Chương 1 - Giới thiệu

Từ khi xuất hiện đầu tiên vào năm 1995 [2] đến nay, bài toán khai thác các chuỗi phổ biến (hay mẫu tuần tự) trên những cơ sở dữ liệu chuỗi (sequence database, *SDB*) đã thu hút sự quan tâm của rất nhiều nhà nghiên cứu vì nó có nhiều ứng dụng thực tế (do dữ liệu thường được thể hiện dưới dạng các chuỗi ký hiệu) trong nhiều lĩnh vực như e-learning [22], phân tích hành vi con người [12], khai thác quá trình [73], phân tích lỗi phần mềm [28], y – tin học [9,43], phân tích văn bản [68] và luồng kích chuột trên web [20]. Bài toán này nhằm mục đích khám phá tập $\mathcal{FS}$ (frequent sequence) chứa tất cả các chuỗi thường xuyên xuất hiện trong *SDB*, nghĩa là số lần xuất hiện của mỗi chuỗi không nhỏ hơn một ngưỡng $minsupp$ cho trước. Cho đến nay, nhiều thuật toán đã được đề xuất để khai thác tập $\mathcal{FS}$, chẳng hạn *PrefixSpan* [63], *GSP* [81], *PSP* [52], *SPADE* [97], *SPAM* [7], *CM-SPADE* và *CM-SPAM* [17].

Bài toán khai thác chuỗi phổ biến với ràng buộc. Khó khăn lớn nhất của bài toán khai thác chuỗi phổ biến là lực lượng của tập kết quả $\mathcal{FS}$ thường quá lớn làm cho các thuật toán trước đây để khai thác nó thường kém hiệu quả. Đặc biệt, người dùng thường cảm thấy không thuận tiện và gặp nhiều khó khăn để hiểu và phân tích tập $\mathcal{FS}$. Do đó, các nhà nghiên cứu đã tập trung trên bài toán khác có nhiều ý nghĩa thực tế hơn, đó là khai thác chỉ các chuỗi phổ biến thỏa những ràng buộc do người dùng đưa vào. Cho trước *SDB* $\mathcal{D}$, ngưỡng hỗ trợ tối thiểu $minsupp$ và ràng buộc $\mathcal{C}$ được định nghĩa bởi người dùng, bài toán khai thác chuỗi phổ biến với ràng buộc là tìm tập $\mathcal{FS}^{\mathcal{C}}$ gồm tất cả các chuỗi phổ biến có trong $\mathcal{D}$ thỏa ràng buộc $\mathcal{C}$. Cụ thể, $\mathcal{FS}^{\mathcal{C}}$ được định nghĩa như sau:

$$\mathcal{FS}^{\mathcal{C}} \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS} \mid \alpha \text{ thỏa } \mathcal{C}\}, \text{ với } \mathcal{FS} \stackrel{\text{def}}{=} \{\alpha \mid supp(\alpha) \geq minsupp\},$$

trong đó ký hiệu $supp(\alpha)$ là độ hỗ trợ của chuỗi α (hay số chuỗi đầu vào trong $\mathcal{D}$ chứa α). $\mathcal{C}$ có thể là loại ràng buộc bất kỳ trong số những loại ràng buộc được định nghĩa trong [65] và có thể thuộc vào hai lớp ràng buộc thông dụng là lớp ràng buộc có tính đơn điệu giảm (C^A) và lớp ràng buộc có tính đơn điệu tăng (C^M) [58,62,61,7]. Một chuỗi α được gọi là thỏa ràng buộc C bất kỳ thuộc C^A (hoặc C^M) thì mọi chuỗi con (hoặc chuỗi cha) của nó cũng thỏa C.

Động cơ của luận án. Đa phần các nghiên cứu trước đây để khai thác tập $\mathcal{FS}^{\mathcal{C}}$ sử dụng phương pháp tích hợp C vào các thuật toán khai thác chuỗi phổ biến đã biết để có thể rút gọn nhanh không gian tìm kiếm nhằm tránh xét mọi chuỗi có thể. Tuy nhiên, theo nhận xét của luận án, các thuật toán dựa trên phương pháp tiếp cận này có một số nhược điểm chính sau. *Thứ nhất*, chúng phải đọc lại *SDB* (thường rất lớn) mỗi khi ràng buộc C thay đổi (do chúng khai thác $\mathcal{FS}^{\mathcal{C}}$ trực tiếp từ *SDB*). *Thứ hai*, chúng thường phải kiểm tra ràng buộc C trên một số lượng lớn các chuỗi ứng viên được sinh ra, đặc biệt khi C thuộc lớp C^M (do lớp ràng buộc này khó có thể được sử dụng hiệu quả để rút gọn nhanh không gian tìm kiếm). *Thứ ba*, các thuật toán này thường được thiết kế cho một loại ràng buộc cụ thể và bởi vậy chúng khó có thể được mở rộng cho các loại ràng buộc khác. *Thứ tư,* việc thiết kế các thuật toán song song để khai thác tập $\mathcal{FS}^{\mathcal{C}}$ trên những dữ liệu lớn thường gặp nhiều thách thức. Bởi vậy, việc đề xuất các phương pháp và thuật toán mới mà có thể khắc phục được các hạn chế này là thật sự cần thiết, góp phần nâng cao hiệu quả của việc khai thác tập $\mathcal{FS}^{\mathcal{C}}$.

Mục tiêu của luận án. Trước những khó khăn của các phương pháp tiếp cận trước đây để khai thác tập $\mathcal{FS}^{\mathcal{C}}$, ba câu hỏi nghiên cứu được đặt ra như sau:

(Q_1) Thay vì kiểm tra ràng buộc C trên một số lượng rất lớn các chuỗi được sinh ra, ta chỉ cần kiểm tra ràng buộc trên *một số lượng nhỏ các chuỗi đặc biệt* (chẳng hạn các chuỗi đóng

và chuỗi sinh phổ biến không thỏa ràng buộc), nhưng có thể loại nhanh một số lượng rất lớn các chuỗi không thỏa ràng buộc mà không cần sinh ra chúng?

(Q_2) Nếu các chuỗi đặc biệt thỏa ràng buộc C, ta có thể sinh ra nhanh tất cả các chuỗi thỏa mãn C chỉ từ những chuỗi đặc biệt này mà không cần đọc lại *SDB*?

(Q_3) Ta có thể phát triển các phương pháp chung hiệu quả để có thể khai thác các chuỗi phổ biến với nhiều loại ràng buộc thuộc lớp C^A hoặc C^M, và đồng thời các phương pháp này cho phép thiết kế dễ dàng các thuật toán song song?

Mục tiêu của luận án là nghiên cứu và đề xuất những kết quả lý thuyết cũng như các thuật toán nhằm đưa ra những câu trả lời khẳng định cho ba câu hỏi này.

Phương pháp tiếp cận của luận án. Để đạt được mục tiêu, phương pháp tiếp cận của luận án là khai thác tập $\mathcal{FS}^C$ từ hai tập biểu diễn súc tích của nó là tập $\mathcal{FCS}$ (gồm tất cả các chuỗi đóng phổ biến) và tập $\mathcal{FGS}$ (gồm tất cả các chuỗi sinh phổ biến), thay vì khai thác $\mathcal{FS}^C$ trực tiếp từ *SDB* như các phương pháp trước đây. Phương pháp này gồm các bước chính sau: (1) khai thác chỉ một lần cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) từ *SDB* ứng với một ngưỡng $minsupp$ cho trước; (2) phân hoạch tập $\mathcal{FS}$ thành các lớp tương đương dựa trên một quan hệ tương đương phù hợp, mỗi lớp được đại diện bởi một số chuỗi đóng trong tập $\mathcal{FCS}$ và chuỗi sinh trong tập $\mathcal{FGS}$; (3) kiểm tra ràng buộc C chỉ trên các chuỗi đại diện (với số lượng bé) của mỗi lớp nhằm loại bỏ nhanh lớp đó (nếu các phần tử đại diện của nó không thỏa ràng buộc) mà không cần sinh ra tất cả các chuỗi phổ biến còn lại của lớp này; (4) sinh ra nhanh tất cả các chuỗi phổ biến thỏa C từ những chuỗi đại diện của mỗi lớp mà không cần truy cập *SDB*. Khi ràng buộc C thay đổi, phương pháp tiếp cận này chỉ cần truy cập lại phân hoạch của tập $\mathcal{FS}$ đã được thực hiện trong bước 2 để tìm tập lời giải $\mathcal{FS}^C$, trong khi các phương pháp tiếp cận khác phải đọc lại *SDB*.

Nội dung của luận án. Từ phương pháp tiếp cận của luận án, nhằm khai thác $\mathcal{FS}^C$ với hiệu quả cao, nội dung của luận án bao gồm ba bài toán chính sau:

- *Bài toán 1* ($\mathcal{BT}_1$): *Nghiên cứu và đề xuất các thuật toán hiệu quả để khai thác tuần tự và song song các chuỗi đóng và/hoặc chuỗi sinh phổ biến* (bước 1).
- *Bài toán 2* ($\mathcal{BT}_2$): *Nghiên cứu và chỉ ra cấu trúc hay biểu diễn tường minh của các chuỗi phổ biến trong tập $\mathcal{FS}$ dựa trên các chuỗi đóng và chuỗi sinh trong hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$. Từ cấu trúc lý thuyết này, ta có thể dẫn xuất nhanh tập $\mathcal{FS}$ từ hai tập này* (bước 2).
- *Bài toán 3* ($\mathcal{BT}_3$): *Phát triển các phương pháp hiệu quả để khai thác $\mathcal{FS}^C$ từ các chuỗi đóng-sinh trong cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) thay vì trực tiếp từ SDB sử dụng mối quan hệ tường minh được đưa ra trong $\mathcal{BT}_2$* (bước 3 và 4).

Các đóng góp chính (và bố cục) của luận án. Những đóng góp chính cho $\mathcal{BT}_1$ là.

(1) *Chỉ ra sự không chính xác trong trường hợp tổng quát của một số kết quả lý thuyết đã được đưa ra trong bài báo* [91] *năm 2003* (*xem* [8]*, Chương 2*).

(2) *Đề xuất số đo SE mới, các điều kiện đủ chính xác để tỉa sớm nhiều chuỗi không là đóng và/hoặc không là sinh dựa trên SE và các chiến lược tỉa địa phương hiệu quả hơn nhằm rút gọn nhanh không gian tìm kiếm (xem* [8,35]*, Chương 2)*.

(3) *Thiết kế các thuật toán FCloSM, FGenSM* [8]*, FGenCloSM* [35] *và Par-GenCloSM* [34] *để khai thác hiệu quả tập $\mathcal{FCS}$ và/hoặc tập $\mathcal{FGS}$* – thông tin cốt lõi phục vụ cho việc giải $\mathcal{BT}_2$ và $\mathcal{BT}_3$ (*Chương 2*).

Đóng góp chính của luận án cho bài toán $\mathcal{BT}_2$ là:

(4) *Chỉ ra cấu trúc hay biểu diễn tường minh của tập các chuỗi phổ biến $\mathcal{FS}$ dựa trên cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) và đồng thời đã thiết kế thuật toán FS-Miner để kiểm chứng lại tính đúng của cấu trúc được đưa ra thông qua thực nghiệm (xem* [36]*, Chương 3)*.

Đóng góp chính của luận án cho bài toán $\mathcal{BT}_3$ là:

(5) *Luận án đã phát triển các phương pháp chung để khai thác hiệu quả các chuỗi phổ biến với tất cả các loại ràng buộc thuộc hai lớp ràng buộc thông dụng C^A và C^M dựa trên mô hình được đề xuất trong Hình 1.1 và đóng góp (4). Ngoài ra, luận án cũng đã đề xuất thêm một kỹ thuật khử ràng buộc cho loại ràng buộc chuỗi cha (có nhiều ứng dụng thực tế) thuộc lớp C^M (xem* [32], *Chương 4).*

(6) *Luận án đã thiết kế hai thuật toán tổng quát MFS-AC và MFS-MC để khai thác các chuỗi phổ biến với các loại ràng buộc khác nhau thuộc lớp C^A và C^M (Chương 4)*.

Những kết quả chính của ba đóng góp (1) – (3) đã được công bố trong [8,34,35] và các đóng góp (4) – (6) đã được công bố trong [32,36].

Tất cả các kết quả lý thuyết được đề xuất trong luận án đều được chứng minh chặt chẽ trước khi được chuyển thành các thuật toán tương ứng và kiểm chứng lại bằng thực nghiệm.

Chương 2 - Khai thác các chuỗi đóng và chuỗi sinh phổ biến

Dựa trên mục tiêu và phương pháp tiếp cận của luận án được trình bày trong *Chương 1*, chương này quan tâm đến các phương pháp hiệu quả để giải quyết bài toán $\mathcal{BT}_1$. Để khai thác hiệu quả cặp ($\mathcal{FCS}$, $\mathcal{FGS}$), vấn đề quan trọng là ta cần rút gọn nhanh không gian tìm kiếm với chi phí về thời gian tính toán ít nhất có thể. Một ý tưởng đã được ra trong [91] là sử dụng một số đo *I* (được tính toán nhanh) và những điều kiện tỉa sớm dựa trên nó để rút gọn nhanh không gian tìm kiếm thông qua việc loại bỏ nhiều ứng viên thừa mà không cần sinh ra chúng. Nhiều thuật toán sau đó đã sử dụng số đo *I* hoặc các cấu trúc dữ liệu mới để khai thác hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ [17,19,25,27,88]. Tuy nhiên, hiệu quả của chúng không cao vì vẫn còn quá nhiều ứng viên thừa được sinh ra. Hơn nữa, đáng chú ý rằng các điều kiện tỉa sớm dựa trên *I* là không đúng trong trường hợp tổng quát, dẫn đến nhiều chuỗi đóng bị thiếu trong tập kết quả $\mathcal{FCS}$.

