1 \le k_1 < \cdots < k_m \le n$ sao cho $a_{i_l} = a_{j_{k_l}}$ và $q_{i_l} = q_{j_{k_l}}$, với mọi $l = 1, 2, \ldots, m$.

Ngoài ra, ta nói α' chứa trong β', ký hiệu là $\alpha' \sqsubseteq \beta'$, nếu $p \le q$ và tồn tại p số nguyên dương, $1 \le j_1 < \cdots < j_p \le q$ sao cho $A'_k \sqsubseteq B'_{j_k}$, với mọi $k = 1, 2, \ldots, p$. Đồng thời, $\alpha' \sqsubset \beta'$ tương đương với $((\alpha' \sqsubset \beta') \wedge (\alpha' \ne \beta'))$.

Tương tự, ta cũng dùng ký hiệu $\sqsubseteq$ để định nghĩa quan hệ chứa trong trên tập tất cả các chuỗi. Ta nói $\alpha \sqsubseteq \beta$ hoặc $\beta \sqsupseteq \alpha$ (β được gọi là chuỗi cha của α) nếu tồn tại p số nguyên dương, $1 \le j_1 < \cdots < j_p \le q$ sao cho $A_k \subseteq B_{j_k}$, với mọi $k = 1, 2, \ldots, p$. Đồng thời, $\alpha \sqsubset \beta$ tương đương với $((\alpha \sqsubseteq \beta) \wedge (\alpha \ne \beta))$.

Ta nói β' chứa α, ký hiệu là $\alpha \sqsubseteq \beta'$ (hay $\beta' \sqsupseteq \alpha$, β' được gọi là q-chuỗi cha của α) nếu $\mathrm{proj}(\beta') \sqsupseteq \alpha$.

Gọi

$$\rho(\alpha) \stackrel{\text{def}}{=} \{\Psi' \in \mathcal{D}' \mid \Psi' \sqsupseteq \alpha\}$$

là tập tất cả các q-chuỗi đầu vào chứa α. Độ hỗ trợ của α được định nghĩa là số các q-chuỗi đầu vào chứa α,

$$\mathrm{supp}(\alpha) \stackrel{\text{def}}{=} |\rho(\alpha)|.$$

Định nghĩa 3 ([7]): Các lợi ích của q-thuộc tính (a, q), q-sự kiện $A' = (a_{i_1}, q_{i_1}), (a_{i_2}, q_{i_2}), \ldots, (a_{i_m}, q_{i_m})$, q-chuỗi α' và QSDB $\mathcal{D}'$ lần lượt được định nghĩa là

$$u((a, q)) \stackrel{\text{def}}{=} P(a) * q,$$
$$u(A') \stackrel{\text{def}}{=} \sum_{j=1}^{m} u((a_{i_j}, q_{i_j})),$$
$$u(\alpha') \stackrel{\text{def}}{=} \sum_{i=1}^{p} u(A'_i),$$
$$u(\mathcal{D}') \stackrel{\text{def}}{=} \sum_{\Psi' \in \mathcal{D}'} u(\Psi').$$

Để tránh tính toán lặp lại lợi ích u của mỗi q-thuộc tính (a, q) trong các q-chuỗi Ψ' của $\mathcal{D}'$, ta tính chúng một lần và thay q bởi $u((a, q)) = \mathrm{P}(a) * q$ trong cơ sở dữ liệu. Biểu diễn tương đương này của QSDB $\mathcal{D}'$ được gọi là QSDB tích hợp của $\mathcal{D}'$, cũng được ký hiệu vắn tắt là $\mathcal{D}'$. Từ đây về sau ta chỉ xét các QSDB tích hợp.

Bảng I
$\mathcal{D}'$ – MỘT QSDB MINH HỌA

SID	Chuỗi
1	$\Psi'_1 = (a,9) \to (c,2)(e,20) \to (c,1)(g,50)$ $\to (c,8)(d,16) \to (c,8)(d,16)(e,35)(g,60)$ $\to (a,3)(c,5)(e,2)$
2	$\Psi'_2 = (c,9)(f,28) \to (a,15)(c,10)$ $\to (d,16)(g,40)(a,15)(e,20) \to (c,6)(d,24)(e,25)$
3	$\Psi'_3 = (g,20) \to (e,40)(f,12) \to (b,45)(c,1) \to (d,56)$
4	$\Psi'_4 = (e,40) \to (c,2)(f,20) \to (c,3)$

Xét QSDB minh họa trong Bảng I, sẽ được dùng cho các ví dụ trong suốt bài báo. Chuỗi $\alpha = e \to ce$ chứa trong Ψ'_1, $\alpha \sqsubseteq \Psi'_1$, vì nó xuất hiện trong Ψ'_i. Lần xuất hiện đầu tiên của α trong Ψ'_1 là q-chuỗi con $\alpha' = (e,20) \to (c,8)(e,35)$ của Ψ'_1. Vậy $\text{proj}(\alpha') = \alpha$ và $u(\alpha') = 20+8+35 = 63$. Ngoài ra, do $\alpha \sqsubseteq \Psi'_2$, nên $\rho(\alpha) = \{\Psi'_1, \Psi'_2\}$ và $\text{supp}(\alpha) = 2$. SDB $\mathcal{D}$ ứng với $\mathcal{D}'$ là

$$\begin{aligned}\mathcal{D} = \{&\Psi_1 = a \to ce \to cg \to cdeg \to ace,\\ &\Psi_2 = cf \to ac \to dg \to ae \to cde,\\ &\Psi_3 = g \to ef \to bc \to d,\\ &\Psi_4 = e \to cf \to c\}.\end{aligned}$$

Định nghĩa 4 ([7]): Giả sử $\alpha \sqsubseteq \beta'$. Gọi

$$O(\alpha, \beta') \stackrel{\text{def}}{=} \{\alpha' \mid (\alpha' \sqsubseteq \beta') \wedge (\text{proj}(\alpha') = \alpha)\}$$

là tập tất cả các lần xuất hiện α' của α trong β'. Lợi ích bé nhất (gọi tắt là lợi ích) của α trong β' được định nghĩa là

$$u_{\min}(\alpha, \beta') \stackrel{\text{def}}{=} \min\{u(\alpha') \mid \alpha' \in O(\alpha, \beta')\}.$$

Lợi ích của α trong $\mathcal{D}'$ được định nghĩa là

$$u_{\min}(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} u_{\min}(\alpha, \Psi')).$$

Lúc đó, α được gọi là chuỗi lợi ích cao (HU) nếu $u_{\min}(\alpha) \geq mu$.

Từ nay, ta viết gọn α_s^u để diễn tả rằng $u_{\min}(\alpha) = u$ và $\text{supp}(\alpha) = s$. Lý do của việc sử dụng $u_{\min}$ có thể tham khảo thêm trong [7, 8, 13]. Lấy ví dụ, xét $\alpha = e \to ce$ với $\rho(\alpha) = \{\Psi'_1, \Psi'_2\}$ và $\text{supp}(\alpha) = 2$. Dễ thấy rằng,

$$\begin{aligned}O(\alpha, \Psi'_4) = \{&(e,20) \to (c,8)(e,35),\ (e,20) \to\\ &(c,5)(e,20),\ (e,35) \to (c,5)(e,20)\}.\end{aligned}$$

Do đó, $u_{\min}(\alpha, \Psi'_1) = \min\{63, 45, 60\} = 45$ và tương tự, $u_{\min}(\alpha, \Psi'_1) = 51$. Vì vậy, ta có α_2^{96}.

Định nghĩa 5: Một chuỗi HU γ được gọi là chuỗi sinh lợi ích cao, GHU, nếu không tồn tại chuỗi con HU có cùng độ hỗ trợ với nó. Tập tất cả các chuỗi GHU được định nghĩa và ký hiệu là

$$\begin{aligned}GHUS \stackrel{\text{def}}{=} \big\{&\gamma \in HUS \mid \nexists \gamma^* \in HUS :\\ &(\gamma^* \sqsubset \gamma) \wedge (\text{supp}(\gamma) = \text{supp}(\gamma^*))\big\}.\end{aligned}$$

Tập các chuỗi sinh phổ biến lợi ích cao, FGHU, được định nghĩa và ký hiệu là

$$\begin{aligned}FGHUS \stackrel{\text{def}}{=} \big\{&\gamma \in FHUS \mid \nexists \gamma^* \in FHUS :\\ &(\gamma^* \sqsubset \gamma) \wedge (\text{supp}(\gamma) = \text{supp}(\gamma^*))\big\}.\end{aligned}$$

Khai thác tập *FGHUS* là mục tiêu của bài báo này. Ví dụ, xét $mu = 226$ và $ms = 2$. Với chuỗi $\gamma = a \to g \to cde$, ta có $\gamma_2^{228} \in FHUS$. Hơn nữa, γ là một chuỗi FGHU, vì không tồn tại $\gamma^* \in FHUS$ sao cho $\gamma^* \sqsubset \gamma$ và $\text{supp}(\gamma^*) = \text{supp}(\gamma)$. Đây cũng là chuỗi FGHU duy nhất, tức là $FGHUS = \{\gamma\}$. Để ý rằng, lực lượng của *FGHUS* thường bé hơn rất nhiều so với *FHUS*, đặc biệt khi mu và ms bé. Chẳng hạn, với $mu = 1$ và $ms = 1$, $|FHUS| = 5235$, trong khi đó $|FGHUS| = 105$ (khoảng 2% của $|FHUS|$).

Định nghĩa 6 ([7]): Ta định nghĩa s-mở rộng và i-mở rộng của α và β lần lượt là

$$\alpha \diamond_s \beta \stackrel{\text{def}}{=} A_1 \to A_2 \to \cdots \to A_p \to B_1 \to B_2 \to \cdots \to B_q,$$

$$\alpha \diamond_i \beta \stackrel{\text{def}}{=} A_1 \to A_2 \to \cdots \to (A_p \cup B_1) \to B_2 \to \cdots \to B_q,$$

trong đó $a \prec b$ với mọi $a \in A_p$ và $b \in B_1$. Một mở rộng tiến (hay mở rộng) của α với β, ký hiệu là $\gamma = \alpha \diamond \beta$, là một i-mở rộng hoặc là s-mở rộng, tức là $\alpha \diamond_i \beta$ hay $\alpha \diamond_s \beta$. Khi đó, α được gọi là một tiền tố của γ và β là một hậu tố (đối với α) của γ. Ngoài ra, nếu δ là tiền tố bé nhất (của γ với $\sqsubseteq$) chứa α, ta ký hiệu δ là $\text{pref}(\gamma, \alpha)$. Hậu tố β của γ (đối với δ, tức là $\gamma = \delta \diamond \beta$) được ký hiệu là $\text{suf}(\gamma, \alpha)$. Cơ sở dữ liệu chiếu (PDB) của α được định nghĩa là

$$\mathcal{D}_\alpha \stackrel{\text{def}}{=} \{\text{suf}(\Psi, \alpha) | (\Psi \in \mathcal{D}) \wedge (\Psi \sqsupseteq \alpha)\}.$$

Ta nói $\mathcal{D}_\beta$ chứa trong $\mathcal{D}_\alpha$, ký hiệu là $\mathcal{D}_\beta \sqsubseteq D_\alpha$, nếu $\rho(\beta) \subseteq \rho(\alpha)$ và với mọi $\Psi \in \rho(\beta)$ ta có $\text{suf}(\Psi, \beta) \sqsubseteq \text{suf}(\Psi, \alpha)$. Khi $\mathcal{D}_\beta \sqsubseteq \mathcal{D}_\alpha$ và $\mathcal{D}_\alpha \sqsubseteq \mathcal{D}_\beta$, ta nói rằng $\mathcal{D}_\beta$ bằng $\mathcal{D}_\alpha$ và ký hiệu $\mathcal{D}_\beta = \mathcal{D}_\alpha$. Dễ thấy rằng $\text{supp}(\alpha) = |\mathcal{D}_\alpha|$.