Mục tiêu của chương này là đưa ra các kết quả lý thuyết mới và các thuật toán tương ứng nhằm khắc phục các hạn chế trên, qua đó giải quyết hiệu quả hơn bài toán $\mathcal{BT}_1$.

2.1. Các khái niệm cơ sở và bài toán khai thác chuỗi đóng và sinh

2.1.1. Các khái niệm cơ sở

Định nghĩa 2.1 (*Cơ sở dữ liệu chuỗi SDB, 1-SDB, n-SDB*).

Đặt $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ là tập tất cả các thuộc tính. Một tập con A của $\mathcal{A}$, $A \subseteq \mathcal{A}$, được gọi là tập thuộc tính. Không mất tính tổng quát, ta giả sử rằng các thuộc tính trong một tập thuộc tính được sắp xếp theo một quan hệ thứ tự toàn phần, chẳng hạn như thứ tự từ điển $\prec$. Một danh sách có thứ tự các tập thuộc tính (hay các sự kiện) $\alpha = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_p$ (hay $\alpha = <E_1, E_2, \ldots, E_p>$) được gọi là một chuỗi, trong đó $\emptyset \neq E_k \subseteq \mathcal{A}$, $\forall k = 1,\ldots, p$ và p là một số nguyên dương. Chuỗi α được gọi là rỗng (null) nếu tất cả các sự kiện $E_k = \emptyset$, $\forall k = 1,\ldots, p$, và khi đó nó được ký hiệu là $\alpha = <>$. Một cơ sở dữ liệu chuỗi (ký hiệu là *SDB*) $\mathcal{D}$ gồm một tập hữu hạn các chuỗi đầu vào: $\mathcal{D} = \{\Psi_i, i = 1, ..., N\}$. Ta ký hiệu $size(\alpha) = p$ và $length(\alpha) = \sum_{k=1}^{p} |E_k|$ lần lượt là kích thước và chiều dài của chuỗi α. Một *1-SDB* (hay *n-SDB*) là một *SDB* trong đó

mỗi sự kiện của các chuỗi đầu vào chứa chỉ một thuộc tính (hay ít nhất một thuộc tính tương ứng). Để cho dễ đọc, quy ước các ký hiệu được sử dụng trong luận án được cho trong *Bảng 2.1*. Cho $\alpha = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_p$, $\beta = E'_1 \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_q$ là hai chuỗi tùy ý.

Định nghĩa 2.2 (*Quan hệ thứ tự $\sqsubseteq$ và toán tử ρ*). Xét quan hệ thứ tự $\sqsubseteq$ và toán tử ρ trên tập các chuỗi:

a) $\alpha \sqsubseteq \beta$ (hay $\beta \sqsupseteq \alpha$) $\Leftrightarrow p \leqslant q$ và tồn tại p số nguyên dương: $1 \leqslant j_1 < j_2 < \ldots < j_p \leqslant q$ và $E_k \subseteq E'_{j_k}$, $\forall k = 1,\ldots, p$. Khi đó, ta nói rằng α là chuỗi con của β, hay β là chuỗi cha của α, hay β chứa α; và $\alpha \sqsubset \beta \Leftrightarrow (\alpha \sqsubseteq \beta \wedge \alpha \neq \beta)$.

b) $\rho(\alpha) \stackrel{\text{def}}{=} \{\Psi \in \mathcal{D} \mid \Psi \sqsupseteq \alpha\}$ – tập các chuỗi đầu vào chứa α và $\rho(<>) = \mathcal{D}$.

Định nghĩa 2.3 (*Chuỗi phổ biến, chuỗi sinh và chuỗi đóng*).

Độ hỗ trợ của chuỗi α, ký hiệu là $supp(\alpha)$, được định nghĩa là số chuỗi cha (trong $\mathcal{D}$) của α, nghĩa là $supp(\alpha) = |\rho(\alpha)|$. Chuỗi α được gọi là chuỗi phổ biến (hay mẫu tuần tự) nếu độ hỗ trợ của nó không nhỏ hơn một ngưỡng $minsupp$ dương cho trước, nghĩa là $supp(\alpha) \geqslant minsupp$. Chuỗi α được gọi là *chuỗi sinh* nếu không có chuỗi con nào của α có cùng độ hỗ trợ với nó, nghĩa là $\nexists \beta$: $\beta \sqsubset \alpha$ và $supp(\beta) = supp(\alpha)$. Chuỗi α được gọi là *chuỗi đóng* nếu không có chuỗi cha nào của α có cùng độ hỗ trợ với nó, nghĩa là $\nexists \beta$: $\beta \sqsupset \alpha$ và $supp(\beta) = supp(\alpha)$.

Định nghĩa 2.4 (*Hai toán tử mở rộng, tiền tố và cơ sở dữ liệu chiếu*).

Mở rộng tập thuộc tính (hay ngắn gọn là mở rộng *i-ext*) của α với β, sao cho $\forall a \in E_p$, $\forall b \in E'_1$, $a \prec b$, là một chuỗi được ký hiệu và định nghĩa như sau: $\alpha \diamond_i \beta = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow (E_p \cup E'_1) \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_q$. Mở rộng chuỗi (hay ngắn gọn là mở rộng *s-ext*) của α với β (được ký hiệu là $\alpha \diamond_s \beta$) là chuỗi $\alpha \diamond_s \beta = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_p \rightarrow E'_1 \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_q$. Mở rộng α với β, ký hiệu là $\gamma = \alpha \diamond \beta$, có thể là $\alpha \diamond_i \beta$ hoặc $\alpha \diamond_s \beta$. Nếu chuỗi $\gamma = \alpha \diamond \beta$, thì α được gọi là tiền tố của γ và β được gọi là hậu tố của γ đối với α. Hơn nữa, nếu δ là *tiền tố nhỏ nhất* (của γ theo quan hệ $\sqsubseteq$) chứa α, khi đó ta ký hiệu $\delta = prefix(\gamma, \alpha)$, và *hậu tố* β của γ đối với δ ($\gamma = \delta \diamond \beta$) được ký hiệu là $suf(\gamma, \alpha)$. Cơ sở dữ liệu chiếu (gọi tắt là *PDB*) theo α được ký hiệu và định nghĩa là $\mathcal{D}_\alpha = \{suf(\Psi, \alpha) \mid \Psi \in \mathcal{D} \wedge \Psi \sqsupseteq \alpha\}$. Ta nói rằng, $\mathcal{D}_\beta$ chứa trong $\mathcal{D}_\alpha$, ký hiệu $\mathcal{D}_\beta \sqsubseteq \mathcal{D}_\alpha$, nếu ($\rho(\beta) \subseteq \rho(\alpha)$ và $\forall \Psi \in \rho(\beta)$, $suf(\Psi, \beta) \sqsubseteq suf(\Psi, \alpha)$), và $\mathcal{D}_\beta = \mathcal{D}_\alpha$ nếu ($\mathcal{D}_\beta \sqsubseteq \mathcal{D}_\alpha$ và $\mathcal{D}_\alpha \sqsubseteq \mathcal{D}_\beta$). Chú ý rằng $supp(\alpha) = |\mathcal{D}_\alpha|$. *Bảng 2.2* trình bày một *SDB* mẫu với $minsupp = 1$, trong đó ta ký hiệu các itemset $\{a, c, e\}$ hoặc $\{b\}$ một cách ngắn gọn như ace, b.

Bảng 2.2. Một *SDB mẫu*.

SID	Chuỗi đầu vào
Ψ_1	*ace→ab→ad→abc→acdf*
Ψ_2	*b→ace→ad→acdf→abc*
Ψ_3	*c→ace→af*
Ψ_4	*d→ace→ag→af*

2.1.2. Bài toán khai thác chuỗi đóng và chuỗi sinh

Cho trước một *SDB* $\mathcal{D}$ và giá trị của $minsupp$ được định nghĩa bởi người dùng, bài toán khai thác chuỗi sinh phổ biến hoặc chuỗi đóng phổ biến là khám phá tất cả các chuỗi tương ứng trong tập $\mathcal{FGS}$ hoặc $\mathcal{FCS}$, trong đó $\mathcal{FGS} = \{\alpha \mid supp(\alpha) \geqslant minsupp \wedge (\nexists \beta: \beta \sqsubset \alpha \wedge supp(\beta) = supp(\alpha))\}$ và $\mathcal{FCS} = \{\alpha \mid supp(\alpha) \geqslant minsupp \wedge (\nexists \beta: \beta \sqsupset \alpha \wedge supp(\beta) = supp(\alpha))\}$. Chú ý rằng, lực lượng của $\mathcal{FS}$ thường lớn hơn nhiều so với lực lượng của $\mathcal{FGS}$ và $\mathcal{FCS}$, đặc biệt trên những *n-SDB* hay *1-SDB* với các giá trị $minsupp$ bé. *Ví dụ*, xét *SDB* $\mathcal{D}$ trong *Bảng 2.2*, trong

đó có 20,054 chuỗi phổ biến (bao gồm cả chuỗi rỗng <>), nhưng chỉ có 20 (0.09%) chuỗi đóng và 154 (0.77%) chuỗi sinh.

2.2. Các nghiên cứu liên quan

Nhiều thuật toán đã được đề xuất đề khai thác tập $\mathcal{FCS}$ [17,24,27,54,82,88,91] và tập $\mathcal{FGS}$ [19,25,49,67]. Tuy nhiên, hiệu quả của các thuật toán này vẫn chưa cao [17,19]. Theo luận án, lý do là các thuật toán vẫn sinh ra nhiều ứng viên thừa, lặp lại phép chiếu cơ sở dữ liệu nhiều lần, và chưa phát hiện thật sớm và tỉa kịp thời các ứng viên ngay khi chúng vừa được sinh ra cùng các hậu duệ của chúng trên cây tiền tố. Mặc dù thuật toán *CloSpan* và các thuật toán sau đó sử dụng số đo đặc trưng *I* [91] cho *PDB* để nhận biết nhanh sự tương đương của các *PDB* và tỉa sớm các chuỗi không là đóng hoặc không là sinh, nhưng không may là những kết quả lý thuyết được dùng trong *CloSpan* là không hoàn toàn chính xác (điều này đã được luận án chỉ ra trong *Ví dụ 2.2*). Ngoài ra, trong các thuật toán ta vẫn phải kiểm tra quan hệ chuỗi con (thường mất nhiều thời gian) khi áp dụng các điều kiện tỉa các ứng viên. Do đó, hầu hết chúng vẫn mất nhiều thời gian xử lý để kiểm tra quan hệ chuỗi con và loại bỏ nhiều ứng viên thừa cũng như tiêu thụ nhiều bộ nhớ để lưu trữ những ứng viên này. Vì vậy, việc đưa ra một số đo mới đặc trưng cho các *PDB*, các điều kiện chính xác trong trường hợp tổng quát để rút gọn nhanh không gian tìm kiếm dựa trên số đo này, và các chiến lược tỉa sớm nhiều ứng viên không hứa hẹn, cũng như là việc thiết kế những thuật toán hiệu quả hơn để khai thác các chuỗi đóng và/hoặc chuỗi sinh vẫn là những công việc thật sự cần thiết và có nhiều ý nghĩa.

2.3. Các kết quả lý thuyết

2.3.1. Số đo *SE* và các tính chất

Để chỉnh sửa một số kết quả lý thuyết không đúng trong [91], đầu tiên, số đo $I(\mathcal{D}_\alpha)$ trong [91] được thay thế bởi một số đo $SI(\mathcal{D}_\alpha)$ khác (α là một chuỗi tùy ý) để kiểm tra sự tương đương của hai *PDB*. Sau đó, luận án đề xuất một số đo *SE* mới để biểu diễn các *PDB* đơn giản hơn và đưa ra các điều kiện chính xác và tổng quát hơn để tỉa sớm các chuỗi không là đóng và/hoặc không là sinh.

Định nghĩa 2.5 (*Số đo SE và SI liên quan đến PDB*).

a) Cho trước chuỗi α, với mỗi $suf = suf(\Psi, \alpha)$ trong $\mathcal{D}_\alpha$, ta ký hiệu $remSize(suf)$ là số sự kiện còn lại của suf trong $\mathcal{D}_\alpha$ (sau khi có $\delta = prefix(\Psi, \alpha)$) với định nghĩa như sau: $remSize(suf) = size(\Psi) - size(\delta)$.

b) Tổng số sự kiện và số thuộc tính còn lại trong $\mathcal{D}_\alpha$ được ký hiệu tương ứng là $SE(\mathcal{D}_\alpha)$ và $SI(\mathcal{D}_\alpha)$, và được định nghĩa như sau:

$SE(\mathcal{D}_\alpha) = \sum_{suf \in \mathcal{D}_\alpha}(remSize(suf) + 1)$ và $SI(\mathcal{D}_\alpha) = \sum_{suf \in \mathcal{D}_\alpha}(length(suf) + 1)$.

Ví dụ 2.3 (*Số đo SE và SI*). Chú ý rằng, số 1 được thêm trong định nghĩa của SE and SI để phân biệt hai *PDB* khác nhau $\mathcal{D}_\beta$ và $\mathcal{D}_\alpha = \mathcal{D}_\beta \cup \{<>, \ldots, <>\}$, mặc dù hai cơ sở dữ liệu chiếu có cùng giá trị I, nhưng số chuỗi của chúng khác nhau nên $supp(\alpha) \neq supp(\beta)$. *Ví dụ*, xét hai chuỗi α = *g→af* và β = *g→afg* và $\alpha \sqsubset \beta$. Ta có $\mathcal{D}_\alpha = \{<>\}$, $\mathcal{D}_\beta = \emptyset$ và $I(\mathcal{D}_\alpha) = I(\mathcal{D}_\beta) = 0$. Mặc dù $I(\mathcal{D}_\alpha) = I(\mathcal{D}_\beta)$ và tổng sự kiện của các chuỗi trong hai *PDB* này đều bằng 0, ta vẫn không thể phân biệt $\mathcal{D}_\alpha$ và $\mathcal{D}_\beta$. Tuy nhiên, với những số đo SI và SE ta có thể phân biệt hai *PDB* này. Thật vậy, vì $SI(\mathcal{D}_\alpha) = 1 \neq 0 = SI(\mathcal{D}_\beta)$ và $SE(\mathcal{D}_\alpha) = 1 \neq 0 = SE(\mathcal{D}_\beta)$, nên $\mathcal{D}_\alpha \neq \mathcal{D}_\beta$.