Ví dụ, với $\alpha = e \to e \sqsubset \beta = e \to ce$, ta có $\rho(\alpha) = \rho(\beta) = \{\Psi_1, \Psi_2\}$, $\text{suf}(\Psi_1, \alpha) = \_g \to ace$ và $\text{suf}(\Psi_2, \alpha) = <>$. Do đó, PDB của α là $\mathcal{D}_\alpha = \{\_g \to ace, <>\}$. Tương tự, ta có $\mathcal{D}_\beta = \mathcal{D}_\alpha$.

Không gian tìm kiếm chuỗi lời giải được biểu diễn bởi một cây tiền tố với gốc là chuỗi rỗng, mỗi nút biểu diễn một chuỗi ứng viên, mỗi nút con biểu diễn một chuỗi mở rộng. Ta ký hiệu $\text{branch}(\alpha)$ là nhánh có gốc tại nút biểu diễn α, nó chứa α và các mở rộng của nó. Với một tiền tố khác rỗng α, $\beta = \alpha \diamond y$, chuỗi $\gamma = \alpha \diamond \varepsilon \diamond y$ mà $\gamma \sqsupseteq \beta$ được gọi là một mở rộng lùi (BE) của β bởi ε, với y là thuộc

tính cuối của β, ký hiệu là lastItem(β). Chẳng hạn, với $\alpha = cd$ thì branch(α) chứa các chuỗi: $cd \rightarrow \varepsilon$, $cde \rightarrow \varepsilon$ và $cdeg \rightarrow \varepsilon$, với mọi ε, kể cả <>. Cho $\beta = ce$, các chuỗi cde và $c \rightarrow cg \rightarrow cde$ là các BE của β. Tuy nhiên, $ce \rightarrow cg \rightarrow e$ là BE của $c \rightarrow e$, nhưng không là BE của β.

Thách thức chính của *HUSM* là $u_{\min}$ không đơn điệu tăng cũng không đơn điệu giảm, tức là

$$\exists \beta, \alpha, \gamma, \delta : (\beta \sqsupset \alpha) \wedge (\gamma \sqsubset \delta) \\ (u_{\min}(\beta) > u_{\min}(\alpha)) \wedge (u_{\min}(\gamma) > u_{\min}(\delta)).$$

Nói cách khác, $u_{\min}$ không thỏa mãn DCP (tính chất của supp được sử dụng trong *FSM*). Thật vậy, với $\alpha = cd$, $\beta = cdg$, và $\delta = c \rightarrow c \rightarrow cd$, dễ thấy rằng $\delta \sqsupset \alpha \sqsubset \beta$ và $u_{\min}(\beta) = 84 > u_{\min}(\alpha) = 54 > u_{\min}(\delta) = 27$. Để khắc phục, các chặn trên $u_{\min}$ thỏa mãn các tính chất tựa đơn điệu giảm (có thể yếu hơn DCP) được đề xuất trong mục tiếp theo.

2. Hai chiến lược tỉa các chuỗi LU và LF theo chiều sâu và chiều rộng

Giả sử $\alpha' \sqsubseteq \Psi'$, $\alpha = \text{proj}(\alpha') \sqsubseteq \Psi'$, với $\Psi' = B'_1 \rightarrow B'_2 \rightarrow \cdots \rightarrow B'_q \in \mathcal{D}'$, tức là tồn tại p số nguyên, $1 \le i_1 < i_2 < \cdots < i_p \le q$ sao cho $A'_k \sqsubseteq B'_{i_k}$ và $A_k = \text{proj}(A'_k) \subseteq \text{proj}(B'_{i_k})$, với mọi $k = 1, 2, \ldots, p$. Chỉ số cuối i_p được gọi là điểm cuối của α (hay α') trong Ψ', ký hiệu là end(α, Ψ') (hay end(α', Ψ')). Thuộc tính cuối của α trong B'_{i_p} được gọi là thuộc tính cuối ứng với i_p, ký hiệu là e_{i_p}. Khi đó, q-chuỗi còn lại của α trong Ψ' (đối với i_p) là phần còn lại của Ψ' sau e_{i_p}, ký hiệu là rem(α, Ψ', i_p). Gọi $i^*_p \stackrel{\text{def}}{=} \text{FEnd}(\alpha, \Psi')$ là điểm cuối đầu tiên của α trong Ψ'. Cơ sở dữ liệu chiếu lượng hóa (PQDB) $\mathcal{D}'_\alpha$ của α chứa tất cả các chuỗi còn lại rem(α, Ψ', i^*_p), với $\Psi' \in \mathcal{D}'$. Nếu $\alpha =<>$, ta quy ước $i^*_p \stackrel{\text{def}}{=} \text{FEnd}(<>, \Psi') \stackrel{\text{def}}{=} 0$, $\mathcal{D}'_{<>} \stackrel{\text{def}}{=} \mathcal{D}'$ và rem($<>, \Psi', i^*_p$) = Ψ'. Với mỗi điểm cuối $i_p = \text{end}(\alpha, \Psi')$, ta định nghĩa

$$u(\alpha, \Psi', i_p) \stackrel{\text{def}}{=} \min\{u(\alpha') \mid \alpha' \in O(\alpha, \Psi') \wedge \text{end}(\alpha', \Psi') = i_p\}.$$

Chặn trên dựa vào lợi ích còn lại của $u_{\min}(\alpha, \Psi')$ được định nghĩa và ký hiệu:

$$ub_{\text{rem}}(\alpha, \Psi') \stackrel{\text{def}}{=} \begin{cases} u(\alpha, \Psi', i^*_p) + u(\text{rem}(\alpha, \Psi', i^*_p)), & \alpha \neq <>, \\ u(\Psi'), & \alpha =<>. \end{cases}$$

Ví dụ, xét $\beta = c \rightarrow d$. Do lần xuất hiện đầu tiên của β trong Ψ'_1 là q-chuỗi con $\beta' = (c, 2) \rightarrow (d, 16) \sqsubseteq \Psi'_1$, nên

$$i^*_p \stackrel{\text{def}}{=} \text{FEnd}(\beta, \Psi'_1) = 4, \\ \text{rem}(\beta, \Psi'_1, i^*_p) = (e, 35)(g, 60) \rightarrow (a, 3)(c, 5)(e, 20), \\ u(\text{rem}(\beta, \Psi'_1, i^*_p)) = 123, \\ u(\beta, \Psi'_1, i^*_p) = \min\{u(\beta'), u((c, 1) \rightarrow (d, 16))\} = 17.$$

Vì vậy

$$ub_{\text{rem}}(\beta, \Psi'_1) = 17 + 123 = 140.$$

Một độ đo ub được gọi là một chặn trên của $u_{\min}$ nếu $u_{\min}(\alpha) \le ub(\alpha)$, với mọi α. Ngoài chặn trên truyền thống $\text{SWU}(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} u(\Psi')$ [5], ta còn có các chặn trên chặt hơn sau đây.

Định nghĩa 7 ([7]): Xét $y \in \mathcal{A}$ và α khác rỗng. Ta định nghĩa các chặn trên sau đây:

$$\text{RBU}(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} ub_{\text{rem}}(\alpha, \Psi'), \\ \text{LRU}(\alpha \diamond y) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha \diamond y)} ub_{\text{rem}}(\alpha, \Psi'), \\ \text{LRU}(y) \stackrel{\text{def}}{=} \text{SWU}(y).$$

Với hai chặn trên ub_1 và ub_2, ub_1 được gọi là chặt hơn ub_2, ký hiệu là $ub_1 \ll ub_2$, nếu $ub_1(\alpha) \le ub_2(\alpha)$ với mọi α. Theo [7, Định lý 1], ta có

$$u_{\min} \ll \text{RBU} \ll \text{LRU} \ll \text{SWU}.$$

Ví dụ, xét chuỗi $\beta = c \rightarrow cd$, $\rho(\beta) = \{\Psi'_1, \Psi'_2\}$. Khi đó

$$u_{\min}(\beta, \Psi'_1) = 25, \quad u_{\min}(\beta, \Psi'_2) = 39, \\ \text{FEnd}(\beta, \Psi'_1) = 4, \quad u(\beta, \Psi'_1, 4) = 25, \\ u(\text{rem}(\beta, \Psi'_1, 4)) = 123.$$

Vì vậy

$$u_{\min}(\beta) = u_{\min}(\beta, \Psi'_1) + u_{\min}(\beta, \Psi'_2) = 64, \\ ub_{\text{rem}}(\beta, \Psi'_1) = 150, ub_{\text{rem}}(\beta, \Psi'_2) = 64, \\ \text{RBU}(\beta) = ub_{\text{rem}}(\beta, \Psi'_1) + ub_{\text{rem}}(\beta, \Psi'_2) = 212.$$

Ngoài ra, với $\alpha = c \rightarrow c$, ta dễ thấy rằng $\beta = \alpha \diamond_i d$ và

$$\text{LRU}(\beta) = UB_{\text{rem}}(\alpha, \Psi'_1) + ub_{\text{rem}}(\alpha, \Psi'_2) \\ = 200 + 165 = 365, \\ \text{SWU}(\beta) = u(\Psi'_1) + u(\Psi'_2) = 229 + 208 = 437.$$

Vì vậy ta có

$$u_{\min}(\beta) < \text{RBU}(\beta) < \text{LRU}(\beta) < \text{SWU}(\beta).$$

Để loại nhanh các chuỗi có lợi ích thấp hoặc ít phổ biến, dựa vào [7, Định lý 2], ta thiết kế hai chiến lược *DPS* và *WPS* nhằm thu hẹp hiệu quả không gian tìm kiếm. Trước hết, ta có chiến lược tỉa theo chiều sâu dựa vào RBU, viết là *DPS*(RBU), như sau: "*Nếu* RBU(α) < *mu thì* branch(α) *có thể được tỉa*".

Gọi hai tập thuộc tính ứng viên cho các $i-$ và s-mở rộng lần lượt là:

$$I_{\text{LRU}}(\alpha) \stackrel{\text{def}}{=} \{z \mid (\text{LRU}(\alpha \diamond_i z) \ge mu) \wedge (\text{supp}(\alpha \diamond_i z) \ge ms)\}, \\ S_{\text{LRU}}(\alpha) \stackrel{\text{def}}{=} \{z \mid (\text{LRU}(\alpha \diamond_s z) \ge mu) \wedge (\text{supp}(\alpha \diamond_s z) \ge ms)\}.$$

Nếu $u_{\min}(\alpha \diamond_i z) \geq mu$ và $\text{supp}(\alpha \diamond_i z) \geq ms$, thì

$$\text{LRU}(\alpha \diamond_i z) \geq u_{\min}(\alpha \diamond_i z) \geq mu,$$

tức là $z \in I_{\text{LRU}}(\alpha)$. Vì $\text{LRU}(\alpha \diamond_i y \diamond_i z) \leq \text{LRU}(\alpha \diamond_i z)$,

$$\max\{\text{LRU}(\gamma) \mid \gamma \in \{\alpha \diamond_i y \diamond_s z, \alpha \diamond_s y \diamond_i z, \alpha \diamond_s y \diamond_s z\}\} \leq \text{LRU}(\alpha \diamond_s z),$$

$$\text{supp}(\alpha \diamond y \diamond z) \leq \text{supp}(\alpha \diamond z), \quad \text{với } \alpha \diamond y \diamond z \sqsupseteq \alpha \diamond z.$$

Cho nên ta có

$$I_{\text{LRU}}(\alpha \diamond_i y) \subseteq I_{\text{LRU}}(\alpha),$$
$$(S_{\text{LRU}}(\alpha \diamond_i y) \cup S_{\text{LRU}}(\alpha \diamond_i y)) \subseteq S_{\text{LRU}}(\alpha).$$

Từ đó, ta có chiến lược tỉa theo chiều rộng dựa vào LRU, viết là *WPS*(LRU), như sau: "*Nếu* $\text{LRU}(\alpha \diamond y) < mu$, *ta không cần xét tất cả các mở rộng tiến của* $\alpha \diamond y$, *tức là* $\text{branch}(\alpha \diamond y)$, *và các mở rộng lùi của nó.*

Lấy ví dụ, với $mu = 230$ và $ms = 2$, ta có $\text{supp}(b) < ms$ và $\text{LRU}(b) = 174 < mu$. Các giá trị LRU của các thuộc tính còn lại, trong tập chứa các thuộc tính ứng viên cho các mở rộng, $I_{\text{LRU}}(<>) = S_{\text{LRU}}(<>) = \{a, c, d, e, f, g\}$, đều lớn hơn hay bằng mu (chẳng hạn $\text{LRU}(a) = 447$). Khi đó, do *WPS*(LRU), ta có thể loại thuộc tính b khỏi $\mathcal{D}'$.

Để minh họa tác dụng tỉa theo chiều sâu của RBU, xét hai giá trị $\text{RBU}(ac) = 199$ và $\text{RBU}(a \rightarrow c) = 229$ đều bé hơn mu. Sử dụng *DPS*(RBU), toàn bộ nhánh $\text{branch}(ac)$ và $\text{branch}(a \rightarrow c)$ bị tỉa. Dễ thấy rằng, sử dụng supp, $I_{\text{LRU}}(a) = \{c, e\}$ và $S_{\text{LRU}}(a) = \{a, c, d, e, g\} (\subseteq S_{\text{LRU}}(<>))$. Vì $S_{\text{LRU}}(a \rightarrow c) \subseteq S_{\text{LRU}}(a)$ và với bất kỳ $x \in S_{\text{LRU}}(a)$, $\text{RBU}(a \rightarrow c \rightarrow x) < \text{LRU}(a \rightarrow c \rightarrow x) = 229 < mu$ hay $\text{supp}(a \rightarrow c \rightarrow x) = 1 < ms$, $S_{\text{LRU}}(a \rightarrow c) = \emptyset$. Do đó, bởi *DPS*(RBU), ta chỉ có thể tỉa nhánh $\text{branch}(a \rightarrow c \rightarrow x)$, với mọi $x \in \{a, c, d, e, g\}$.

Tuy nhiên, sử dụng *WPS*(LRU), ta vẫn có thể loại bỏ tất cả các nhánh mở rộng lùi của $\text{branch}(a \rightarrow c \rightarrow x)$, ví dụ như $\text{branch}(a \rightarrow c \rightarrow g \rightarrow d)$, $\text{branch}(a \rightarrow ce \rightarrow cg \rightarrow cd)$ (của $\text{branch}(a \rightarrow c \rightarrow d)$), v.v. Thật vậy, lý do là $I_{\text{LRU}}(a \rightarrow ce \rightarrow cg \rightarrow c) \subseteq S_{\text{LRU}}(a \rightarrow ce) \subseteq S_{\text{LRU}}(a \rightarrow c) = \emptyset$. Ngoài ra, mặc dù chặt hơn LRU, nhưng RBU không có tác dụng tỉa theo chiều rộng. Nếu dùng RBU để tỉa theo chiều rộng, ta có thể tỉa nhầm một số lời giải. Thật vậy, với $mu = 225$ và $ms = 2$, vì $I_{\text{RBU}}(a \rightarrow g \rightarrow e \rightarrow c) (\subseteq S_{\text{RBU}}(a \rightarrow g \rightarrow e) = \{c\})$, nên $I_{\text{RBU}}(a \rightarrow g \rightarrow e \rightarrow c) = \emptyset$. Tuy nhiên, $a \rightarrow g \rightarrow e \rightarrow ce_2^{225} \in FHUS$. Có thể kết luận rằng, mặc dù LRU lỏng hơn RBU, tác dụng tỉa của *WPS* thật sự mạnh hơn *DPS*.

Vì vậy, *DPS* và *WPS* được dùng để tỉa các nhánh LU (và LF) trên cây tìm kiếm tiền tố, có hiệu quả với các giá trị mu cao [7]. Tuy nhiên, nhiều chuỗi HU có thể không là các chuỗi sinh HU (non-GHU). Do đó, trong phần tiếp theo, chiến lược *LPG* tỉa các ứng viên non-GHU sẽ được đề xuất. Để ý rằng, *LPG* sẽ hiệu quả với các giá trị mu thấp.

III. CHIẾN LƯỢC LPG TỈA SỚM CÁC CHUỖI KHÔNG LÀ SINH LỢI ÍCH CAO

1. Điều kiện tỉa sớm tổng quát

Trước hết, chúng tôi nhắc lại hai độ đo SE, SLIP và điều kiện tỉa sớm tổng quát đã được dùng để tỉa các chuỗi phổ biến đóng và các chuỗi phổ biến đóng có lợi ích cao [13], hoặc các chuỗi sinh phổ biến [12].

Định nghĩa 8 ([12]): Tổng các sự kiện còn lại trong một PDB $\mathcal{D}_\alpha$ được định nghĩa và ký hiệu là:

$$\text{SE}(\alpha) \stackrel{\text{def}}{=} \sum_{\substack{\Psi \in \mathcal{D}: \\ \text{suf}(\Psi, \alpha) \in \mathcal{D}_\alpha}} \big(\text{size}(\Psi) - \text{size}(\text{pref}(\Psi, \alpha)) + 1\big).$$

Với chuỗi α, gọi δ là tiền tố bé nhất của Ψ chứa α, $\Psi \in \rho(\alpha)$. Đặt $fi(\alpha, \Psi)$ là chỉ số (trong Ψ) của sự kiện cuối của δ và $\text{lastEvent}(\alpha)$ là sự kiện cuối của α. Gọi $\text{LP}(\alpha, \Psi)$ là danh sách các vị trí thứ i khác nhau trong Ψ mà $\text{lastEvent}(\alpha) \subseteq \Psi[i]$ và $fi(\alpha, \Psi) \leq i \leq \text{size}(\Psi)$. Không mất tổng quát, giả sử rằng các chỉ số trong $\text{LP}(\alpha, \Psi)$ được sắp tăng. Khi đó,

$$\text{slip}(\alpha, \Psi) \stackrel{\text{def}}{=} |\text{LP}(\alpha, \Psi)|$$

là số các vị trí xuất hiện của sự kiện cuối của δ trong Ψ.

Định nghĩa 9 ([13]): Số đo SLIP của PDB $\mathcal{D}_\alpha$ được định nghĩa và ký hiệu là

$$\text{SLIP}(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi \in \rho(\alpha)} \text{slip}(\alpha, \Psi).$$

Ví dụ, xét $\alpha = e \rightarrow e$. Vì $\mathcal{D}_\alpha = \{\_g \rightarrow ace, <>\}$, nên $\text{SE}(\alpha) = 3$. Ta có $\rho(\alpha) = \{\Psi_1, \Psi_2\}$. Vì sự kiện cuối e xuất hiện hai lần trong Ψ_1 tại vị trí thứ 4 và thứ 5, nên $\text{LP}(\alpha, \Psi_1) = \{4, 5\}$. Tương tự, $\text{LP}(\alpha, \Psi_2) = \{5\}$. Vì vậy, $\text{SLIP}(\alpha) = 3$. Từ SE và SLIP, ta có điều kiện tỉa sớm tổng quát (GEP) trong Định lý 1 sau đây.

Định lý 1 (Điều kiện tỉa sớm tổng quát [13]): Xét hai chuỗi α và β thỏa mãn $\alpha \sqsubseteq \beta$. Khi đó:

a) Nếu $\text{SE}(\alpha) = \text{SE}(\beta)$, thì $\text{supp}(\alpha) = \text{supp}(\beta)$ và $\mathcal{D}_\gamma = \mathcal{D}_\lambda$ với mọi s-mở rộng γ của α và λ của β với cùng một chuỗi khác rỗng;

b) Nếu $\text{SE}(\alpha) = \text{SE}(\beta)$ và $\text{SLIP}(\alpha) = \text{SLIP}(\beta)$, thì $\text{supp}(\alpha) = \text{supp}(\beta)$ và $\mathcal{D}_\gamma = \mathcal{D}_\lambda$ với tất cả mở rộng γ của α và λ của β với cùng một chuỗi khác rỗng.

2. Chiến lược LPG

Dựa vào GEP và chặn dưới SF sau đây của $u_{\min}$, chiến lược *LPG* tỉa sớm các chuỗi non-GHU được đề xuất.