Mối quan hệ giữa $remSize$ và chiều dài của các hậu tố được chỉ ra trong *Bổ đề 2.2*. Ngoài ra, để chứng minh *Định lý 2.2*, tính đơn điệu giảm của toán tử ρ, độ hỗ trợ, *PDB* và số đo SE đã được trình bày trong *Mệnh đề 2.1*.

2.3.2. Các điều kiện tỉa sớm

Dựa trên những số đo SE, SI và *Mệnh đề 2.1*, *Định lý 2.2* dưới đây chỉ ra những điều kiện để hai *PDB* tương đương và để loại sớm các ứng viên không hứa hẹn.

Định lý 2.2 (*Sự tương đương của các PDB và sự loại bỏ sớm mở rộng (Extended Early Elimination – 3E)*).

Cho α và β là hai chuỗi tùy ý sao cho $\alpha \sqsubseteq \beta$.

a) (*Sự tương đương của các PDB dựa trên số đo SI và SE*)

$\mathcal{D}_\alpha = \mathcal{D}_\beta \Leftrightarrow SI(\mathcal{D}_\alpha) = SI(\mathcal{D}_\beta) \Leftrightarrow SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta)$ và $lastItemOf(\alpha) = lastItemOf(\beta)$.

b) (*Sự loại bỏ sớm mở rộng – 3E*). Nếu $SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta)$, thì:

(*i*). $\rho(\alpha) = \rho(\beta)$ và do đó $supp(\alpha) = supp(\beta)$.

(*ii*). Tất cả các chuỗi mở rộng *s-ext* γ (của α) và δ (của β) với cùng tập thuộc tính sẽ có cùng *PDB* và độ hỗ trợ, nghĩa là $\mathcal{D}_\gamma = \mathcal{D}_\delta$ và $supp(\gamma) = supp(\delta)$.

(*iii*). Ngoài ra, nếu $lastEventOf(\alpha) = lastEventOf(\beta)$, thì tất cả các chuỗi mở rộng *i-ext* γ' (của α) và δ' (của β) với cùng tập thuộc tính sẽ có cùng *PDB* và độ hỗ trợ, nghĩa là $\mathcal{D}_{\gamma'} = \mathcal{D}_{\delta'}$ và $supp(\gamma') = supp(\delta')$.

Với mỗi chuỗi α tương ứng với một nút trong cây tiền tố, ta mở rộng α với các thuộc tính phổ biến fi để có được các chuỗi mở rộng $\alpha \diamond fi$ (cả hai loại mở rộng $\alpha \diamond_i fi$ và $\alpha \diamond_s fi$) sao cho chuỗi $\alpha \diamond fi$ cũng phổ biến. Ta ký hiệu $iEB(\alpha)$ (hay $sEB(\alpha)$) là tập tất cả các nhánh mở rộng tập thuộc tính (hay mở rộng chuỗi) của α với tất cả các thuộc tính phổ biến fi trong cây tiền tố, nghĩa là, $iEB(\alpha)$ (hay $sEB(\alpha)$) gồm α, tất cả các nút $\alpha \diamond_i fi$ (hay $\alpha \diamond_s fi$) và tất cả các nút hậu duệ của chúng. Khi đó, tập gồm cả hai $iEB(\alpha)$ và $sEB(\alpha)$ được ký hiệu là $EB(\alpha)$.

Dựa trên *Định lý 2.2*, *Hệ quả 2.3* trong luận án đã chỉ ra phương pháp áp dụng các điều kiện tỉa sớm mở rộng *3E* để tỉa các chuỗi không là đóng hoặc không là sinh trong cây tiền tố. Tương tự, trong *Hệ quả 2.4* dưới đây luận án cũng chỉ ra cách sử dụng các điều kiện này để tỉa sớm các chuỗi không là đóng và cũng không là sinh.

Hệ quả 2.4 (*Tỉa sớm các chuỗi không là đóng và cũng không là sinh – EPGenClo*).

Cho trước một *SDB* $\mathcal{D}$ và ba chuỗi α, β, γ sao cho $\alpha \sqsubseteq \beta \sqsubseteq \gamma$, khi đó:

a) Nếu $$SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta) = SE(\mathcal{D}_\gamma), \quad (2.3)$$

thì các chuỗi δ, ε và λ có cùng *PDB* và độ hỗ trợ, trong đó δ, ε và λ là tất cả các chuỗi mở rộng *s-ext* tương ứng của α, β và γ với cùng một tập thuộc tính. Ngoài ra, tất cả các chuỗi mở rộng *s-ext* và *i-ext* của δ, ε và λ (với cùng tập thuộc tính) cũng có cùng *PDB* và độ hỗ trợ. Khi đó, tất cả các chuỗi trong tập $sEB(\beta)$ chắc chắn không là chuỗi đóng và sinh, do đó chúng có thể được tỉa. Tập tất cả các chuỗi không là đóng và sinh như vậy được ký hiệu là $nonCG\_sEB(\beta)$.

b) Nếu $SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta) = SE(\mathcal{D}_\gamma)$ và

$$lastEventOf(\alpha) = lastEventOf(\beta) = lastEventOf(\gamma), \quad (2.4)$$

thì các chuỗi δ, ε và λ có cùng *PDB* và độ hỗ trợ, trong đó δ, ε và λ là tất cả các chuỗi mở rộng (*s-ext* và *i-ext*) tương ứng của α, β và γ với cùng một tập thuộc tính. Khi đó, hai tập $sEB(\beta)$ và $iEB(\beta)$ không thể chứa bất kỳ chuỗi đóng và chuỗi sinh nào và bởi vậy chúng có thể được loại

bỏ khỏi cây tiền tố. Tập tất cả các chuỗi không là đóng và sinh trong hai tập này được ký hiệu là $nonCG\_EB(\beta)$.

Chú ý rằng, các điều kiện trong *Hệ quả 2.3* và *Hệ quả 2.4* cho phép tỉa sớm một chuỗi và tất cả các nhánh mở rộng *s-ext* và *i-ext* của nó trên cây tiền tố mà không chứa bất kỳ chuỗi đóng và/hoặc chuỗi sinh nào, và số lượng các chuỗi được tỉa có thể là cỡ mũ. Điều này đã được minh họa trong *Ví dụ 2.5* và *Ví dụ 2.6*.

2.4. Các thuật toán khai thác chuỗi đóng và chuỗi sinh phổ biến

2.4.1. Cấu trúc dữ liệu *IDList*

Các thuật toán trong chương này được cài đặt dựa trên định dạng *SDB* theo chiều dọc (*VDF*) được biểu diễn qua một cấu trúc gọi là *IDList* [7,97] gắn với mỗi chuỗi.

2.4.2. Các chiến lược tỉa địa phương

Để tránh việc kiểm tra chuỗi con trên nhiều chuỗi đã được sinh ra, luận án đưa ra các *Hệ quả 2.5 – 2.7* (áp dụng các kỹ thuật tỉa *EPClo*, *EPGen* và *EPGenClo* trong các *Hệ quả 2.3 – 2.4*) để tỉa các chuỗi không là đóng và/hoặc không là sinh tại hai hoặc ba mức liên tiếp trên cây tiền tố. Cách làm như vậy được gọi là *kỹ thuật* hay *chiến lược tỉa địa phương*. Chú ý rằng, *Hệ quả 2.5* (hoặc *Hệ quả 2.6*) sẽ được áp dụng vào thuật toán *FCloSM* (hoặc *FGenSM*) để khai thác các chuỗi đóng phổ biến (hoặc các chuỗi sinh phổ biến). Trong khi đó, *Hệ quả 2.7* dưới đây sẽ được áp dụng vào thuật toán *FGenCloSM* để khai thác đồng thời cả hai tập này.

Hệ quả 2.7 (*Tỉa địa phương các chuỗi không là đóng và không là sinh (LPGenClo) trong ba mức liên tiếp của cây tiền tố mà không cần kiểm tra quan hệ chuỗi con*).

Cho hai chuỗi bất kỳ có cùng cha (hay cùng tiền tố) trong cây tiền tố, $\varepsilon = \lambda \diamond b$ và $\beta = \lambda \diamond v$.

a) Với mỗi chuỗi mở rộng *i-ext* mới của ε với v, $\gamma = i\_new = \varepsilon \diamond_i v$, nếu
   (i). $SE(\mathcal{D}_\lambda) = SE(\mathcal{D}_\varepsilon) = SE(\mathcal{D}_\gamma)$ hoặc ($\pi = \delta \diamond b$, $parent(\lambda) = \delta$ và $SE(\mathcal{D}_\pi) = SE(\mathcal{D}_\varepsilon) = SE(\mathcal{D}_\gamma)$), thì các nhánh $nonCG\_sEB(\varepsilon)$ có thể được tỉa.
   (ii). $SE(\mathcal{D}_\lambda) = SE(\mathcal{D}_\beta) = SE(\mathcal{D}_\gamma)$ hoặc ($\alpha = \delta \diamond v$ và $SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta) = SE(\mathcal{D}_\gamma)$), thì các nhánh $nonCG\_sEB(\beta)$ có thể được tỉa.

b) Với mỗi chuỗi mở rộng *s-ext* mới của ε với thuộc tính v, $\gamma = s\_new = \varepsilon \diamond_s v$ sao cho $\beta = \lambda \diamond_s v$, $\alpha = \delta \diamond_s v$, $parent(\lambda) = \delta$ và $SE(\mathcal{D}_\alpha) = SE(\mathcal{D}_\beta) = SE(\mathcal{D}_\gamma)$, khi đó các chuỗi trong tập $nonCG\_EB(\beta)$ chắc chắn không là chuỗi đóng và cũng không là chuỗi sinh phổ biến, do đó chúng có thể được tỉa.

   Các *Hệ quả 2.5 – 2.7* được minh họa chi tiết thông qua *Ví dụ 2.7*.

2.4.3. Các thuật toán

Từ *Hệ quả 2.7*, luận án thiết kế thủ tục *HybridSearch* (*Hình 2.5*) để khám phá không gian tìm kiếm ứng với chuỗi β, trong đó các điều kiện trong hệ quả này được áp dụng để tỉa sớm nhiều chuỗi không là đóng và không là sinh. Sau đó, dựa trên thủ tục này, luận án thiết kế thuật toán *FGenCloSM* trong *Hình 2.4* để khai thác đồng thời hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$. Dựa trên các *Hệ quả 2.5 – 2.6*, luận án cũng thiết kế thuật toán *FCloSM* trong *Hình 2.9* để khai thác $\mathcal{FCS}$ và thuật toán *FGenSM* trong *Hình 2.10* để khai thác $\mathcal{FGS}$. Ngoài ra, luận án cũng đã phát triển thuật toán song song *Par-GenCloSM* để khai thác hiệu quả hơn nữa cả hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ dựa trên *Hệ quả 2.7*, *FGenCloSM* và kiến trúc bộ xử lý đa nhân (*Hình 2.12*). Tính đúng của các

thuật toán *FGenCloSM*, *Par-GenCloSM*, *FCloSM* và *FGenSM* được đảm bảo bởi *Định lý 2.2* và các *Hệ quả 2.3 – 2.7* (đã được chứng minh một cách chặt chẽ).

($\mathcal{FCS}$, $\mathcal{FGS}$) **FGenCloSM** ($\mathcal{D}$, $minsupp$)
1. Quét $\mathcal{D}$ để tạo *V($\mathcal{D}$)* và xác định danh sách các thuộc tính phổ biến $\mathcal{F}^1$.
2. $\mathcal{FCS}$:= $\mathcal{FGS}$:= $\mathcal{F}^1$;
3. **for each** chuỗi $\beta \in \mathcal{F}^1$ **do**
4. HybridSearch (β , $minsupp$, ($\mathcal{FCS}$, $\mathcal{FGS}$));
5. **return** ($\mathcal{FCS}$, $\mathcal{FGS}$);

Hình 2.4. Thuật toán *FGenCloSM* khai thác đồng thời $\mathcal{FCS}$ và $\mathcal{GS}$.

2.5. Đánh giá thực nghiệm

Các thực nghiệm được thực hiện trên nhiều *SDB* thực và tổng hợp với đặc trưng được mô tả trong *Bảng 2.3*. Hiệu quả của các thuật toán được đề xuất trong chương này được so sánh với các thuật toán tiêu biểu trước đây, chẳng hạn như *CloSpan* [91], *BIDE* [88], *ClaSP* [27] và *CM-ClaSP* [17] dùng để khai thác các chuỗi đóng phổ biến, và *VGEN* [19], *FEAT* [25] và *FSGP* [92] dùng để khai thác các chuỗi sinh phổ biến. Các kết quả thực nghiệm đã cho thấy rằng các thuật toán được đề xuất hiệu quả hơn nhiều (về thời gian chạy và bộ nhớ sử dụng) so với các thuật toán đã biết trước đây. Trên các *SDB* lớn và/hoặc các giá trị $minsupp$ bé, các thuật toán được đề xuất có thể chạy nhanh hơn lên đến hàng trăm lần so với các thuật toán trước đây.

Chú ý rằng các kết quả lý thuyết và thuật toán được đề xuất trong chương này có vai trò quan trọng trong các phương pháp tiếp cận được đưa ra trong các *Chương 3-4*.

Chương 3 - Cấu trúc của tập các chuỗi phổ biến dựa trên chuỗi đóng và chuỗi sinh

Như được trình bày trong *Chương 2*, hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ là những biểu diễn không mất thông tin và súc tích của tập các chuỗi phổ biến $\mathcal{FS}$. Sau khi có được cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) bằng cách sử dụng các thuật toán hiệu quả được đề xuất trong *Chương 2*, về mặt lý thuyết, một câu hỏi tự nhiên được đặt ra là, các chuỗi trong $\mathcal{FS}$ có mối quan hệ mật thiết như thế nào với các chuỗi đóng và chuỗi sinh tương ứng trong $\mathcal{FCS}$ và $\mathcal{FGS}$? Nói cách khác, trong chương này, luận án muốn chỉ rõ cấu trúc hay biểu diễn tường minh của $\mathcal{FS}$ dựa vào cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) (bài toán $\mathcal{BT}_2$). Như một hệ quả, luận án muốn sinh ra nhanh $\mathcal{FS}$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$ (thay vì từ *SDB*) sử dụng cấu trúc đã chỉ ra. Cấu trúc này sau đó sẽ được áp dụng để phát triển các phương pháp hiệu quả giải quyết bài toán $\mathcal{BT}_3$ trong *Chương 4*. Trước tiên, luận án trình bày các khái niệm cơ sở dưới đây.

3.1. Các khái niệm cơ sở

Định nghĩa 3.1 (*Quan hệ tương đương* $\sim$). Cho trước hai chuỗi bất kỳ $\alpha = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_p$ và $\beta = E'_1 \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_q$, xét quan hệ tương đương $\sim$ trên các chuỗi như sau:

$$\alpha \sim \beta \Leftrightarrow \rho(\alpha) = \rho(\beta),$$

trong đó toán tử ρ được trình bày trong *Định nghĩa 2.2b*. Lớp tương đương (hay gọi tắt là lớp) chứa α được ký hiệu là $[\alpha]$ và được định nghĩa như sau: $[\alpha] \stackrel{\text{def}}{=} \{\beta \mid \rho(\beta) = \rho(\alpha)\}$. Quan hệ tương

đương này sẽ được sử dụng để phân hoạch tập các chuỗi phổ biến $\mathcal{FS}$ thành các lớp tương đương như được trình bày trong *Mục 3.2*.

Định nghĩa 3.2 (*Những khái niệm và định nghĩa liên quan đến chuỗi đóng và sinh*). Cho ba chuỗi α, γ và σ.

a) Gọi $\mathcal{CS}$ và $\mathcal{GS}$ tương ứng là tập tất cả các chuỗi đóng và chuỗi sinh, ta ký hiệu: $\mathcal{FCS} \stackrel{\text{def}}{=} \mathcal{CS} \cap \mathcal{FS}$ và $\mathcal{FGS} \stackrel{\text{def}}{=} \mathcal{GS} \cap \mathcal{FS}$. Rõ ràng, khi $minsupp$ = 1, ta có $\mathcal{FCS} = \mathcal{CS}$ và $\mathcal{FGS} = \mathcal{GS}$.

b) σ được gọi là chuỗi đóng của α nếu σ là một chuỗi đóng chứa α và chúng có cùng độ hỗ trợ, nghĩa là: $\sigma \in \mathcal{CS}$, $\sigma \sqsupseteq \alpha$ và $supp(\alpha) = supp(\sigma)$.

c) γ được gọi là chuỗi sinh của α nếu γ là một chuỗi sinh chứa trong α và chúng có cùng độ hỗ trợ, nghĩa là: $\gamma \in \mathcal{GS}$, $\alpha \sqsupseteq \gamma$ và $supp(\alpha) = supp(\gamma)$.

d) Vì một chuỗi α có thể có nhiều chuỗi đóng (chứa α) và nhiều chuỗi sinh (nằm trong α), nên luận án định nghĩa $CloSet(\alpha) \stackrel{\text{def}}{=} \{\sigma \in \mathcal{FCS} \mid \sigma \sqsupseteq \alpha$ và $supp(\sigma) = supp(\alpha)\}$ là tập các chuỗi đóng của α và $GenSet(\alpha) \stackrel{\text{def}}{=} \{\gamma \in \mathcal{FGS} \mid \gamma \sqsubseteq \alpha$ và $supp(\gamma) = supp(\alpha)\}$ là tập các chuỗi sinh của α.

Hiển nhiên, $CloSet(\alpha) = \{\sigma \in \mathcal{FCS} \mid \sigma \sqsupseteq \alpha$ và $\rho(\sigma) = \rho(\alpha)\}$ và $GenSet(\alpha) = \{\gamma \in \mathcal{FGS} \mid \gamma \sqsubseteq \alpha$ và $\rho(\gamma) = \rho(\alpha)\}$, vì $\forall \beta \sqsupseteq \alpha, \rho(\beta) \subseteq \rho(\alpha)$ và $(supp(\beta) = supp(\alpha) \Leftrightarrow \rho(\beta) = \rho(\alpha))$. Từ *Định nghĩa 3.2*, ta có *Mệnh đề 3.1* dưới đây.

Mệnh đề 3.1 (*Tính chất của những chuỗi đóng và chuỗi sinh của một chuỗi*).
Cho các chuỗi α, γ, σ, λ, δ, ta có:

a) $CloSet(\alpha) \neq \emptyset$ và $GenSet(\alpha) \neq \emptyset$.

b) $\sigma \in CloSet(\alpha) \Leftrightarrow \sigma$ là chuỗi đóng, $\sigma \sqsupseteq \alpha$ và $\rho(\sigma) = \rho(\alpha)$. $\sigma \in CloSet(\alpha)$, $\forall \lambda$: $\alpha \sqsubseteq \lambda \sqsubseteq \sigma$, thì $\rho(\alpha) = \rho(\lambda) = \rho(\sigma)$, nên $supp(\alpha) = supp(\lambda) = supp(\sigma)$.

c) $\gamma \in GenSet(\alpha) \Leftrightarrow \gamma$ là chuỗi sinh, $\gamma \sqsubseteq \alpha$ và $\rho(\gamma) = \rho(\alpha)$. $\gamma \in GenSet(\alpha)$, $\forall \delta$: $\gamma \sqsubseteq \delta \sqsubseteq \alpha$, thì $\rho(\gamma) = \rho(\delta) = \rho(\alpha)$, nên $supp(\gamma) = supp(\delta) = supp(\alpha)$.

d) $\forall \gamma \in GenSet(\alpha)$, $\forall \sigma \in CloSet(\alpha)$, thì $\gamma \sqsubseteq \alpha \sqsubseteq \sigma, \rho(\gamma) = \rho(\alpha) = \rho(\sigma)$, do đó $supp(\gamma) = supp(\alpha) = supp(\sigma)$.

e) σ là chuỗi đóng $\Leftrightarrow CloSet(\sigma) = \{\sigma\}$, vì vậy, nếu $\sigma \notin CloSet(\sigma)$ hoặc $|CloSet(\sigma)| > 1$, thì σ không là chuỗi đóng. γ là một chuỗi sinh $\Leftrightarrow GenSet(\gamma) = \{\gamma\}$, do đó nếu $\gamma \notin GenSet(\gamma)$ hoặc $|GenSet(\gamma)| > 1$, thì γ không là chuỗi sinh.

Chú ý rằng, với bài toán khai thác các tập (itemset) phổ biến trên cơ sở dữ liệu giao dịch (transaction database), ứng với mỗi tập thuộc tính A, chỉ tồn tại duy nhất một tập đóng chứa A và có cùng độ hỗ trợ với A (còn gọi là bao đóng của A) (xem [5]). Vì vậy, việc chỉ ra cấu trúc của $\mathcal{FS}$ dựa vào $\mathcal{FCS}$ và $\mathcal{FGS}$ gặp nhiều thách thức hơn hẳn. Thách thức này đến từ thực tế rằng mỗi chuỗi có thể có nhiều chuỗi đóng chứa nó, và nó có thể xuất hiện nhiều lần ở những vị trí khác nhau trong một chuỗi đóng cũng như một chuỗi dữ liệu đầu vào.

3.2. Các kết quả lý thuyết

Với bất kỳ tập con A của một tập Ω cho trước, dãy n các tập con của A (A_1, A_2, …, A_n) được gọi là một phân hoạch của A (ký hiệu là $A = \sum_{i=1,\dots n} A_i$, hay A là hợp rời của chúng) nếu chúng là rời nhau, nghĩa là $A_i \cap A_j = \emptyset$, $\forall i, j$ =1, 2, …, n: $i \neq j$. Vì vậy, từ đây trở đi, khi nói phân hoạch A thành các lớp tương đương A_1, A_2, …, A_n có nghĩa là chúng rời nhau.

3.2.1. Phân hoạch của tập $\mathcal{FS}$

Đặt $RO \stackrel{\text{def}}{=} \{\rho(\alpha) \mid \alpha \in \mathcal{FS}\}$ và $\forall ro \in RO, CS(ro) \stackrel{\text{def}}{=} \{\sigma \in \mathcal{FCS} \mid \rho(\sigma) = ro\}$, khi đó $RO = \{\rho(\sigma) \mid \sigma \in \mathcal{FCS}\}$. Do $\sim$ là quan hệ tương đương nên tập $\mathcal{FS}$ có thể được phân hoạch thành các lớp tương đương (rời nhau) $FS(ro)$ như sau:

$$\mathcal{FS} = \sum_{ro \in RO} FS(ro), \tag{3.1}$$

trong đó, $FS(ro) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS} \mid \rho(\alpha) = ro\}$. Dễ dàng thấy rằng $FS(ro) = \bigcup_{\sigma \in CS(ro)} \bigcup_{\gamma \in GenSet(\sigma)} \mathcal{FS}(\sigma, \gamma)$, hay $FS(ro)$ là hợp (không nhất thiết rời nhau) của những tập con $\mathcal{FS}(\sigma, \gamma) \stackrel{\text{def}}{=} \{\alpha \mid \gamma \sqsubseteq \alpha \sqsubseteq \sigma\}$, với $\sigma \in CS(ro)$ và $\gamma \in GenSet(\sigma)$. Vì vậy, để sinh nhanh và không trùng lặp các chuỗi phổ biến trong $\mathcal{FS}$, ta cần chỉ ra biểu diễn tường minh hay phương pháp sinh không trùng lặp các chuỗi trong các tập $\mathcal{FS}(\sigma, \gamma)$ khác nhau.

3.2.2. Dẫn xuất tất cả các chuỗi phổ biến trong tập $\mathcal{FS}(\sigma, \gamma)$

Từ công thức (3.1), do các lớp $FS(ro)$ là rời nhau, để dẫn xuất không trùng lặp tất cả các chuỗi phổ biến trong tập $\mathcal{FS}$, ta cần chỉ ra cấu trúc của các chuỗi phổ biến α trong mỗi tập con $\mathcal{FS}(\sigma, \gamma)$. Cấu trúc ở đây có nghĩa là mối quan hệ tường minh giữa α và các chuỗi đại diện σ và γ. Nói cách khác, ta cần chỉ ra cách biểu diễn hay phương pháp sinh ra α từ σ và γ.

Cho hai chuỗi bất kỳ $\gamma = E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_p$ và $\sigma = E'_1 \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_q$ sao cho $\gamma \sqsubseteq \sigma$. Gọi $lp \stackrel{\text{def}}{=} [j_1, j_2, \ldots, j_p]$ là danh sách các vị trí mà tại đó γ xuất hiện trong σ, trong đó $1 \leqslant j_1 < j_2 < \ldots < j_p \leqslant q$ và $E_k \subseteq E'_{j_k}$, $\forall k = 1..p$. *Ví dụ*, với *SDB* trong *Bảng 2.2*, xét tập đóng $\sigma = ace \rightarrow ad \rightarrow ac \rightarrow ac \in CS(\{\Psi_1, \Psi_2\})$ và tập sinh γ của σ, $\gamma = a \rightarrow c \in GenSet(\sigma)$. Ta thấy rằng, hai tập thuộc tính $\{a\}$ và $\{c\}$ trong γ có thể là những tập con tương ứng của *ace* và *ac* trong σ tại những vị trí 1 và 3. Khi đó, ta nói rằng γ xuất hiện trong σ theo $lp_1 = [1, 3]$. Ngoài ra, γ cũng xuất hiện trong σ tại những danh sách vị trí khác, đó là $lp_2 = [1, 4]$, $lp_3 = [2, 3]$, $lp_4 = [2, 4]$ và $lp_5 = [3, 4]$. Phương pháp sinh tường minh tất cả các chuỗi phổ biến trong mỗi tập $\mathcal{FS}(\sigma, \gamma)$ gồm những bước chính sau:

(1) Xác định tập $LP(\sigma, \gamma) \stackrel{\text{def}}{=} \{lp_1, lp_2, \ldots, lp_n\}$ gồm những danh sách vị trí mà γ xuất hiện trong σ. *Ví dụ*, với $\sigma = ace \rightarrow ad \rightarrow ac \rightarrow ac$ và $\gamma = a \rightarrow c \in GenSet(\sigma)$, ta có $LP(\sigma, \gamma) = \{lp_i, i = 1, \ldots, 5\}$.

(2) Mở rộng γ theo kích thước q của σ theo mỗi $lp \in LP(\sigma, \gamma)$ để đạt được chuỗi mở rộng (*extension*) của γ trong σ theo lp: $Ex(\gamma, lp) \stackrel{\text{def}}{=} E_1^{\sim} \rightarrow E_2^{\sim} \rightarrow \ldots \rightarrow E_q^{\sim}$ với

$$E_i^{\sim} = \begin{cases} \emptyset, & \text{nếu } i \notin lp \\ E_k, & \text{nếu } i = j_k \in lp \end{cases}, \forall i = 1, \ldots, q.$$

Ví dụ, xét $lp_2 = [1, 4] \in LP(\sigma, \gamma)$, khi đó $Ex(\gamma, lp_2) = a \rightarrow \emptyset \rightarrow \emptyset \rightarrow c$.