Định nghĩa 10: Cho chuỗi phổ biến α, tức là $\text{supp}(\alpha) \geq k$, với $k \stackrel{\text{def}}{=} ms$. Sau khi sắp xếp tăng dần dãy

$$\mathcal{U}(\alpha) \stackrel{\text{def}}{=} \{u_{\min}(\alpha, \Psi') \mid \Psi' \in \rho(\alpha)\},$$

ta thu được dãy $\{u_i, 1 \le i \le n\}$ với $n \ge k$. Chặn dưới SF của $u_{\min}$ được định nghĩa là tổng k giá trị bé nhất của $\mathcal{U}(\alpha)$,

$$\mathrm{SF}(\alpha) \stackrel{\text{def}}{=} \sum_{1 \le i \le k} u_i.$$

Định lý 2: Ta có:

(a) SF là một chặn dưới của $u_{\min}$, tức là $\mathrm{SF}(\alpha) \le u_{\min}(\alpha)$, với mọi α;

(b) SF là đơn điệu tăng đối với các mở rộng, tức là $\mathrm{SF}(\beta) \ge \mathrm{SF}(\alpha)$, với mọi $\beta = \alpha \diamond \delta \sqsupseteq \alpha$.

Chứng minh: (a) Với mọi α, ta có $\mathrm{SF}(\alpha) = \sum_{i=1}^{k} u_i \le \sum_{i=1}^{n} u_i = u_{\min}(\alpha)$.

(b) Với mở rộng bất kỳ $\beta = \alpha \diamond \delta \sqsupseteq \alpha$, ta có $\rho(\beta) \subseteq \rho(\alpha)$ và $u_{\min}(\beta, \Psi') \ge u_{\min}(\alpha, \Psi')$, với mọi $\Psi' \in \rho(\beta)$. Thật vậy $u_{\min}(\beta, \Psi') \stackrel{\text{def}}{=} \min\{u(\beta') \mid \beta' \in O(\beta, \Psi')\} = u(\beta'_{\min})$, $\exists \alpha'_* \in O(\alpha, \Psi')$, $\varepsilon'_* \sqsubseteq \beta'_{\min}$ sao cho $\alpha \sqsubseteq \beta'_{\min} = \alpha'_* \diamond \varepsilon'_* \in O(\beta, \Psi')$ và $u(\beta'_{\min}) = u(\alpha'_*) + u(\varepsilon'_*) \ge u(\alpha'_*) \ge \min\{u(\alpha') \mid \alpha' \in O(\alpha, \Psi')\} \stackrel{\text{def}}{=} u_{\min}(\alpha, \Psi')$. Sau khi sắp tăng $\mathcal{U}(\alpha)$ và $\mathcal{U}(\beta)$, ta có được hai dãy $\{u_i, 1 \le i \le n\}$, $\{u'_j, 1 \le j \le m\}$ với $n \ge m \ge k$. Khi đó, tồn tại các số nguyên $1 \le i_1 < i_2 < \cdots < i_m$ sao cho $u_{i_j} \le u'_j$, với $1 \le j \le m$, nên ta có

$$\mathrm{SF}(\alpha) = \sum_{i=1}^{k} u_i \le \sum_{j=1}^{k} u_{i_j} \le \sum_{j=1}^{k} u'_j = \mathrm{SF}(\beta).$$

■

Ví dụ, khi xét hai ngưỡng $mu = 85$, $ms = 2$ và mở rộng $\beta = e \to e$ của $\alpha = e$, ta có $\rho(\beta) = \{\Psi'_1, \Psi'_2\} \subset \rho(\alpha) = \mathcal{D}'$, $u_{\min}(\alpha) = 120$, $u_{\min}(\beta) = 85$, và $\mathcal{U}(\alpha) = \{20; 20; 40; 40\}$, $\mathcal{U}(\beta) = \{40; 45\}$, nên $\mathrm{SF}(\alpha) = 20 + 20 = 40$ và $\mathrm{SF}(\beta) = 40 + 45 = 85$. Khi đó, $\mathrm{SF}(\alpha) < u_{\min}(\alpha)$, $\mathrm{SF}(\beta) \le u_{\min}(\beta)$, và $\mathrm{SF}(\alpha) < \mathrm{SF}(\beta)$.

Trên cây tiền tố với gốc *Root* =<>, với mỗi nút q (1-thuộc tính), ta có chuỗi tương ứng $\alpha = \mathrm{path}(q)$, trong đó $\mathrm{path}(q)$ là chuỗi đầy đủ thu được bằng cách duyệt từ *Root* đến q. Nút q có i-mở rộng và s-mở rộng bởi hai thuộc tính u và v tương ứng, ký hiệu là, $\alpha \diamond_i u$ và $\alpha \diamond_s v$. Khi đó, ta gán $u.type$ = *i-ext* và $v.type$ = *s-ext*. Ký hiệu tập chứa α và tất cả các mở rộng (hay các s-mở rộng, các i-mở rộng) cũng như tất cả các hậu duệ của các mở rộng này (hay các s-mở rộng và các i-mở rộng) là $\mathrm{branch}(\alpha)$ (hay tương ứng là *s-child branches*(α) và *i-child branches*(α)). Nếu tất cả các chuỗi trong các nhánh này không là chuỗi sinh HU, ta ký hiệu chúng là *non-GHU branch*(α) (hay tương ứng là *non-GHU s-child branches*(α) và *non-GHU i-child branches*(α)).

Định lý 3 (Chiến lược *LPG*): Xét q, u, v là các nút có cùng tiền tố và $\alpha = \mathrm{path}(q)$ là chuỗi ứng với q.

(a) Với mỗi i-mở rộng, $i_{\text{new}} = \alpha \diamond_i u$:

(i) Nếu $\mathrm{SE}(\alpha) = \mathrm{SE}(i_{\text{new}})$ và $\mathrm{SF}(\alpha) \ge mu$, thì *non-GHU s-child branches*(i_{new}) có thể được tỉa. Hơn nữa, nếu $\mathrm{SLIP}(\alpha) = \mathrm{SLIP}(i_{\text{new}})$, thì toàn bộ nhánh *non-GHU branch*(i_{new}) cũng được tỉa;

(ii) Nếu $\mathrm{SE}(\mathrm{path}(u)) = \mathrm{SE}(i_{\text{new}})$ và $\mathrm{SF}(\mathrm{path}(u)) \ge mu$, thì *non-GHU s-child branches*(i_{new}) được tỉa. Ngoài ra, nếu $\mathrm{SLIP}(\mathrm{path}(u)) = \mathrm{SLIP}(i_{\text{new}})$, thì *non-GHU branch*$(i_{\text{new}})$ được tỉa;

(iii) Nếu $q.type$ = *s-ext*, $q.Parent.Item$ = $q.Item$ (tồn tại $r \in q.Parent.iChildren$ sao cho $r.Item$ = $u.Item$), $q.Parent.type$ = *s-ext*, $\mathrm{SE}(\mathrm{path}(r)) = \mathrm{SE}(i_{\text{new}})$ và $\mathrm{SF}(\mathrm{path}(r)) \ge mu$, thì ta có thể tỉa *non-GHU branch*(i_{new}).

(b) Với mỗi s-mở rộng, $s_{\text{new}} = \alpha \diamond_s v$ sao cho $v.type$ = *s-ext*, nếu $\mathrm{SE}(\mathrm{path}(v)) = \mathrm{SE}(s_{\text{new}})$ và $\mathrm{SF}(\mathrm{path}(v)) \ge mu$, thì *non-GHU branch*$(s_{\text{new}})$ có thể được tỉa.

Chứng minh: Ta sẽ chứng minh a(i). Các khẳng định còn lại có thể được chứng minh tương tự. Giả sử rằng $\mathrm{SE}(\alpha) = \mathrm{SE}(i_{\text{new}})$ và $\mathrm{SF}(\alpha) \ge mu$. Vì $\alpha \sqsubset i_{\text{new}}$, với mọi s-mở rộng γ và λ của α và i_{new} tương ứng với cùng một chuỗi ε (kể cả chuỗi rỗng), $\gamma = \alpha \diamond_s \varepsilon$ và $\lambda = i_{\text{new}} \diamond_s \varepsilon$, thì $\gamma \sqsubset \lambda$ và $\mathrm{supp}(\gamma) = \mathrm{supp}(\lambda)$, vì do phần (a) của định lý 1, với ε khác rỗng, $\mathcal{D}_\gamma = \mathcal{D}_\lambda$, nên $|\mathcal{D}_\gamma| = |\mathcal{D}_\lambda|$. Hơn nữa, nếu $\mathrm{SLIP}(\alpha) = \mathrm{SLIP}(i_{\text{new}})$, do phần (b) của định lý 1, ta cũng có $\gamma \sqsubset \lambda$ và $\mathrm{supp}(\gamma) = \mathrm{supp}(\lambda)$, với bất kỳ i-mở rộng $\lambda = i_{\text{new}} \diamond_i \varepsilon$ của i_{new} và i-mở rộng tương ứng $\gamma = \alpha \diamond_i \varepsilon$ của α với cùng ε. Vì vậy, với bất kỳ mở rộng $\lambda = i_{\text{new}} \diamond \varepsilon$ của i_{new} và $\gamma = \alpha \diamond \varepsilon$ của α, ta luôn có $\gamma \sqsubset \lambda$ và $\mathrm{supp}(\gamma) = \mathrm{supp}(\lambda)$. Ngoài ra, do tính đơn điệu tăng của SF đối với phép toán mở rộng (phần (b) của định lý 2), nên $u_{\min}(\gamma) \ge \mathrm{SF}(\gamma) \ge \mathrm{SF}(\alpha) \ge mu$, nghĩa là $\exists \gamma \in HUS$: $\gamma \sqsubset \lambda$ và $\mathrm{supp}(\gamma) = \mathrm{supp}(\lambda)$. Vì vậy, $\lambda \notin GHUS$. ■

Chiến lược *LPG* cho phép tỉa sớm các chuỗi non-GHU ngay khi chúng vừa được tạo ra trong hai mức liền kề nhau trên cây tiền tố. Ngoài ra, ta không cần kiểm tra quan hệ cha con trên các chuỗi sử dụng trong định lý 3, chẳng hạn, $\alpha \sqsubset i_{\text{new}} = \alpha \diamond_i u$ trong a(i). Do đó, nó giúp rút ngắn đáng kể thời gian thực thi cũng như giảm lượng bộ nhớ lưu trữ trong quá trình khai thác. Sau đây, ta sẽ minh họa các trường hợp áp dụng của định lý 3.

3. Minh họa chiến lược *LPG*

Trước hết, ta minh họa trường hợp a(i) của định lý 3. Xét hai ngưỡng $mu = 85$, $ms = 2$ và $\alpha = a \to dg \to a \sqsubseteq i_{\text{new}} = \alpha \diamond_i e = a \to dg \to ae$. Ta có $\mathrm{SE}(i_{\text{new}}) = \mathrm{SE}(\alpha) = 3$, $\mathrm{SF}(\alpha) = 174 \ge mu$, và $\mathrm{SLIP}(i_{\text{new}}) = \mathrm{SLIP}(\alpha) = 2$. Theo phần a(i) của định lý 3, toàn bộ nhánh *non-GHU branch*(i_{new}) được tỉa.