(3) Với mỗi $lp = \{j_1, j_2, \ldots, j_p\} \in LP(\sigma, \gamma)$, tìm chuỗi khác biệt $\Delta(lp)$ của σ và $Ex(\gamma, lp)$, trong đó $\Delta(lp) \stackrel{\text{def}}{=} \sigma \ominus Ex(\gamma, lp) = D_1 \rightarrow D_2 \rightarrow \ldots \rightarrow D_q$ và $D_i \stackrel{\text{def}}{=} E'_i \backslash E_i^{\sim}$, $\forall i = 1, \ldots, q$.
Ví dụ, với $lp_2 = [1, 4]$, ta có $\Delta(lp_2) = ce \rightarrow ad \rightarrow ac \rightarrow a$, vì $D_1 = ace \setminus a = ce$, $D_2 = ad \setminus \emptyset = ad$, $D_3 = ac \setminus \emptyset = ac$ và $D_4 = ac \setminus c = a$.

(4) Với mỗi $lp \in LP(\sigma, \gamma)$, tất cả các chuỗi $\alpha = F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_q$ trong $\mathcal{FS}(\sigma, \gamma)$ được sinh ra dưới dạng $\alpha = Ex(\gamma, lp) \oplus \delta(lp)$ với mọi $\delta(lp) = d_1 \rightarrow d_2 \rightarrow \ldots \rightarrow d_q \sqsubseteq \Delta(lp)$, trong đó phép toán $\oplus$ là tổng trực tiếp của hai chuỗi trên những tập thuộc tính xuất hiện tại những ví trị giống nhau trong $Ex(\gamma, lp)$ và $\delta(lp)$, nghĩa là, $F_i = E_i^{\sim} + d_i$ và $d_i \subseteq D_i$, $\forall i = 1, \ldots, q$. Sau đó, ta rút gọn α, $reduce(\alpha)$, bằng cách bỏ đi các tập thuộc tính rỗng. Khi đó, ta có thể xem α và $reduce(\alpha)$ là như nhau.

Ví dụ, với $\Delta(lp_2)$ = *ce→ad→ac→a*, xét một chuỗi con $\delta(lp_2)$ = *e→a→∅→∅* của $\Delta(lp_2)$, khi đó $\alpha = Ex(\gamma, lp_2) \oplus \delta(lp_2)$ = *a→∅→∅→c* $\oplus$ *e→a→∅→∅* = *ae→a→∅→c*. Bằng cách bỏ đi những tập thuộc tính rỗng, ta có được chuỗi rút gọn $\alpha = reduce(\alpha)$ = *ae→a→c*. Chú ý rằng, số chuỗi con $\delta(lp_2)$ của $\Delta(lp_2)$ là $2^{length(\Delta(lp_2))} = 2^7 = 128$. Vì vậy, ta có 128 chuỗi con α được sinh ra theo lp_2 = [1, 4].

(5) Cuối cùng, ta có được tập $\mathcal{FS}'(\sigma,\gamma) \stackrel{\text{def}}{=} \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \mid lp \in LP(\sigma,\gamma), \delta(lp) \sqsubseteq \Delta(lp)\}$ chứa các chuỗi được sinh ra một cách tường minh từ cặp (σ, γ).

Dựa trên phương pháp sinh tường minh các chuỗi phổ biến này, ta có mệnh đề sau.

Mệnh đề 3.2 ($\mathcal{FS}'(\sigma,\gamma)$ *– biểu diễn tường minh của* $\mathcal{FS}(\sigma,\gamma)$). $\forall ro \in RO$, $\forall \sigma \in CS(ro)$, $\forall \gamma \in GenSet(\sigma)$, ta có:

$$\mathcal{FS}(\sigma,\gamma) = \mathcal{FS}'(\sigma,\gamma). \quad (3.2)$$

Trước tiên, chú ý rằng, mặc dù phân hoạch trong (3.1) được đưa ra để tránh sự trùng lặp các chuỗi giữa các lớp $FS(ro)$ ($\forall ro \in RO$) khác nhau, phương pháp sinh các chuỗi phổ biến này vẫn có thể dẫn đến nhiều chuỗi trùng lặp giữa các tập con $\mathcal{FS}'(\sigma_i, \gamma_j)$ khác nhau của $FS(ro)$, trong đó $\sigma_i \in CS(ro)$ và $\gamma_j \in GenSet(\sigma_i)$, như được chỉ ra trong *Ví dụ 3.3*. Bởi vậy, trong mục tiếp theo, luận án đề xuất ba chiến lược tỉa để khắc phục vấn đề này.

3.2.3. Dẫn xuất không trùng lặp tất cả các chuỗi phổ biến trong tập $\mathcal{FS}$

Với mỗi $ro \in RO$, $CS(ro) = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ và $GenSet(\sigma_i) = \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$, xét bất kỳ hai chuỗi đóng σ_k, σ_i trong $CS(ro)$, $k < i$, và hai chuỗi sinh γ_l, $\gamma_j \in GenSet(\sigma_i)$, $l < j$.

Chiến lược tỉa 1 (*Loại bỏ các chuỗi con trùng lặp được sinh ra dựa trên hai chuỗi đóng khác nhau* σ_k *và* σ_i *trong* $CS(ro)$). Bất kỳ chuỗi α nào được sinh ra trong tập $\mathcal{FS}'(\sigma_i, \gamma_j)$ thỏa điều kiện trùng lặp $DCondC(\alpha, \sigma_i, CS(ro)) \stackrel{\text{def}}{=} (\exists \sigma_k \in CS(ro): k < i, \alpha \sqsubseteq \sigma_k)$ (α chứa trong một chuỗi đóng σ_k trước đó, $k < i$) đều bị loại bỏ.

Chiến lược tỉa 2 (*Loại bỏ các chuỗi con trùng lặp được sinh ra dựa trên hai chuỗi sinh khác nhau* γ_l *và* γ_j *trong* $GenSet(\sigma_i)$). Bất kỳ chuỗi α nào được sinh ra trong tập con $\mathcal{FS}'(\sigma_i, \gamma_j)$, $\gamma_j \in GenSet(\sigma_i)$, thỏa điều kiện trùng lặp $DCondG(\alpha, \sigma_i, \gamma_j) \stackrel{\text{def}}{=} (\exists \gamma_l \in GenSet(\sigma_i): l < j, \alpha \sqsupseteq \gamma_l)$ (α chứa một chuỗi sinh γ_l trước đó của σ_i, $l < j$) đều bị loại bỏ.

Ta định nghĩa $\mathcal{FS}^*(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}'(\sigma_i, \gamma_j) \mid not(DCondC(\alpha, \sigma_i, CS(\rho(\sigma_i)))) \text{ and } not(DCondG(\alpha, \sigma_i, \gamma_j))\}$, khi đó không có sự trùng lặp chuỗi giữa những tập con khác nhau $\mathcal{FS}^*(\sigma_i, \gamma_j)$ trong mỗi lớp $FS(ro)$, với $\sigma_i \in CS(ro)$, $\gamma_j \in GenSet(\sigma_i)$. Dựa trên (3.1), (3.2) và các *Chiến lược tỉa 1-2*, ta có phân hoạch mịn hơn sau đây.

Định lý 3.1 (*Các phân hoạch của* $\mathcal{FS}$).

a. $\mathcal{FS} = \sum_{ro \in RO} FS(ro)$ (*phân hoạch thô*).

b. $FS(ro) = \sum_{\sigma_i \in CS(ro)} \sum_{\gamma_j \in GenSet(\sigma_i)} \mathcal{FS}^*(\sigma_i, \gamma_j)$, $\forall ro \in RO$.

Do đó, $\mathcal{FS} = \sum_{ro \in RO} \sum_{\sigma_i \in CS(ro)} \sum_{\gamma_j \in GenSet(\sigma_i)} \mathcal{FS}^*(\sigma_i, \gamma_j)$ (*phân hoạch mịn*).

Để việc trình bày những nội dung còn lại của chương này được ngắn gọn, với *SDB* cho trong *Bảng 2.2*, ta luôn xét $ro_1 = \{\Psi_1, \Psi_2\} \in RO$, $\sigma = \sigma_2$ = *ace→ad→ac→ac* $\in CS(ro_1)$, $\gamma = \gamma_2$ = *a→c* $\in GenSet(\sigma_2)$, nên $LP(\sigma,\gamma) = \{lp_1 = [1, 3], lp_2 = [1, 4], lp_3 = [2, 3], lp_4 = [2, 4], lp_5 = [3, 4]\}$. Khi đó, $Ex(\gamma, lp_1)$ = *a→∅→c→∅*, $\Delta(lp_1)$ = *ce→ad→a→ac*, $Ex(\gamma, lp_2)$ = *a→∅→∅→c* và $\Delta(lp_2)$ = *ce→ad→ac→a*.

Lưu ý rằng, các chiến lược tỉa 1-2 chỉ nhằm đảm bảo sự không trùng lặp giữa các tập con $\mathcal{FS}^*(\sigma_i, \gamma_j)$ khác nhau. Tuy nhiên, thách thức lớn nhất làm cho bài toán khai thác chuỗi phổ biến từ các chuỗi đóng-sinh khó khăn hơn nhiều so với bài toán khai thác tập (itemset) phổ biến từ các tập đóng-sinh là: mỗi chuỗi phổ biến có thể xuất hiện trong mỗi chuỗi đóng ở nhiều vị trí khác nhau. *Ví dụ*, chuỗi sinh $\gamma = a \rightarrow c$ xuất hiện trong chuỗi đóng $\sigma = ace \rightarrow ad \rightarrow ac \rightarrow ac$ tương ứng với năm tập vị trí lp_{1-5}. Khi đó, một số lượng rất lớn các chuỗi trùng lặp trong mỗi tập con $\mathcal{FS}'(\sigma, \gamma)$ có thể được sinh ra. Vì vậy, Chiến lược tỉa 3 dưới đây được đề xuất để phát hiện và tỉa sớm tất cả các chuỗi trùng lặp trong mỗi tập $\mathcal{FS}'(\sigma, \gamma)$ mà không cần sinh ra chúng.

Giả sử rằng X là một tập (hay một chuỗi) bất kỳ với nhiều phần tử (hay nhiều tập thuộc tính), ta ký hiệu $X[k]$ là phần tử thứ k của X, ví dụ, $lp_1[2] = 3$ hay $\Delta(lp_1)[3] = a$. Tập lp_m được gọi là đứng trước tập lp_n, ký hiệu là $lp_m \prec_{pre} lp_n$, nếu ($lp_m[1] < lp_n[1]$) hay ($lp_m[1] = lp_n[1]$ và $lp_m[2] < lp_n[2]$), hay ($lp_m[1] = lp_n[1]$ và $lp_m[2] = lp_n[2]$ và $lp_m[3] < lp_n[3]$), v.v. Chú ý rằng, những phần tử trong $LP(\sigma, \gamma)$ được sắp xếp theo thứ tự tăng dần đối với quan hệ thứ tự $\prec_{pre}$ này. Cho trước $ro \in RO$, $\sigma \in CS(ro)$, $\gamma \in GenSet(\sigma)$, $q = size(\sigma)$, lp_m và $lp_n \in LP(\sigma, \gamma)$ sao cho $lp_m \prec_{pre} lp_n$, gọi $k_0 = FI(m, n)$ là chỉ số đầu tiên tại đó lp_m khác lp_n, ví dụ, $k_0 = FI(1, 2) = 2$ vì $lp_1[1] = lp_2[1] = 1$ và $lp_1[2] = 3 \neq lp_2[2] = 4$. Xét bất kỳ chuỗi $\alpha = Ex(\gamma, lp) \oplus \delta(lp)$ trong $\mathcal{FS}'(\sigma, \gamma)$.

Chiến lược tỉa 3 (*Loại bỏ các chuỗi trùng lặp trong mỗi tập con* $\mathcal{FS}'(\sigma, \gamma)$). Với mỗi $lp \in LP(\sigma, \gamma)$ và $\delta(lp) \sqsubseteq \Delta(lp)$, bất kỳ chuỗi $\alpha = Ex(\gamma, lp) \oplus \delta(lp)$ trong $\mathcal{FS}'(\sigma, \gamma)$ mà thỏa một trong ba điều kiện trùng lặp dưới đây, $DCond_{1-3}(\alpha, lp) \stackrel{\text{def}}{=} \{DCond_i(\alpha, lp), i = 1..3\}$, thì α và những chuỗi cha tương ứng của nó sẽ bị loại bỏ. Chi tiết về ba điều kiện trùng lặp này được trình bày như sau.

(*i*). **Điều kiện trùng lặp 1** (*$DCond_1$ – dùng cho mỗi danh sách vị trí lp trong $LP(\sigma, \gamma)$*). Với mỗi $\delta(lp) = d_1 \rightarrow d_2 \rightarrow \ldots \rightarrow d_q \sqsubseteq \Delta(lp) = D_1 \rightarrow D_2 \rightarrow \ldots \rightarrow D_q$ và $\alpha = Ex(\gamma, lp) \oplus \delta(lp) = F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_q$, xét bất kỳ $i \notin lp$ sao cho $1 < i \leqslant q$. Nếu tồn tại một *dãy liên tiếp lớn nhất các sự kiện rỗng* trong α từ vị trí $(i-1)$ giảm xuống vị trí $n_1 \geqslant 1$ (nghĩa là, $F_k = \emptyset$, $\forall k = n_1, \ldots, i-1$), và tồn tại một chỉ số k như vậy sao cho tập thuộc tính d_i ($\neq \emptyset$) chứa trong tập $D_i \cap D_k$, thì α chắc chắn trùng lặp với một chuỗi được sinh trước đó theo chỉ số k. Ngoài ra, việc sinh tất cả các chuỗi phổ biến khác từ α bằng cách mở rộng các tập thuộc tính d_j trong $\delta(lp)$ sao cho $j \notin \{n_1, \ldots, i-1\}$ và $d_i \subseteq D_i \cap D_k$ cũng sẽ dẫn đến sự trùng lặp. Bởi vậy, ta có thể tỉa sớm những chuỗi này mà không cần sinh ra chúng.