Bây giờ, ta minh họa trường hợp a(ii). Đặt $mu = 75$ và $ms = 2$. Cho trước hai nút $q = a$ và $u = e$ với cùng tiền tố $a \to d$, ta có $\alpha \stackrel{\text{def}}{=} \mathrm{path}(q) = a \to d \to a$ và $\gamma \stackrel{\text{def}}{=} \mathrm{path}(u) = a \to d \to e$. Khi đó, $i_{\text{new}} = a \to d \to ae \sqsupseteq \gamma$, $\rho(i_{\text{new}}) = \rho(\gamma) = \{\Psi_1, \Psi_2\}$. Ta có $u_{\min}(i_{\text{new}}) = 96 \ge mu$

Thuật toán 1: Thuật toán FGenHUSM

Dữ liệu vào: $\mathcal{D}'$, ms, mu
Dữ liệu ra: $FGHUS$

```
1  FGHUS ← ∅;
2  S ← I ← {i ∈ 𝒜 | (LRU(i) ≥ mu) ∧ supp(i) ≥ ms)};
3  Loại các thuộc tính không thuộc S khỏi 𝒟′;
4  for i ∈ S do
5  |  DfsFGHUS(i, S, I, ms, mu);
6  end
```

và $\text{supp}(i_{\text{new}}) = 2 \geq ms$, nên i_{new} là chuỗi FHU. Tuy nhiên, i_{new} và các s-mở rộng của nó không là các chuỗi FGHU. Thật vậy, ta có $\text{SF}(\gamma) = 78 \geq mu$ và $\text{SE}(i_{\text{new}}) = \text{SE}(\gamma) = 3$. Tuy nhiên, mặc dù $\text{LP}(i_{\text{new}}, \Psi_1) = \text{LP}(\gamma, \Psi_1) = 5$, nhưng do $\text{LP}(i_{\text{new}}, \Psi_2) = 5 \neq \text{LP}(\gamma, \Psi_2) = \{4, 5\}$, cho nên $\text{SLIP}(i_{\text{new}}) = 2 \neq \text{SLIP}(\gamma) = 3$. Theo phần a(ii) của định lý 3, các nhánh *non-GHU s-child branches*(i_{new}) được tỉa.

Để minh họa trường hợp (b), xét hai nút $q = g$ và $v = e$ với cùng tiền tố $a \rightarrow d$, ta có $\alpha \overset{\text{def}}{=} \text{path}(q) = a \rightarrow dg$, $\gamma \overset{\text{def}}{=} \text{path}(v) = a \rightarrow d \rightarrow e$ và $v.type = s\text{-}ext$. Khi đó, $s_{\text{new}} = \alpha \diamond_s e = a \rightarrow dg \rightarrow e \sqsupseteq \gamma$, $\text{SE}(s_{\text{new}}) = \text{SE}(\gamma) = 3$ và $\text{SF}(\gamma) = 78 \geq mu$. Vì $\text{lastEvent}(\gamma) = \text{lastEvent}(s_{\text{new}}) = e$, nên $\text{SLIP}(s_{\text{new}}) = \text{SLIP}(\gamma) = 3$. Theo phần (b) của định lý 3, ta có thể tỉa ngay nhánh *non-GHU branch*(s_{new}).

Bây giờ, đặt $mu = 65$, $ms = 2$ và xét hai nút $q = c$ và $(u \equiv) r = e$ ứng với các chuỗi $\alpha \overset{\text{def}}{=} \text{path}(q) = c \rightarrow c \rightarrow c$ và $\delta \overset{\text{def}}{=} \text{path}(r) = c \rightarrow ce$ có cùng chuỗi cha *parent* $= c \rightarrow c$ và $parent.type = q.type = s\text{-}ext$. Khi đó, ta có $i_{\text{new}} = \alpha \diamond_i u = c \rightarrow c \rightarrow ce \sqsupseteq \delta$, $\text{SE}(i_{\text{new}}) = \text{SE}(\delta) = 3$, $\text{SF}(\delta) = 66 \geq mu$ và $\text{SLIP}(i_{\text{new}}) = \text{SLIP}(\delta) = 3$. Vì vậy, theo phần a(iii) của định lý 3, toàn bộ nhánh *non-GHU branch*(i_{new}) có thể được tỉa.

Để ý rằng, điều kiện "$\text{SF}(\alpha) \geq mu$" là cần thiết cho việc tỉa. Thật vậy, với $mu = 99$ và $ms = 3$, hai nút $q = c$ và $v = d$ có cùng tiền tố <>, ta có $\alpha \overset{\text{def}}{=} \text{path}(q) = c$, $\gamma \overset{\text{def}}{=} \text{path}(v) = d$ và $v.type = s\text{-}ext$, $s_{\text{new}} = \alpha \diamond_s d = c \rightarrow d \sqsupseteq \gamma$ với $\text{SE}(s_{\text{new}}) = \text{SE}(\gamma) = 6$ và $\text{SLIP}(s_{\text{new}}) = \text{SLIP}(\gamma) = 3$. Do đó, nếu dùng phần (b) của định lý 3 mà bỏ qua phép kiểm tra "$\text{SF}(\gamma) = 88 \geq mu = 99$", nhánh $\text{branch}(s_{\text{new}})$ sẽ bị tỉa nhầm. Thật vậy, s_{new} là chuỗi FGHU, vì $u_{\min}(s_{\text{new}}) = 99 \geq mu$, $\text{supp}(s_{\text{new}}) = 3 \geq ms$ và không tồn tại $\gamma^* \in HUS$ mà $\gamma^* \sqsubset s_{\text{new}}$ và $\text{supp}(\gamma^*) = \text{supp}(s_{\text{new}})$. Thật vậy, với mọi $c, d \sqsubset s_{\text{new}}$, $u_{\min}(c) = 10 < u_{\min}(d) = 88 < mu$.

IV. THUẬT TOÁN VÀ THỬ NGHIỆM

1. Thuật toán FGenHUSM

Thuật toán *FGenHUSM* khai thác tập *FGHUS* được cho trong thuật toán 1. Đầu vào của nó là QSDB $\mathcal{D}'$, hai ngưỡng ms và mu; đầu ra là *FGHUS* (để tiện trình bày, ta xem nó như biến toàn cục đối với các thủ tục và hàm con). Ở các

Thuật toán 2: Thủ tục DfsFGHUS

Dữ liệu vào: γ, S, I, ms, mu
Dữ liệu ra: $FGHUS$

```
1  update ← UpdateFGHUS(γ, mu);
2  if update is true then
3  |  return;
4  end
5  S_new ← I_new ← ∅;
6  for i ∈ I mà i > lastItem(γ) do
7  |  if LRU(γ ⋄_i i) ≥ mu và supp(γ ⋄_i i) ≥ ms then
8  |  |  I_new ← I_new ∪ {i};
9  |  end
10 end
11 for i ∈ I_new mà RBU(γ ⋄_i i) ≥ mu do
12 |  i_new ← γ ⋄_i i ;
13 |  LocalPruningGHU(i_new, γ, mu);
14 |  β ← anh em của γ ứng với i, có cùng kiểu mở rộng
   |    với γ;
15 |  LocalPruningGHU(i_new, β, mu);
16 |  if γ.type = s-ext, γ.typeOfParent = s-ext,
   |    γ.lastItemOfParent = γ.lastItem và
   |    ∃δ ∈ i- anh em của γ mà δ.lastItem = i then
17 |  |  LocalPruningGHU(i_new, δ, mu);
18 |  end
19 end
20 if γ.do-s-ext hoặc γ.do-ext then
21 |  for i ∈ S do
22 |  |  if LRU(γ ⋄_s i) ≥ mu và supp(γ ⋄_s i) ≥ ms then
23 |  |  |  S_new ← S_new ∪ {i};
24 |  |  end
25 |  end
26 |  for i ∈ S_new mà RBU(γ ⋄_s i) ≥ mu do
27 |  |  s_new ← γ ⋄_s i ;
28 |  |  α ← anh em của γ ứng với i và có kiểu mở
   |  |    rộng s-ext;
29 |  |  LocalPruningGHU(s_new, α, mu);
30 |  end
31 |  for i ∈ S_new do
32 |  |  DfsFGHUS(γ ⋄_s i, S_new, S_new, ms, mu);
33 |  end
34 else
35 |  S_new ← S;
36 end
37 for i ∈ I_new do
38 |  DfsFGHUS(γ ⋄_i i, S_new, I_new, ms, mu);
39 end
```

dòng 2 và 3, *WPS* dựa vào LRU và supp được sử dụng. Ở dòng 5, nó gọi thủ tục *DfsFGHUS* (Thuật toán 2) tìm kiếm các chuỗi FGHU. Thủ tục này nhận vào chuỗi γ, hai tập thuộc tính S và I chứa các thuộc tính cho các s- và i-mở rộng tương ứng của γ. Trong thủ tục, tại các dòng 1–3, hàm *UpdateFGHUS* (Thuật toán 3) kiểm tra xem γ có là GHU không và cập nhật *FGHUS* khi cần. Tại các dòng 13, 15, 17, 20 và 29 trong *DfsFGHUS* và các dòng 4, 7 và 15 trong *UpdateFGHUS*, chiến lược *LPG* được sử dụng thông qua thủ tục *LocalPruningGHU* (Thuật toán 4).

Tính đúng của thuật toán được bảo đảm bởi định lý 3 trong bài báo này và định lý 2 trong [7]. Cụ thể, việc áp dụng hai chiến lược tỉa *WPS* và *DPS* nhằm loại nhanh các

Thuật toán 3: Hàm UpdateFGHUS

Dữ liệu vào: γ, mu
Dữ liệu ra: *true* nếu γ là GHU và *false* trong trường hợp ngược lại và cập nhật FGHUS khi cần

```
1  if RBU(γ) < mu then
2  |   return true;
3  end
4  if γ.do-ext is false then
5  |   return true;
6  end
7  if γ.do-s-ext is false then
8  |   return false;
9  end
10 if u_min(γ) < mu then
11 |   return false;
12 end
13 for α ∈ FGHUS do
14 |   if supp(α) = supp(γ) và γ ⊐ α then
15 |   |   LocalPruningGHU(γ, α);
16 |   |   if γ.do-ext is false then
17 |   |   |   return true;
18 |   |   end
19 |   |   return false;
20 |   end
21 end
22 for α ∈ FGHUS do
23 |   if supp(α) = supp(γ) và α ⊐ γ then
24 |   |   Loại α ra khỏi FGHUS;
25 |   end
26 end
27 FGHUS ← FGHUS ∪ {γ};
28 return false;
```

Thuật toán 4: Thủ tục LocalPruningGHU

Dữ liệu vào: *cha, con, mu*
Dữ liệu ra: *cha*

```
1 if SF(con) ≥ mu và SE(cha) = SE(con) then
2 |   cha.do-s-ext ← false;
3 |   if SLIP(cha) = SLIP(con) then
4 |   |   cha.do-ext ← false;
5 |   end
6 end
```

nhánh chỉ chứa các chuỗi có lợi ích thấp hoặc ít phổ biến, và mục đích của chiến lược *LPG* là tỉa sớm các nhánh chỉ chứa các chuỗi không là chuỗi sinh lợi ích cao. Vì vậy, việc dùng ba chiến lược trên vào thuật toán *FGenHUSM* không làm mất đi bất kỳ chuỗi sinh phổ biến lợi ích cao nào. Chú ý rằng, khác với chiến lược *LPC* trong [13] nhằm tỉa sớm các chuỗi không là chuỗi đóng phổ biến lợi ích cao, chiến lược *LPG* cần sử dụng thêm chặn dưới SF của độ đo $u_{\min}$. Việc áp dụng *LPG* mà không dùng SF có thể dẫn đến việc tỉa nhầm một số nhánh như được minh họa trong mục III.3.