(*ii*). **Điều kiện trùng lặp 2** (*$DCond_2$ – dùng cho hai danh sách vị trí khác nhau lp_n và lp_m trong $LP(\sigma, \gamma)$*). Với bất kỳ $lp_n \in LP(\sigma, \gamma)$, $\delta(lp_n) = d_1 \rightarrow d_2 \rightarrow \ldots \rightarrow d_q \sqsubseteq \Delta(lp_n)$ và $\alpha = Ex(\gamma, lp_n) \oplus \delta(lp_n) = F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_q$, nếu tồn tại một danh sách vị trí $lp_m \in LP(\sigma, \gamma)$ và một chỉ số $i = lp_m[k_0]$ sao cho $lp_m \prec_{pre} lp_n$ và d_i chứa $\gamma[k_0]$, thì α trùng với một chuỗi α' được sinh ra trước đó từ lp_m. Ngoài ra, việc sinh tất cả những chuỗi khác mà được tạo nên từ α bằng cách mở rộng những tập thuộc tính trong $\delta(lp_n)$ cũng sẽ dẫn đến sự trùng lặp. Do đó, ta có thể kết thúc việc sinh ra chúng.

(*iii*). **Điều kiện trùng lặp 3** (*$DCond_3$ – Dùng cho hai danh sách vị trí khác nhau, lp_n và $lp_m \in LP(\sigma, \gamma)$*). Với bất kỳ $lp_n \in LP(\sigma, \gamma)$, $\delta(lp_n) = d_1 \rightarrow d_2 \rightarrow \ldots \rightarrow d_q \sqsubseteq \Delta(lp_n)$ và $\alpha = Ex(\gamma, lp_n) \oplus \delta(lp_n) = F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_q$, nếu tồn tại một danh sách vị trí $lp_m \in LP(\sigma, \gamma)$ sao cho $lp_m \prec_{pre} lp_n$ và một chỉ số $i = lp_n[k_0]$ sao cho tất cả các sự kiện của α từ vị trí $j = lp_m[k_0]$ đến $i-1$ bằng $\emptyset$ (nghĩa là, $F_k = \emptyset$, $\forall k = j, \ldots, i-1$) và sự kiện d_i của $\delta(lp_n)$ chứa trong $\Delta(lp_n)[i]$

$\cap\, \Delta(lp_m)[j]$, thì α trùng với một chuỗi α' được sinh ra từ lp_m. Ngoài ra, sự trùng lặp cũng xảy ra khi sinh tất cả các chuỗi khác mà được tạo nên từ α bằng cách mở rộng những tập thuộc tính d_k trong $\delta(lp_n)$ sao cho $k \notin \{j, \ldots, i-1\}$ và $d_i \subseteq \Delta(lp_n)[i] \cap \Delta(lp_m)[j]$. Vì vậy, ta có thể tỉa sớm những chuỗi này mà không cần sinh ra chúng.

Chú ý rằng, số chuỗi trùng lặp được tỉa bởi Chiến lược tỉa 3 dựa trên ba điều kiện trùng lặp, $DCond_{1-3}$, là cỡ mũ (xem *Ví dụ 3.5 – 3.7*). Bởi vậy, để tránh sự trùng lặp khi sinh các chuỗi trong lớp $\mathcal{FS}'(\sigma,\gamma)$, ta định nghĩa một *biểu diễn không trùng lặp* của $\mathcal{FS}'(\sigma,\gamma)$ dựa trên *Chiến lược tỉa 3* như sau:

$$\mathcal{FS}''(\sigma,\gamma) \stackrel{\text{def}}{=} \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'(\sigma,\gamma) \mid not(DCond_{1-3}(\alpha, lp))\}.$$

Với biểu diễn $\mathcal{FS}''(\sigma,\gamma)$ của $\mathcal{FS}'(\sigma,\gamma)$, các chuỗi trong tập $\mathcal{FS}''(\sigma,\gamma)$ sẽ được sinh ra không trùng lặp (xem *Định lý 3.2*). Tuy nhiên, một câu hỏi khó được phát sinh một cách tự nhiên là, ngoại trừ ba tình huống dẫn đến trùng lặp như được trình bày trong chiến lược tỉa 3, còn có bất kỳ tình huống trùng lặp nào khác xảy ra trong quá trình sinh các chuỗi trong lớp $\mathcal{FS}''(\sigma,\gamma)$ hay không? Dựa trên *Định lý 3.1* và ba điều kiện trùng lặp $DCond_{1-3}$, ta có *Định lý 3.2* cho câu trả lời phủ định với câu hỏi này. *Khẳng định b)* của định lý này chỉ ra cấu trúc của $\mathcal{FS}$ dựa vào $\mathcal{FCS}$ và $\mathcal{FGS}$ thông qua các tập $CS(ro)$ chứa các chuỗi đóng phổ biến σ trong từng lớp ro và các chuỗi sinh phổ biến $GenSet(\sigma)$ của σ. Đó là cơ sở lý thuyết để thiết kế một thuật toán hiệu quả khai thác $\mathcal{FS}$ từ hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$.

Định lý 3.2 (*Biểu diễn tường minh và không trùng lặp các chuỗi phổ biến trong mỗi tập* $\mathcal{FS}'(\sigma,\gamma)$ *và trong tập* $\mathcal{FS}$).

a) $\mathcal{FS}'(\sigma,\gamma) = \mathcal{FS}''(\sigma,\gamma)$.

b) Tất cả chuỗi phổ biến trong lớp $\mathcal{FS}''(\sigma,\gamma)$ được sinh không trùng lặp. Do đó, tất cả chuỗi phổ biến trong tập $\mathcal{FS} = \sum_{ro \in RO} \sum_{\sigma_i \in CS(ro)} \sum_{\gamma_j \in GenSet(\sigma_i)} \mathcal{FS}^*(\sigma_i, \gamma_j)$ được sinh đầy đủ và không trùng lặp, trong đó $\mathcal{FS}^*(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}''(\sigma_i, \gamma_j) \mid not(DCondC(\alpha, \sigma_i, CS(\rho(\sigma_i))))$ và $not(DCondG(\alpha, \sigma_i, \gamma_j))\}$.

3.3. Thuật toán *FS-Miner* và kết quả thực nghiệm

Từ những kết quả lý thuyết trên, luận án thiết kế thuật toán *FS-Miner* để sinh ra nhanh tập $\mathcal{FS}$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$.

3.3.1. Thuật toán *FS-Miner*

Từ các tập $\mathcal{FCS}^* \stackrel{\text{def}}{=} \{(\sigma, \rho(\sigma), supp(\sigma)) \mid \sigma \in \mathcal{FCS}\}$ và $\mathcal{FGS}^* \stackrel{\text{def}}{=} \{(\gamma, supp(\gamma)) \mid \gamma \in \mathcal{FGS}\}$, với mỗi $ro \in RO \stackrel{\text{def}}{=} \{\rho(\sigma) \mid \sigma \in \mathcal{FCS}\}$, các danh sách tương ứng $\mathcal{FCGS}(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet(\sigma), supp(\sigma)) \mid \sigma \in \mathcal{FCS}, \rho(\sigma) = ro\}$ và $\mathcal{FCGS} \stackrel{\text{def}}{=} \{\mathcal{FCGS}(ro) \mid ro \in RO\}$ được tạo ra và làm đầu vào cho *FS-Miner* (*Hình 3.4*).

```
FS FS-Miner(FCGS)
1. FS = ∅;
2. for each FCGS(ro) ∈ FCGS do
3.     for each (σ_i, GenSet(σ_i), supp(σ_i)) ∈ FCGS(ro) do
4.         for each γ_k ∈ GenSet(σ_i) do {
5.             FS*(σ_i, γ_k) = FSMiner-OneSubClass(σ_i, γ_k);
6.             FS = FS + FS*(σ_i, γ_k);
7.         }
8. return FS;
```

Hình 3.4. Thuật toán *FS-Miner*.

Trong thuật toán này, với mỗi cặp chuỗi đóng σ_i và chuỗi sinh γ_k của nó, thủ tục *FSMiner-OneSubClass* (*Hình 3.5*) được gọi để khai thác các chuỗi phổ biến trong mỗi tập $\mathcal{FS}^*(\sigma_i, \gamma_k)$ (dòng 5) và sau đó tập $\mathcal{FS}^*(\sigma_i, \gamma_k)$ được thêm vào $\mathcal{FS}$ (dòng 6). Cuối cùng, *FS-Miner* cho ra tập kết quả $\mathcal{FS}$ (dòng 8).

3.3.2. Kết quả thử nghiệm

Trong thực nghiệm, tính đúng của thuật toán *FS-Miner* đã được kiểm tra lại bằng cách so sánh kết quả khai thác được của nó với các thuật toán tiêu biểu trước đây như *CM-SPADE*, *CM-SPAM* [17] và *PrefixSpan* [63] để khai thác các chuỗi phổ biến trực tiếp từ *SDB*. Ngoài ra, luận án cũng đánh giá hiệu quả về thời gian chạy của *FS-Miner* so với ba thuật toán này. Các kết quả thử nghiệm cho thấy rằng, tập kết quả được sinh ra bởi *FS-Miner* luôn trùng khớp với tập kết quả được khai thác bởi *CM-SPAM*, *CM-SPADE* và *PrefixSpan*, và thời gian chạy của *FS-Miner* thường bé hơn nhiều so với các thuật toán còn lại. Trên tất cả bốn *SDB*, *FS-Miner* nhanh hơn khoảng 3-27 lần so với *CM-SPAM,* 4-50 lần so với *CM-SPADE* và 4-117 lần so với *PrefixSpan*. Điều này khẳng định rằng các kết quả lý thuyết được đề xuất trong chương này không chỉ có ý nghĩa về mặt lý thuyết trong việc chỉ ra cấu trúc và biểu diễn không trùng lặp của các chuỗi phổ biến dựa trên các chuỗi đóng và chuỗi sinh phổ biến mà chúng còn là cơ sở để thiết kế thuật toán *FS-Miner* sinh ra (hay phục hồi) nhanh tập $\mathcal{FS}$ từ các tập $\mathcal{FCS}$ và $\mathcal{FGS}$. Điều này một lần nữa khẳng định lại các tập $\mathcal{FCS}$ và $\mathcal{FGS}$ là biểu diễn không mất thông tin cho $\mathcal{FS}$.

Trong chương tiếp theo, luận án sẽ áp dụng cấu trúc được đưa ra trong chương này để phát triển các phương pháp mới nhằm khai thác hiệu quả tập $\mathcal{FS}^{\mathcal{C}}$.

Chương 4 - Khai thác các chuỗi phổ biến với ràng buộc

Hầu hết các thuật toán trước đây khai thác $\mathcal{FS}^{\mathcal{C}}$ (tập tất cả các chuỗi phổ biến thỏa mãn ràng buộc $\mathcal{C}$) trực tiếp từ các *SDB* nên hiệu quả của chúng không cao khi $\mathcal{C}$ thuộc lớp ràng buộc phức tạp C^M hoặc $\mathcal{C}$ được thay đổi thường xuyên bởi người dùng. Để nâng cao hiệu quả khai thác tập $\mathcal{FS}^{\mathcal{C}}$ và đưa ra câu trả lời khẳng định cho các câu hỏi Q_1 - Q_3 được nêu ra trong phần mục tiêu của luận án, chương này đưa ra một hướng tiếp cận mới để khai thác tập $\mathcal{FS}^{\mathcal{C}}$ từ hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ dựa trên những kết quả lý thuyết được đề xuất trong *Chương 3*. Phương pháp được đề xuất khắc phục được những nhược điểm quan trọng của các phương pháp tiêu biểu trước đây.

4.1.Các bài toán

Trong chương này, luận án quan tâm đến hai bài toán tổng quát liên quan đến hai lớp ràng buộc C^A và C^M thường gặp rất nhiều trong thực tế.

Cho trước *SDB* $\mathcal{D}$, ngưỡng hỗ trợ tối thiểu $minsupp$ và tập $\mathcal{FS} \stackrel{\text{def}}{=} \{\alpha \mid supp(\alpha) \geqslant minsupp\}$ chứa tất cả các chuỗi phổ biến, bài toán đầu tiên được phát biểu như sau.

$\mathcal{BT}_{3.1}$: Tìm tập $\mathcal{FS}^A$ gồm tất cả các chuỗi phổ biến α (trong $\mathcal{D}$) thỏa mãn một ràng buộc C_A bất kỳ thuộc lớp C^A, nghĩa là ta cần tìm tập $\mathcal{FS}^A \stackrel{\text{def}}{=} \{\alpha \mid supp(\alpha) \geqslant minsupp \wedge C_A(\alpha)\} = \{\alpha \in \mathcal{FS} \mid C_A(\alpha)\}$.

Ràng buộc C_A thuộc lớp C^A có thể là một trong những loại ràng buộc như *ràng buộc chuỗi con*, *ràng buộc thuộc tính*, v.v. [64,65]. Một ràng buộc chuỗi con, được ký hiệu là C_{SubS}, có dạng $C_{SubS}(\alpha) \equiv (\alpha \sqsubseteq \Gamma_1)$, trong đó Γ_1 là một chuỗi cho trước, nghĩa là tất cả các chuỗi α cần khai thác phải là chuỗi con (sub-sequence) của Γ_1. Ràng buộc thuộc tính, được ký hiệu là C_I, thuộc

lớp C^A có dạng $C_I(\alpha) \equiv (\alpha[k] \trianglelefteq \mathcal{E}, \forall k: 1 \leq k \leq size(\alpha))$, trong đó $\mathcal{E}$ là tập thuộc tính cho trước, $\alpha[k]$ là sự kiện thứ *k* của α và toán tử quan hệ $\trianglelefteq \in \{\sqsubseteq, \not\supseteq\}$.