2. Minh họa thuật toán

Trong ví dụ này, chúng tôi minh họa quá trình khai thác các chuỗi sinh phổ biến lợi ích cao trên QSDB cho trong Bảng II bằng thuật toán *FGenHUSM* với $mu = 200$ và $ms = 1$. Để tiện trình bày, với chuỗi α, chúng tôi viết thêm các giá trị $u_{\min}$, RBU, LRU, supp, SE, SLIP và SF($u_{\min}$) của nó ở dạng $\alpha^{u_{\min},\text{RBU},\text{LRU}}_{\text{supp;SE,SLIP;SF}}$. Ký hiệu (_) cũng được dùng để dấu đi một hoặc một dãy các giá trị kề nhau.

Bảng II
QSDB $\mathcal{D}''$ MINH HỌA THUẬT TOÁN *FGenHUSM*

SID	Chuỗi
1	$\Psi'_1 = (a,2)(c,5)(e,6) \rightarrow (a,3)(b,6) \rightarrow (a,5)(d,50)$ $\rightarrow (a,5)(b,9)(c,40) \rightarrow (a,4)(c,10)(d,10)(f,36)$
2	$\Psi'_2 = (b,12) \rightarrow (a,2)(c,20)(e,6) \rightarrow (a,3)(d,20)$ $\rightarrow (a,1)(c,20)(d,10)(f,9) \rightarrow (a,4)(b,9)(c,15)$
3	$\Psi'_3 = (c,20) \rightarrow (a,4)(c,10)(e,4) \rightarrow (a,1)(f,18)$
4	$\Psi'_4 = (d,80) \rightarrow (a,7)(c,50)(e,6) \rightarrow (a,2)(g,2)$ $\rightarrow (a,9)(f,72)$

Tại dòng 2, ta có

$$S = I = \{a^{6,495,607}_{4;\_}, b^{15,306,322}_{2;\_}, c^{80,504,607}_{4;\_}, d^{100,480,550}_{3;10,3;\_}, e^{22,395,607}_{4;\_}, f^{135,163,607}_{4;\_}, g^{2,83,228}_{1;\_}\}.$$

Trước hết, dễ thấy rằng, các lần gọi *DfsFGHUS* trên các nhánh có gốc tại f và g sẽ dừng ngay tại dòng 3 (vì RBU(f) = 163 và RBU(g) = 83 < mu nên *UpdateFGHUS* trả về *true*). Nói cách khác, hai nhánh tại f và g được tỉa bằng *DPS*. Các chuỗi ứng với các nút a, b, c, d và e bị bỏ qua vì các giá trị $u_{\min}$ của chúng đều bé hơn mu, tuy nhiên, tìm kiếm trên chúng vẫn được tiếp tục. Sau khi kết thúc tìm kiếm trên a, b và c, ta có

$$\begin{aligned} FGHUS = \{&c \rightarrow f^{220}_4, ac \rightarrow a \rightarrow f^{211}_3, ce \rightarrow a \rightarrow f^{218}_3; \\ &ac \rightarrow ad \rightarrow c \rightarrow ac^{200}_2, ac \rightarrow ad \rightarrow cdf^{202}_2, \\ &ace \rightarrow d \rightarrow ac \rightarrow c^{202}_2, c \rightarrow ad \rightarrow ac \rightarrow ac^{202}_2, \\ &c \rightarrow ad \rightarrow acdf^{203}_2, ce \rightarrow ad \rightarrow c \rightarrow c^{200}_2, \\ &ec \rightarrow d \rightarrow c \rightarrow ac^{200}_2, ce \rightarrow d \rightarrow cdf^{200}_2\}, \end{aligned}$$

ta chỉ lưu lại các giá trị $u_{\min}$ (ở trên) và supp (ở dưới).

Bây giờ, ta xét quá trình khai thác trên nhánh $d^{100,480,550}_{3;\_}$. Sau khi hàm *UpdateFGHUS* trả về giá trị *false*, thực hiện các dòng 6–10, ta có $I_{\text{new}}(d) = \{f\}$. Các i–mở rộng da, db, dc và dd bị tỉa vì a, b, c và d không lớn hơn d. Hai i-mở rộng de và dg bị tỉa vì de, dg không xuất hiện trong QSDB $\mathcal{D}''$. Tương tự, ta có $S_{\text{new}}(d) = \{a,b,c,d,e,f,g\}$. Không có gì xảy ra khi thực hiện các dòng 6–12. Lúc này, ta gọi *DfsFGHUS* cho các s–mở rộng của d với mỗi thuộc tính trong $S_{\text{new}}(d)$ trước và sau đó cho i-mở rộng df của nó. Năm nhánh $d \rightarrow b^{\_,193,252}$, $d \rightarrow d^{\_,163,252}$, $d \rightarrow e^{\_,171,228}$, $d \rightarrow g^{\_,163,228}$ và $df^{\_,93,252}$ bị tỉa vì các giá trị RBU của chúng bé hơn mu. Ta chỉ còn lại ba nhánh $d \rightarrow a^{150,480,480}_{3;7,3;14}$, $d \rightarrow c^{215,458,480}_{3;7,3;25}$ và $d \rightarrow f^{267,295,480}_{3;4,3;29}$ cần xét với $S_{\text{new}}(d)$.

Xét việc thực thi *DfsFGHUS* với $d \rightarrow a\_$. Trước hết, sau các dòng 3–7, ta có: $I_{\text{new}}(d \rightarrow a) = \{b,c,d,e,f,g\}$. Sau

đó, ở cuối dòng 23, ta có $S_{\text{new}}(d \to a) = \{a, c, f, g\}$. Thật vậy, hai nhánh $d \to a \to b$ và $d \to a \to d$ bị tỉa bởi *WPS* vì LRU của chúng lần lượt là 83 và 164 đều bé hơn mu; còn chuỗi $d \to a \to e$ không xuất hiện trong $\mathcal{D}''$. Thủ tục gọi đệ quy trên các nhánh $d \to a \to a$, $d \to a \to c$, $d \to a \to f$ và $d \to a \to g$ với $S_{\text{new}}(d \to a)$ tại dòng 32; và cho các nhánh $d \to ab$, $d \to ac$, $d \to ad$, $d \to ae$, $d \to af$ và $d \to ag$ với $S_{\text{new}}(d \to a)$ và $I_{\text{new}}(d \to a)$ tại dòng 38.

Trên nhánh $d \to a \to a$, ta đưa chuỗi $d \to a \to af^{258}_{\_:2,\_}$ vào *FGHUS* (ở dòng 28 trong *UpdateFGHUS*). Các nhánh $d \to a \to c$ và $d \to a \to g$ bị tỉa bởi *DPS*. Trên nhánh $d \to a \to f$, ta thay thế $d \to a \to af$ bởi $d \to a \to f$ trong *FGHUS* (các dòng 22–27 trong *UpdateFGHUS*) vì chúng có cùng độ hỗ trợ mà $d \to a \to f \sqsubset d \to a \to af$.

Vì các giá trị RBU của $d \to ad$, $d \to ae$ và $d \to ag$ bé hơn mu, ta xét $d \to ab$, $d \to af$ và $d \to ac$. Với $d \to ab$, không có thay đổi nào trên *FGHUS*. Với $d \to af$ và $d \to ac$, ta đẩy $d \to af_3^{281}$ và $d \to ac_3^{230}$ vào *FGHUS*. Khi đó, ta có $I_{\text{new}}(d \to ac) = \{d, e, f\}$ và $S_{\text{new}}(d \to ac) = \{a, c, f, g\}$.

Không có thay đổi nào trên *FGHUS* khi xét $d \to ac \to c$ và $d \to ac \to f$. Với $d \to ac \to a$, ta tìm thấy trong *FGHUS* chuỗi con $d \to ac_3^{230}$ có cùng độ hỗ trợ tại dòng 14 trong *UpdateFGHUS*. Tại dòng tiếp theo, ta thực hiện thủ tục *LocalPruningGHU* với $d \to ac \to a$ và $d \to ac$. Mặc dù $\text{SE}(\mathcal{D}''_{d \to ac \to a}) = \text{SE}(\mathcal{D}''_{d \to ac}) = 7$, chiến lược tỉa *LPG* không thể được áp dụng vì

$$\text{SF}(u_{\min}(d \to ac)) = 29 < mu.$$

Tiếp tục, ta có

$$I_{\text{new}}(d \to ac \to a) = \{c, f, g\},$$
$$S_{\text{new}}(d \to ac \to a) = \{a, f\}.$$

Với s-mở rộng $d \to ac \to a \to a$, ta có $I_{\text{new}}(d \to ac \to a \to a) = \{f\}$. Vì $\text{RBU}(d \to ac \to a \to af) \geq mu$, tại dòng 13, ta gọi *LocalPruningGHU* (trường hợp (i)) với $i_{\text{new}} = d \to ac \to a \to af$ và $d \to ac \to a \to a$. Trong thủ tục ta không áp dụng được *LPG*, vì

$$\text{SF}(u_{\min}(d \to ac \to a \to a)) = 148 < mu.$$

Tiếp tục các dòng 14–15, trường hợp (ii) được xét giữa i_{new} với $\beta = d \to ac \to a \to f$. Vì

$$\begin{aligned}&\text{SE}(\mathcal{D}''_{d \to ac \to a \to f}) = \text{SLIP}(\mathcal{D}''_{d \to ac \to a \to af})\\ &\quad = \text{SE}(\mathcal{D}''_{d \to ac \to a \to f}) = \text{SLIP}(\mathcal{D}''_{d \to ac \to a \to f}) = 1,\\ &\text{SF}(u_{\min}(d \to ac \to a \to f)) = 211 \geq mu,\end{aligned}$$

ta tỉa toàn bộ nhánh *non-GHU branch*$(d \to ac \to a \to af)$ bởi *LPG*, cụ thể là đặt giá trị *false* cho trường *do-ext* của i_{new}.

Sau đó, ta xét s-mở rộng $d \to ac \to a \to f$ và chèn nó vào *FGHUS*. Với các i-mở rộng $d \to ac \to ac$ cũng như $d \to ac \to af$, không có thay đổi nào diễn ra vì $d \to ac \to af$ là chuỗi cha của chuỗi $d \to a \to f$ trong *FGHUS* với cùng độ hỗ trợ 2. Cuối cùng, với i-mở rộng $d \to ac \to ag$. Ta có, $I_{\text{new}}(d \to ac \to ag) = \varnothing$ và $S_{\text{new}}(d \to ac \to ag) = \{a, f\}$. Ở các dòng 26–30 trong *DfsFGHUS*, trường hợp (b) xảy ra với lời gọi *LocalPruningGHU*$(d \to ac \to ag \to f, d \to ac \to a \to f)$. Khi đó, ta tỉa toàn bộ nhánh $d \to ac \to ag \to f$ không chứa chuỗi sinh phổ biến lợi ích cao.

Quay trở lại nút $d \to ac$ và xét các i-mở rộng của nó, ta không tìm thấy lời giải nào, đồng thời cũng không có nhánh nào bị tỉa. Tuy nhiên, với $d \to ace$, ta tỉa được 7 nhánh không chứa chuỗi sinh và đưa thêm ứng viên $d \to ace \to f_1^{215}$ vào *FGHUS*.

Khi xét hai nhánh bắt đầu tại $d \to f_3^{267}$ và $d \to c_3^{215}$, ta chèn thêm chúng vào *FGHUS* đồng thời loại bỏ $d \to ace \to f_1^{215}$. Tìm kiếm trên $d \to f\_$ kết thúc và ta tiếp tục trên $d \to c\_$. Trong quá trình này, ta phát hiện ra 12 lần sử dụng chiến lược tỉa *LPG*, và 64 lần sử dụng *WPS* và *DPS*. Năm ứng viên sau được thêm vào *FGHUS*: $d \to ce \to f_1^{208}$, $d \to c \to a \to f_1^{204}$, $d \to c \to f_2^{328}$ và $d \to c \to g \to f_1^{204}$. Các ứng viên $d \to ac_3^{230}$, $d \to af_3^{231}$, $d \to ac \to a \to f_1^{211}$ và $d \to ace \to f_1^{215}$ bị loại vì $d \to c_3^{215}$, $d \to f_3^{267}$, $d \to c \to a \to f_1^{204}$ và $d \to ce \to f_1^{208}$ đã có trong *FGHUS* tương ứng.

Sau đó, quá trình tìm kiếm trên e, f, và g diễn ra. Đáng tiếc là, ta không thu được gì. Cuối cùng, ta có

$$\begin{aligned}FGHUS = \{&c \to f_4^{220}; ac \to a \to f_3^{211}, ce \to a \to f_3^{218},\\ &d \to f_3^{267}, d \to c_3^{215}; ac \to ad \to c \to ac_2^{200},\\ &ac \to ad \to cdf_2^{202}, ace \to d \to ac \to c_2^{202},\\ &c \to ad \to ac \to ac_2^{202}, c \to ad \to acdf_2^{203},\\ &ce \to ad \to c \to c_2^{200}, ce \to d \to c \to ac_2^{200},\\ &ce \to d \to cdf_2^{200}, d \to a \to f_2^{245},\\ &d \to c \to f_2^{328}; d \to ce \to f_1^{208},\\ &d \to c \to a \to f_1^{204}, d \to c \to g \to f_1^{204}\},\end{aligned}$$

với 18 lời giải, chiếm tỉ lệ 22% so với 83 chuỗi phổ biến lợi ích cao.

3. Thử nghiệm

Các thử nghiệm nhằm minh họa tính hiệu quả của *FGenHUSM* được tiến hành trên các cơ sở dữ liệu được mô tả trong Bảng III. Kosarak và Snake là hai QSDB thực tế, trong khi D4C7T5N5S6I4 và D0.5C10T15N2S6I4 là hai QSDB tổng hợp, với các giá trị về số lượng và lợi ích đơn vị của các thuộc tính trong các QSDB tương ứng được tạo ngẫu nhiên bởi chương trình sinh dữ liệu của IBM (IBM Quest data generator) từ thư viện SPMF [21].

Bảng III
QSDB $\mathcal{D}''$ MINH HỌA THUẬT TOÁN *FGenHUSM*

Tên	Số chuỗi	Số thuộc tính	Độ dài trung bình	Loại dữ liệu
Kosarak	10000	10094	8,14	Duyệt web
D4C7T5N5S6I4	4000	5000	28,7	Tổng hợp
D0.5C10T15N2S6I4	500	2000	127,7	Tổng hợp
Snake	163	20	60,6	Chuỗi protein

Với mỗi QSDB Q cố định, chúng tôi xác định một ngưỡng *ms* tương đối (tần suất tính theo % đã được dùng phổ biến trong các thực nghiệm cho các thuật toán về lĩnh vực này, $ms\% \stackrel{\text{def}}{=} ms * 100/|Q|(\%)$, trong đó $|Q|$ là số các q-chuỗi đầu vào của Q). Nếu không gây hiểu nhầm ta cũng ký hiệu *ms* tương đối là *ms*.

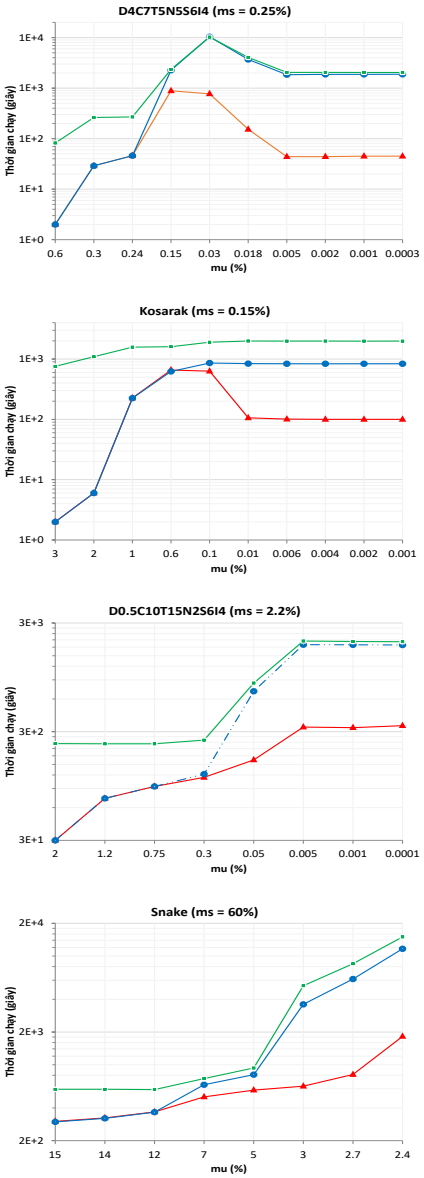
Chúng tôi đã tiến hành khai thác các chuỗi sinh phổ biến lợi ích cao bằng *FGenHUSM* trong ba trường hợp: (i) ***Non***: không dùng chiến lược tỉa nào (tìm các chuỗi phổ biến lợi ích cao trước và sau đó lọc ra các chuỗi sinh); (ii) ***WDPS***: sử dụng *WPS* và *DPS*; (iii) ***All***: dùng cả ba chiến lược *WPS*, *DPS* và *LPG*.

Trước hết, chúng tôi nhận thấy rằng hai tập kết quả của *FGenHUSM* luôn trùng nhau tương ứng với hai trường hợp: khi dùng cả ba chiến lược tỉa (*All*) và khi không dùng bất cứ chiến lược nào (*Non*) mà chỉ đơn thuần dùng định nghĩa của chuỗi sinh lợi ích cao phổ biến (Định nghĩa 5). Do đó, tính đúng của *FGenHUSM* đã được kiểm chứng thêm thông qua thực nghiệm.

Thời gian chạy của *FGenHUSM* được cho trong hình 1. Khi ngưỡng *mu* cao, các điều kiện tỉa $\text{LRU}(\alpha) < mu$ và $\text{RBU}(\alpha) < mu$ (trong hai chiến lược *WPS* và *DPS*, hay *WDPS*) dễ có cơ hội xảy ra hơn, nên số ứng viên (sinh ra và chưa bị tỉa) khi dùng *WDPS* là bé hơn đáng kể so với *Non* (không dùng chiến lược nào). Khi *mu* giảm dần, tác dụng của *WDPS* cũng giảm theo. Ngược lại, với các ngưỡng *mu* thấp, vì điều kiện $\text{SF}(\alpha) \geq mu$ (trong chiến lược *LPG* khi dùng *All*) dễ xảy ra hơn, nên số ứng viên sinh ra khi dùng *All* (áp dụng cả ba chiến lược tỉa nêu trên) là bé hơn đáng kể so với chỉ dùng *WDPS*. Vì vậy, so với *Non*, việc áp dụng đồng thời cả ba chiến lược tỉa sẽ tỉa nhiều hơn các chuỗi ứng viên, do đó thời gian thi hành của thuật toán sẽ nhanh hơn đáng kể.

Ghi nhận số lượng các ứng viên được sinh ra tương ứng trong hình 2 của ba trường hợp lý giải thêm về sự khác nhau về thời gian chạy của chúng.

Hình 3 chỉ ra số lượng các chuỗi FGHU khai thác được bởi *FGenHUSM* (*#fGenHU*) và số lượng các chuỗi FHU (*#fHU*). Có thể thấy rằng, *#fGenHU* bé hơn *#fHU* từ 5 đến 100 lần theo trung bình (đặc biệt khi *mu* bé). Vì vậy, theo

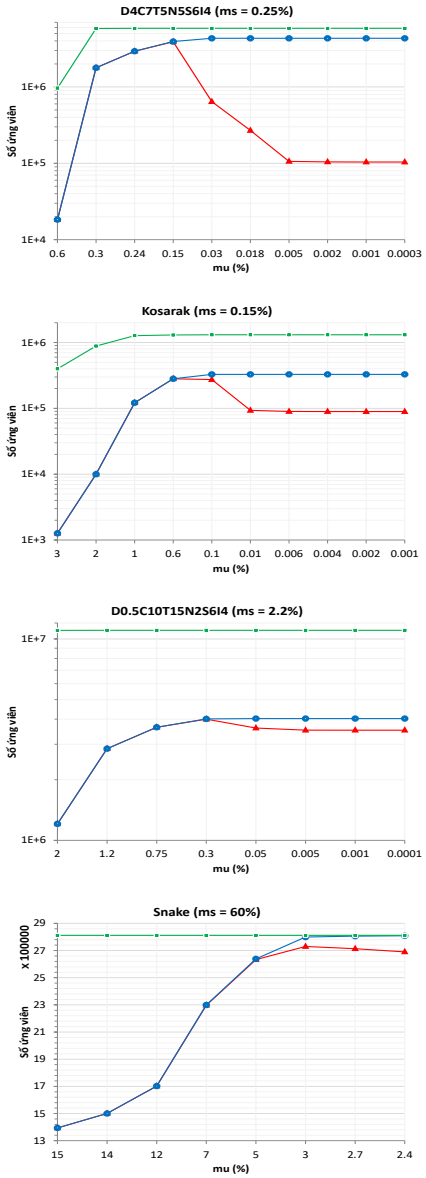


Hình 1. Thời gian chạy của *FGenHUSM*. Ghi chú: *All* (màu đỏ, hình tam giác), *WDPS* (màu xanh đậm, hình tròn), *Non* (màu xanh lá cây, hình vuông).