Bài toán tổng quát thứ hai được đưa ra trong chương này được phát biểu như sau:

$\mathcal{BT}_{3.2}$: Tìm tập $\mathcal{FS}^M$ gồm tất cả các chuỗi phổ biến α (trong $\mathcal{D}$) thỏa mãn một ràng buộc C_M bất kỳ thuộc lớp C^M, nghĩa là ta cần tìm tập $\mathcal{FS}^M \stackrel{\text{def}}{=} \{\alpha \mid supp(\alpha) \geqslant minsupp \wedge C_M(\alpha)\} = \{\alpha \in \mathcal{FS} \mid C_M(\alpha)\}$.

Trong [64,65], các tác giả đã trình bày nhiều loại ràng buộc khác nhau, trong đó có một số loại ràng buộc thuộc lớp C^M, chẳng hạn như *ràng buộc chiều dài*, *ràng buộc thuộc tính*, *ràng buộc chuỗi cha*, v.v. Một ràng buộc chiều dài đòi hỏi rằng chiều dài của một chuỗi cần khai thác phải lớn hơn hoặc bằng một số nguyên dương cho trước. Ràng buộc thuộc tính C_I thuộc lớp C^M có dạng $C_I(\alpha) \equiv (\alpha[k] \trianglerighteq \mathcal{E}, \forall k: 1 \leq k \leq size(\alpha))$, $C_I(\alpha) \equiv (\alpha[k] \trianglerighteq \mathcal{E}, \exists k: 1 \leq k \leq size(\alpha))$, $C_I(\alpha) \equiv (\alpha[k] \cap \mathcal{E} \neq \emptyset, \forall k: 1 \leq k \leq size(\alpha))$ hoặc $C_I(\alpha) \equiv (\alpha[k] \cap \mathcal{E} \neq \emptyset, \exists k: 1 \leq k \leq size(\alpha))$, trong đó $\mathcal{E}$ là tập thuộc tính cho trước và toán tử quan hệ $\trianglerighteq \in \{\sqsupseteq, \not\subseteq\}$. Một ràng buộc chuỗi cha, kí hiệu là C_{SupS}, có dạng $C_{SupS}(\alpha) \equiv (\Gamma_0 \sqsubseteq \alpha)$, trong đó Γ_0 là một chuỗi cho trước, nghĩa là tất cả các chuỗi α cần khai thác phải là chuỗi cha của Γ_0.

Để giải hai bài toán $\mathcal{BT}_{3.1}$ và $\mathcal{BT}_{3.2}$, ta có thể sử dụng phương pháp hậu xử lý hoặc phương pháp tích hợp ràng buộc vào một trong các thuật toán đã biết. Tuy nhiên, cả hai phương pháp này vẫn còn nhiều hạn chế như được thảo luận ở trên. Vì vậy, trong các *Mục 4.2 – 4.3*, luận án đề xuất các phương pháp mới để giải quyết hiệu quả hai bài toán này dựa trên ý tưởng của phương pháp phân hoạch trong *Chương 3* kết hợp với kỹ thuật tích hợp ràng buộc.

4.2. Phương pháp giải bài toán tổng quát $\mathcal{BT}_{3.1}$

4.2.1. Phân hoạch tập lời giải $\mathcal{FS}^A$

Trong *Chương 3*, dựa trên ý tưởng phân hoạch, luận án đã đề xuất phương pháp để sinh ra nhanh tập $\mathcal{FS}$ từ cặp $(\mathcal{FCS}, \mathcal{FGS})$ thông qua việc sinh ra tất các chuỗi phổ biến α trong mỗi tập con $\mathcal{FS}(\sigma, \gamma)$ với $\sigma \in \mathcal{FCS}$ và $\gamma \in GenSet(\sigma)$ sao cho $\gamma \sqsubseteq \alpha \sqsubseteq \sigma$ và $\rho(\gamma) = \rho(\alpha) = \rho(\sigma)$. Khi áp dụng phương pháp này để khai thác tập $\mathcal{FS}^A$ bằng cách tích hợp ràng buộc C_A vào trong quá trình khai thác, ta quan sát rằng nếu một chuỗi sinh γ không thỏa C_A thì tất cả các chuỗi phổ biến α được sinh ra trong tập $\mathcal{FS}(\sigma, \gamma)$ cũng không thỏa ràng buộc C_A. Bởi vậy, để giảm số tập con $\mathcal{FS}(\sigma, \gamma)$ cần khai thác, ta nên loại ngay từ đầu các chuỗi sinh $\gamma \in GenSet(\sigma)$ (do đó tất cả các tập con $\mathcal{FS}(\sigma, \gamma)$ tương ứng cũng được loại bỏ) mà γ không thỏa C_A. Đây là một trong những ưu điểm nổi trội của việc tích hợp ràng buộc C_A vào phương pháp phân hoạch. Dựa trên quan sát này và phương pháp phân hoạch trong *Chương 3*, luận án trình bày phương pháp phân hoạch dựa trên ràng buộc để giải bài toán tổng quát $\mathcal{BT}_{3.1}$ như sau.

Đặt $GenSet^A(\sigma) \stackrel{\text{def}}{=} \{\gamma \in GenSet(\sigma) \mid C_A(\gamma)\}$, $RO^A \stackrel{\text{def}}{=} \{\rho(\sigma) \mid \sigma \in \mathcal{FCS} \wedge GenSet^A(\sigma) \neq \emptyset\}$, $FS^A(ro) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^A \mid \rho(\alpha) = ro\}$ và $CS^A(ro) \stackrel{\text{def}}{=} \{\sigma \in \mathcal{FCS} \mid \rho(\sigma) = ro \wedge GenSet^A(\sigma) \neq \emptyset\}$, $\forall ro \in RO^A$. Chú ý rằng $FS^A(ro)$ là lớp tương đương của các chuỗi phổ biến thỏa mãn ràng buộc C_A ứng với mỗi $ro \in RO^A$. Khi đó, $\forall \sigma_i \in CS^A(ro)$, $\forall \gamma_j \in GenSet^A(\sigma_i)$, luận án định nghĩa các tập con sau: $\mathcal{FS}^A(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \sqsubseteq \sigma_i \mid C_A(\alpha) \wedge \rho(\alpha) = \rho(\sigma_i)\}$, $\mathcal{FS}^A(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^A(\sigma_i) \mid \alpha \sqsupseteq \gamma_j\} = \{\alpha \mid \gamma_j \sqsubseteq \alpha \sqsubseteq \sigma_i \wedge C_A(\alpha)\}$, $\mathcal{FS}^{*A}(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^A(\sigma_i) \mid not(DCondC(\alpha, \sigma_i))\}$, $\mathcal{FS}^{*A}(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^{*A}(\sigma_i) \mid \alpha \sqsupseteq \gamma_j \wedge not(DCondG(\alpha, \sigma_i, \gamma_j))\} = \{\alpha \in \mathcal{FS}^A(\sigma_i, \gamma_j) \mid not(DCondC(\alpha, \sigma_i)) \wedge not(DCondG(\alpha, \sigma_i, \gamma_j))\}$, trong đó $DCondC(\alpha, \sigma_i)$ và $DCondG(\alpha, \sigma_i, \gamma_j)$ là những điều kiện phát hiện trùng lặp như được trình bày trong *Mục*

3.2.3. Từ biểu diễn của $FS(ro)$ trong *Mục 3.2.1*, ta có ngay các biểu diễn của $FS^A(ro)$ và $\mathcal{FS}^A(\sigma_i)$ như sau:

$$\mathcal{FS}^A(\sigma_i) = \bigcup_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^A(\sigma_i, \gamma_j),\ FS^A(ro) = \bigcup_{\sigma_i \in CS^A(ro)} \mathcal{FS}^A(\sigma_i) = \bigcup_{\sigma_i \in CS^A(ro)} \bigcup_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^A(\sigma_i, \gamma_j).$$

Dựa trên các biểu diễn này, ta có *Định lý 4.1* dưới đây.

Định lý 4.1 (*Các phân hoạch của* $\mathcal{FS}^A$).

a) $\mathcal{FS}^A = \sum_{ro \in RO^A} FS^A(ro)$ (*phân hoạch thô của* $\mathcal{FS}^A$).

b) $\forall ro \in RO^A$, $\forall \sigma_i \in CS^A(ro)$, $\forall \gamma_j \in GenSet^A(\sigma_i)$, ta có phân hoạch mịn của $FS^A(ro)$ và $\mathcal{FS}^A$ như sau:

$$FS^A(ro) = \sum_{\sigma_i \in CS^A(ro)} \sum_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^{*A}(\sigma_i, \gamma_j) \text{ và}$$
$$\mathcal{FS}^A = \sum_{ro \in RO^A} \sum_{\sigma_i \in CS^A(ro)} \sum_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^{*A}(\sigma_i, \gamma_j),$$

trong đó $FS^A(ro) = \sum_{\sigma_i \in CS^A(ro)} \mathcal{FS}^{*A}(\sigma_i)$ và $\mathcal{FS}^{*A}(\sigma_i) = \sum_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^{*A}(\sigma_i, \gamma_j)$.

Chú ý rằng sự biểu diễn của các chuỗi trong mỗi tập $\mathcal{FS}^A(\sigma_i, \gamma_j)$ xuất hiện trong định nghĩa của $\mathcal{FS}^{*A}(\sigma_i, \gamma_j)$ là không tường minh (nghĩa là nó không chỉ ra cách thức sinh cụ thể các chuỗi trong tập $\mathcal{FS}^A(\sigma_i, \gamma_j)$). Tập $\mathcal{FS}'^A(\sigma, \gamma)$ dưới đây chỉ ra một biểu diễn tường minh của $\mathcal{FS}^A(\sigma, \gamma)$.

$$\mathcal{FS}'^A(\sigma, \gamma) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}'(\sigma, \gamma) \mid C_A(\alpha)\} = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'(\sigma, \gamma) \mid lp \in LP(\sigma, \gamma), \delta(lp) \sqsubseteq \Delta(lp), C_A(\alpha)\}.$$

Ưu điểm nổi bật của phương pháp phân hoạch được đưa ra trong *Định lý 4.1* là thông qua việc chỉ kiểm tra ràng buộc trên số ít các chuỗi sinh, ta có thể loại bỏ nhiều lớp tương đương $FS(ro)$, tập con $\mathcal{FS}^*(\sigma)$ ở mức 1 hoặc tập con $\mathcal{FS}^*(\sigma, \gamma)$ ở mức 2 như được minh họa trong *Ví dụ 4.2* dưới đây.

Ví dụ 4.2 (*Minh họa ưu điểm của phương pháp phân hoạch để tìm tập* $\mathcal{FS}^A$).

Với *SDB* cho trong *Bảng 2.2* và $minsupp = 2$, ta có $RO = \{ro_k, k = 1..6\}$, trong đó $ro_1 = \{\Psi_1, \Psi_2\}$, $ro_2 = \{\Psi_1, \Psi_2, \Psi_4\}$, $ro_3 = \{\Psi_1, \Psi_4\}$, $ro_4 = \{\Psi_1, \Psi_2, \Psi_3, \Psi_4\}$, $ro_5 = \{\Psi_1, \Psi_3\}$ và $ro_6 = \{\Psi_1, \Psi_2, \Psi_3\}$. Xét loại ràng buộc $C_A \equiv C_I$ thuộc lớp C^A với $\trianglelefteq \equiv \subseteq$ và $\mathcal{E}$ = *aeg*. *Hình 4.1* minh họa ưu điểm của việc sử dụng phương pháp phân hoạch để tìm tập $\mathcal{FS}^E$ so với phương pháp hậu xử lý.

Ta có, $\forall ro_k \in RO \backslash \{ro_1, ro_2, ro_4\}$, $\forall \sigma_i \in CS(ro_k)$ thì $GenSet^E(\sigma_i) = \emptyset$, nên $CS^E(ro_k) = \emptyset$ và $RO^E = \{ro_1, ro_2, ro_4\}$. Do đó, $\mathcal{FS}^E = FS^E(ro_1) + FS^E(ro_2) + FS^E(ro_4)$. Vì vậy, ta chỉ cần xét ba lớp $FS^E(ro_1)$, $FS^E(ro_2)$ và $FS^E(ro_4)$ với ràng buộc thuộc tính C_I, thay vì phải xét sáu lớp $\{FS(ro_k), k = 1..6\}$ trong $\mathcal{FS} = \sum_{k=1..6} FS(ro_k)$ nếu dùng phương pháp hậu xử lý.