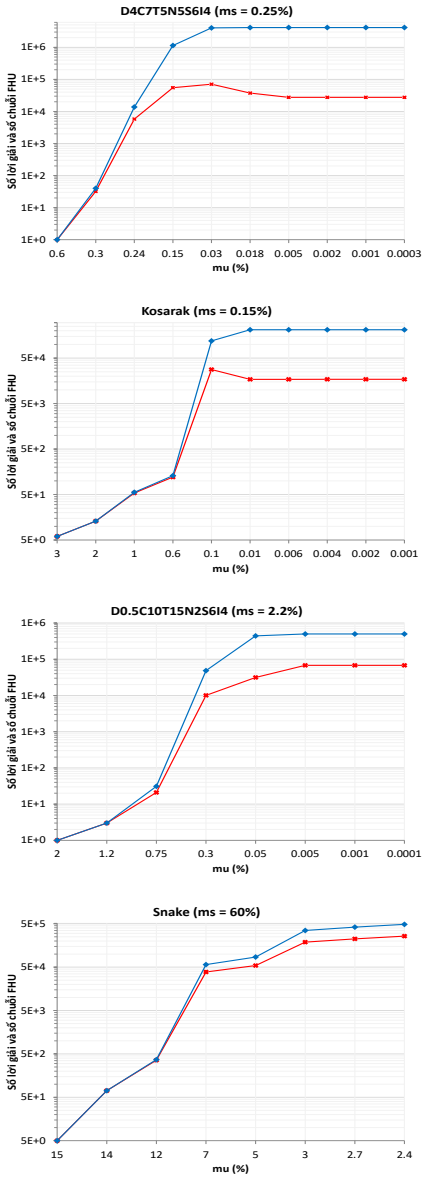
lực lượng, có thể xem *FGHUS* là một biểu diễn súc tích của *FHUS*.

V. KẾT LUẬN

Bài báo này đã đề xuất khái niệm tập *FGHUS* các chuỗi sinh phổ biến lợi ích cao và thuật toán hiệu quả *FGenHUSM* khai thác chúng, thông qua các kỹ thuật tỉa các nhánh ứng viên trên cây tìm kiếm tiền tố. Trước hết, hai chiến lược tỉa theo chiều rộng *WPS* và chiều sâu *DPS* (đã dùng trong các kết quả trước đây cho việc tỉa các chuỗi lợi ích thấp) được điều chỉnh phù hợp để tỉa các chuỗi ít



Hình 2. Số ứng viên sinh bởi *FGenHUSM*. Ghi chú: *All* (màu đỏ, hình tam giác), *WDPS* (màu xanh đậm, hình tròn), *Non* (màu xanh lá cây, hình vuông).



Hình 3. Số lời giải và số chuỗi FHU. Ghi chú: *#fGenHU* (màu xanh đậm, hình thoi), *#fHU* (màu đỏ, hình dấu nhân).

phổ biến hoặc các chuỗi lợi ích thấp. Sau đó, chúng tôi đã chỉ ra một chặn dưới SF của độ đo lợi ích tối thiểu $u_{\min}$. Dựa vào SF và điều kiện tỉa sớm tổng quát [13], chiến lược tỉa *LPG* được đề xuất và dùng để tỉa các ứng viên không là chuỗi sinh phổ biến lợi ích cao. Để ý rằng, *WPS* và *DPS* có tác dụng tỉa mạnh với các ngưỡng lợi ích tối thiểu *mu* cao, trong khi *LPG* có hiệu quả cao với các *mu* thấp. Do đó, chúng tôi đã tích hợp tất cả chúng vào thuật toán *FGenHUSM*. Thực nghiệm trên bốn cơ sở dữ liệu lớn (thực tế lẫn tổng hợp) đã chỉ ra rằng *FGenHUSM* khai thác nhanh *FGHUS* – một biểu diễn súc tích của tập các chuỗi phổ biến lợi ích cao.

LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi Quỹ Phát triển Khoa học và Công nghệ Quốc gia (NAFOSTED) trong đề tài mã số 102.05-2017.300.

TÀI LIỆU THAM KHẢO

[1] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, “Mining high utility web access sequences in dynamic web log data,” in *11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2010, pp. 76–81.

[2] B.-E. Shie, H.-F. Hsiao, V. S. Tseng, and S. Y. Philip, “Mining high utility mobile sequential patterns in mobile

commerce environments," in *International Conf. Database Systems for Advanced Applications*, 2011, pp. 224–238.
[3] M. Zihayat, H. Davoudi, and A. An, "Top-k utility-based gene regulation sequential pattern discovery," in *IEEE International Conference on Bioinformatics and Biomedicine*, 2016, pp. 266–273.
[4] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "A novel approach for mining high-utility sequential patterns in sequence databases," *ETRI Journal*, vol. 32, no. 5, pp. 676–686, 2010.
[5] J. Yin, Z. Zheng, and L. Cao, "USpan: An efficient algorithm for mining high utility sequential patterns," in *ACM/SIGKDD Int'l Conf. Knowl. Disc. Data Mining*, 2012, pp. 660–668.
[6] J. C.-W. Lin, J. Zhang, and P. Fournier-Viger, "High-utility sequential pattern mining with multiple minimum utility thresholds," in *Asia-Pacific Web and Web-Age Infor. Management Joint Conf. Web and Big Data*, 2017, pp. 215–229.
[7] T. Truong, A. Tran, H. Duong, B. Le, and P. Fournier-Viger, "EHUSM: Mining high utility sequences with a pessimistic utility model," *1st Int'l Work. Utility-Driven Mining*, 2018.
[8] T. C. Tin, T. N. Anh, D. Van Hai, and L. H. Bac, "HUPSMT: An efficient algorithm for mining high utility-probability sequences in uncertain databases with multiple minimum utility thresholds," *Journal of Computer Science and Cybernetics*, vol. 35, no. 1, pp. 1–20, 2019.
[9] T. Truong-Chi and P. Fournier-Viger, "A survey of high utility sequential pattern mining," in *High-Utility Pattern Mining*, P. Fournier-Viger, J. Lin, R. Nkambou, B. Vo, and V. Tseng, Eds. Springer, 2019, pp. 97–129.
[10] W. Gan, J. C.-W. Lin, J. Zhang, H.-C. Chao, H. Fujita, and S. Y. Philip, "ProUM: Projection-based utility mining on sequence data," *Info. Sciences*, vol. 513, pp. 222–240, 2020.
[11] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining closed sequential patterns in large datasets," in *SIAM International Conference on Data Mining*, 2003, pp. 166–177.
[12] B. Le, H. Duong, T. Truong, and P. Fournier-Viger, "FCloSM, FGenSM: Two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy," *Knowledge and Information Systems*, vol. 53, no. 1, pp. 71–107, 2017.
[13] T. Truong, H. Duong, B. Le, and P. Fournier-Viger, "FMax-CloHUSM: An efficient algorithm for mining frequent closed and maximal high utility sequences," *Eng. Applications of Artificial Intelligence*, vol. 85, pp. 1–20, 2019.
[14] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin, "Efficient vertical mining of frequent closures and generators," in *Int'l Symp. Intelligent Data Analysis*, 2009, pp. 393–404.
[15] A. Tran, T. Truong, and B. Le, "Simultaneous mining of frequent closed itemsets and their generators: Foundation and algorithm," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 64–80, 2014.
[16] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Novel concise representations of high utility itemsets using generator patterns," in *International Conference on Advanced Data Mining and Applications*, 2014, pp. 30–43.
[17] P. Fournier-Viger, A. Gomariz, M. Šebek, and M. Hlosta, "VGEN: Fast vertical mining of sequential generator patterns," in *International Conference on Data Warehousing and Knowledge Discovery*, 2014, pp. 476–488.
[18] H. Duong, T. Truong, and B. Le, "Efficient algorithms for simultaneously mining concise representations of sequential patterns based on extended pruning conditions," *Eng. Applications of Artificial Intelligence*, vol. 67, pp. 197–210, 2018.
[19] P. D. Grünwald, I. J. Myung, and M. A. Pitt, *Advances in minimum description length: Theory and applications*. MIT press, 2005.
[20] M.-T. Tran, B. Le, B. Vo, and T.-P. Hong, "Mining non-redundant sequential rules with dynamic bit vectors and pruning techniques," *Applied Intelligence*, vol. 45, no. 2, pp. 333–342, 2016.
[21] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: A Java open-source pattern mining library," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

Trương Chí Tín là giảng viên tại Khoa Toán – Tin học, Trường Đại học Đà Lạt. Tác giả tốt nghiệp Cử nhân Toán tại Trường Đại học Đà Lạt năm 1983 và nhận bằng Tiến sĩ về Điều khiển tối ưu ngẫu nhiên năm 1990 tại Đại học Quốc gia Hà Nội. Hướng nghiên cứu hiện nay của tác giả là trí tuệ nhân tạo và khai thác dữ liệu.

Trần Ngọc Anh đang giảng dạy và nghiên cứu tại Khoa Toán – Tin học, Trường Đại học Đà Lạt. Tác giả tốt nghiệp Đại học ngành Toán – Tin học vào năm 1999 tại Đại học Đà Lạt, nhận bằng Thạc sĩ và Tiến sĩ về Khoa học máy tính vào các năm 2004 và 2016 tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh. Tác giả đang nghiên cứu về khai thác dữ liệu và trí tuệ nhân tạo.

Dương Văn Hải tốt nghiệp Trường Đại học Đà Lạt ngành Tin học năm 2004. Tác giả tốt nghiệp Cao học ngành Công nghệ thông tin năm 2009 và đang là Nghiên cứu sinh tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh. Tác giả hiện đang giảng dạy và nghiên cứu về khai thác dữ liệu tại Khoa Toán – Tin học, Trường Đại học Đà Lạt.

Lê Hoài Bắc nhận bằng Cử nhân Toán năm 1984, Cao học Khoa học máy tính năm 1990 và hoàn thành Luận án Tiến sĩ về Đảm bảo Toán học cho máy tính năm 2000 tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh. Tác giả hiện là giảng viên tại Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Tp. Hồ Chí Minh và đang nghiên cứu về trí tuệ nhân tạo, tính toán mềm, khai thác dữ liệu và khoa học dữ liệu.