Với lớp $FS^E(ro_2)$, ta có $CS(ro_2) = \{\sigma_1 = ace{\rightarrow}a{\rightarrow}af, \sigma_2 = d{\rightarrow}ac{\rightarrow}a, \sigma_3 = d{\rightarrow}af\}$. Vì $\forall i = 2..3$, $\sigma_i \in CS(ro_2)$, $GenSet(\sigma_i) = \{\gamma = d\}$, ràng buộc $C_I(\gamma)$ không thỏa mãn, nên $GenSet^E(\sigma_i) = \emptyset$, $\sigma_i \notin CS^E(ro_2)$. Với $\sigma_1 \in CS(ro_2)$, thì $GenSet(\sigma_1) = \{\gamma_1 = a{\rightarrow}a{\rightarrow}f, \gamma_2 = e{\rightarrow}a{\rightarrow}f, \gamma_3 = a{\rightarrow}a{\rightarrow}a, \gamma_4 = e{\rightarrow}a{\rightarrow}a\}$. Vì cả hai γ_1 và γ_2 không thỏa ràng buộc C_I nên $GenSet^E(\sigma_1) = \{\gamma_3, \gamma_4\} \subset GenSet(\sigma_1)$, $CS^E(ro_2) = \{\sigma_1\} \subset CS(ro_2)$ và $\mathcal{FS}^{*E}(\sigma_1) = \sum_{j=3..4} \mathcal{FS}^{*E}(\sigma_1, \gamma_j)$, trong khi $\mathcal{FS}^*(\sigma_1) = \sum_{j=1..4} \mathcal{FS}^*(\sigma_1, \gamma_j)$, nghĩa là, với $\sigma_1 \in CS(ro_2)$, ta chỉ cần khai thác hai tập con $\{\mathcal{FS}^{*E}(\sigma_1, \gamma_j), j = 3..4\}$ ở mức hai với ràng buộc C_I thay vì xét bốn tập con $\{\mathcal{FS}^*(\sigma_1, \gamma_j), j = 1..4\}$ khi sử dụng phương pháp hậu xử lý. Do đó, $FS^E(ro_2) = \sum_{j=3..4} \mathcal{FS}^{*E}(\sigma_1, \gamma_j)$, nghĩa là ta chỉ phải khai thác một tập con $\mathcal{FS}^{*E}(\sigma_1)$ ở mức một với ràng buộc C_I thay vì phải xét ba tập $\{\mathcal{FS}^*(\sigma_i), i = 1..3\}$ trong $FS(ro_2) = \sum_{i=1..3} \mathcal{FS}^*(\sigma_i)$ khi dùng phương pháp hậu xử lý. Như

được đề cập ở trên, đây là một trong những ưu điểm nổi trội của việc tích hợp ràng buộc vào phương pháp phân hoạch để tỉa sớm nhiều tập con bằng cách chỉ kiểm tra ràng buộc trên số ít các chuỗi sinh. Với hai tập con còn lại $\mathcal{FS}^{*E}(\sigma_1, \gamma_3)$ and $\mathcal{FS}^{*E}(\sigma_1, \gamma_4)$, bằng việc sử dụng phương pháp sinh các chuỗi phổ biến α như được trình bày trong *Mục 3.2.3* của *Chương 3* và kiểm tra ràng buộc $C_I(\alpha)$, ta có được tập kết quả $FS^E(ro_2) = \mathcal{FS}^{*E}(\sigma_1, \gamma_3) + \mathcal{FS}^{*E}(\sigma_1, \gamma_4) = \{a{\rightarrow}a{\rightarrow}a, e{\rightarrow}a{\rightarrow}a, ae{\rightarrow}a{\rightarrow}a\}$. Tương tự, khi khai thác hai lớp $FS^E(ro_1)$ và $FS^E(ro_4)$ ta sẽ có thêm được 10 chuỗi thỏa C_I. Cuối cùng, ta có tập lời giải $\mathcal{FS}^E$ với $|\mathcal{FS}^E| = 13$ chuỗi.

4.2.2. Sinh đầy đủ và không trùng lặp các chuỗi trong $\mathcal{FS}^A(\sigma, \gamma)$ và $\mathcal{FS}^A$

Khi sinh tất cả các chuỗi trong tập $\mathcal{FS}'(\sigma_i, \gamma_j)$, chú ý rằng nếu $|LP(\sigma, \gamma')| > 1$ hay $\Delta(lp)[i] \cap \Delta(lp)[k] \neq \emptyset$, với bất kỳ $i, k \notin lp$: $i \neq k$, thì nhiều chuỗi được sinh ra trong $\mathcal{FS}'(\sigma_i, \gamma_j)$ có thể bị trùng lặp (xem *Ví dụ 3.5 – Ví dụ 3.7* trong *Chương 3*). Để giải quyết vấn đề này, luận án đề xuất tập $\mathcal{FS}''^A(\sigma, \gamma)$, biểu diễn không trùng lặp của $\mathcal{FS}'^A(\sigma, \gamma)$ hay biểu diễn tường minh và không trùng lặp của $\mathcal{FS}^A(\sigma, \gamma)$. Tập $\mathcal{FS}''^A(\sigma, \gamma)$ được định nghĩa như sau:

$$\mathcal{FS}''^A(\sigma, \gamma) \stackrel{\text{def}}{=} \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^A(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp))\}.$$

Vì vậy, $\mathcal{FS}^{*A}(\sigma, \gamma) = \{\alpha \in \mathcal{FS}''^A(\sigma, \gamma) \mid not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))\} = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^A(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))\}$.

Dựa trên các tập con $\mathcal{FS}''^A(\sigma, \gamma)$ và $\mathcal{FS}^{*A}(\sigma, \gamma)$, ta có *Định lý 4.2* dưới đây.

Định lý 4.2 (*Biểu diễn tường minh và không trùng lặp của $\mathcal{FS}^A(\sigma, \gamma)$ và $\mathcal{FS}^A$*).

a) ($\mathcal{FS}''^A(\sigma, \gamma)$ *– biểu diễn tường minh và không trùng lặp của $\mathcal{FS}^A(\sigma, \gamma)$*).
$\forall ro \in RO^A$, $\forall \sigma \in CS^A(ro)$, $\forall \gamma \in GenSet^A(\sigma)$, thì $\mathcal{FS}^A(\sigma, \gamma) = \mathcal{FS}'^A(\sigma, \gamma) = \mathcal{FS}''^A(\sigma, \gamma)$ và $\mathcal{FS}^{*A}(\sigma, \gamma) = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^A(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))\}$.

b) Tất cả các chuỗi trong tập $\mathcal{FS}''^A(\sigma, \gamma)$ được sinh ra không trùng lặp. Vì vậy, tất cả các chuỗi trong tập
$$\mathcal{FS}^A = \sum_{ro \in RO^A} \sum_{\sigma_i \in CS^A(ro)} \sum_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^{*A}(\sigma_i, \gamma_j)$$
cũng được sinh ra đầy đủ, tường minh và không trùng lặp.

4.2.3. Thuật toán *MFS-AC*

Dựa vào các kết quả lý thuyết trên, luận án đề xuất thuật toán *MFS-AC* để khai thác các chuỗi phổ biến với bất kỳ loại ràng buộc nào thuộc lớp C^A (bài toán $\mathcal{BT}_{3.1}$). Từ hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$, ta có thể dễ dàng xác định được tập $\mathcal{FCGS} \stackrel{\text{def}}{=} \{(\mathcal{FCGS}(ro), |ro|) \mid ro \in RO\}$ (đầu vào của *MFS-AC*), trong đó $\mathcal{FCGS}(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet(\sigma)) \mid \sigma \in \mathcal{FCS}, \rho(\sigma) = ro\}$. Với mỗi $ro \in RO$, từ tập $\mathcal{FCGS}(ro)$, ta sinh ra tập $\mathcal{FCGS}^{*A}(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet^A(\sigma)) \mid (\sigma, GenSet(\sigma)) \in \mathcal{FCGS}(ro) \wedge GenSet^A(\sigma) \neq \emptyset\}$, trong đó $GenSet^A(\sigma) \stackrel{\text{def}}{=} \{\gamma \in GenSet(\sigma) \mid C_A(\gamma)\}$. Đặt $RO^A \stackrel{\text{def}}{=} \{ro \in RO \mid \mathcal{FCGS}^{*A}(ro) \neq \emptyset\}$ và $\mathcal{FCGS}^A \stackrel{\text{def}}{=} \{(\mathcal{FCGS}^{*A}(ro), |ro|) \mid ro \in RO^A\}$. Khi đó, ta có $CS^A(ro) = \{\sigma \mid (\sigma, GenSet^A(\sigma)) \in \mathcal{FCGS}^{*A}(ro)\}$ và $supp(\sigma) = |ro|, \forall \sigma \in CS^A(ro)$. *MFS-AC* được thiết kế dựa chính trên hai công thức sau:

$\mathcal{FS}^{*A}(\sigma, \gamma) = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^A(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))\}$ và

$\mathcal{FS}^A = \sum_{ro \in RO^A} \sum_{\sigma_i \in CS^A(ro)} \sum_{\gamma_j \in GenSet^A(\sigma_i)} \mathcal{FS}^{*A}(\sigma_i, \gamma_j)$.

Khi đó, $supp(\alpha) = |ro|$, $\forall \alpha \in \mathcal{FS}^{*A}(\sigma,\gamma)$: $\gamma \in GenSet^A(\sigma)$ và $(\sigma, GenSet^A(\sigma)) \in \mathcal{FCGS}^{*A}(ro)$.

Mã giả của thuật toán *MFS-AC* được mô tả trong *Hình 4.2*. Tại dòng 5 của hình này, với mỗi cặp chuỗi (σ_i, γ_k), thuật toán *MFS-AC* gọi thủ tục *MFS-AC-FromOnePair* trong *Hình 4.3* để khai thác các chuỗi phổ biến với ràng buộc C_A trong tập $\mathcal{FS}^{*A}(\sigma_i, \gamma_k)$. Cuối cùng, thủ tục *MFS-AC-FromOnePair* trả về tập kết quả $\mathcal{FS}^{*A}(\sigma_i, \gamma_k)$ và thuật toán *MFS-AC* cho ra kết quả cuối cùng là tập $\mathcal{FS}^A$.

```
FS^A MFS-AC (FCGS, C_A)
1. Tìm FCGS^A ≝ {FCGS^A(ro) | ro ∈ RO^A} từ FCGS;
2. for each (FCGS^A(ro), |ro|) ∈ FCGS^A do
3.   for each (σ_i, GenSet^A(σ_i)) ∈ FCGS^A(ro) do
4.     for each γ_k ∈ GenSet^A(σ_i) do {
5.         FS^{*A}(σ_i, γ_k) = MFS-AC-FromOnePair(σ_i, γ_k);
6.         FS^A = FS^A ∪ FS^{*A}(σ_i, γ_k);
7.     }
8. return FS^A;
```

Hình 4.1. Thuật toán *MFS-AC*.

4.3. Phương pháp giải bài toán tổng quát $\mathcal{BT}_{3.2}$

4.3.1. Phân hoạch tập lời giải $\mathcal{FS}^M$

Với phương pháp phân hoạch được đề xuất trong *Chương 3*, tất cả các chuỗi phổ biến α trong tập $\mathcal{FS}(\sigma,\gamma)$ được sinh ra dựa trên mối quan hệ tường minh giữa α với cặp chuỗi đóng σ và chuỗi sinh γ của nó, sao cho $\gamma \sqsubseteq \alpha \sqsubseteq \sigma$ và $\rho(\gamma) = \rho(\alpha) = \rho(\sigma)$, trong đó $\sigma \in \mathcal{FCS}$ và $\gamma \in GenSet(\sigma)$. Để áp dụng một cách hiệu quả phương pháp phân hoạch này cho việc giải bài toán $\mathcal{BT}_{3.2}$, luận án nhận thấy rằng nếu một chuỗi đóng σ không thỏa ràng buộc C_M thì tất cả các chuỗi phổ biến α trong mọi tập con $\mathcal{FS}(\sigma,\gamma)$ với $\gamma \in GenSet(\sigma)$ cũng không thỏa C_M. Do đó, ta có thể loại bỏ sớm toàn bộ các chuỗi trong các tập con $\mathcal{FS}(\sigma,\gamma)$ mà không cần sinh ra chúng. Nói cách khác, ta chỉ cần xét các chuỗi đóng $\sigma \in \mathcal{FCS}$ mà ràng buộc $C_M(\sigma)$ được thỏa mãn, nghĩa là ta vẫn có thể tích hợp hiệu quả việc kiểm tra ràng buộc C_M vào quá trình khai thác $\mathcal{FS}^M$. Đây chính là một trong những ưu điểm nổi bật của phương pháp phân hoạch dựa trên ràng buộc so với các phương pháp trước đây trong việc giải bài toán $\mathcal{BT}_{3.2}$. Dựa trên nhận xét này, phương pháp phân hoạch dựa trên ràng buộc C_M dưới đây được đưa ra.

Đặt $RO^M \stackrel{\text{def}}{=} \{\rho(\sigma) \mid \sigma \in \mathcal{FCS} \wedge C_M(\sigma)\}$. Với mỗi $ro \in RO^M$, ta gọi $\mathcal{FS}^M(ro) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^M \mid \rho(\alpha) = ro\}$ ($\subseteq FS(ro)$) là lớp tương đương (theo quan hệ $\sim$) của các chuỗi phổ biến thỏa ràng buộc C_M và $CS^M(ro) \stackrel{\text{def}}{=} \{\sigma \in \mathcal{FCS} \mid \rho(\sigma) = ro \wedge C_M(\sigma)\}$ là tập tất cả các chuỗi đóng có cùng ro thỏa ràng buộc C_M. Với $\forall \sigma_i \in CS^M(ro)$, $\forall \gamma_j \in Genset(\sigma_i)$, ta định nghĩa các tập con sau: $\mathcal{FS}^M(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \sqsubseteq \sigma_i \mid C_M(\alpha) \wedge \rho(\alpha) = \rho(\sigma_i)\}$, $\mathcal{FS}^M(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^M(\sigma_i) \mid \alpha \sqsupseteq \gamma_j\} = \{\alpha \mid \gamma_j \sqsubseteq \alpha \sqsubseteq \sigma_i \wedge C_M(\alpha)\}$, $\mathcal{FS}^{*M}(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^M(\sigma_i) \mid not(DCondC(\alpha, \sigma_i))\}$ và $\mathcal{FS}^{*M}(\sigma_i, \gamma_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^{*M}(\sigma_i) \mid \alpha \sqsupseteq \gamma_j \wedge not(DCondG(\alpha, \sigma_i, \gamma_j))\} = \{\alpha \in \mathcal{FS}^M(\sigma_i, \gamma_j) \mid not(DCondC(\alpha, \sigma_i)) \wedge not(DCondG(\alpha, \sigma_i, \gamma_j))\}$. Khi đó, ta có biểu diễn của lớp $\mathcal{FS}^M(ro)$ và $\mathcal{FS}^M(\sigma_i)$ như sau:

$\mathcal{FS}^M(\sigma_i) = \bigcup_{\gamma_j \in Genset(\sigma_i)} \mathcal{FS}^M(\sigma_i, \gamma_j)$ và

$\mathcal{FS}^M(ro) = \bigcup_{\sigma_i \in CS^M(ro)} \mathcal{FS}^M(\sigma_i) = \bigcup_{\sigma_i \in CS^M(ro)} \bigcup_{\gamma_j \in Genset(\sigma_i)} \mathcal{FS}^M(\sigma_i, \gamma_j)$.

Dựa trên các biểu diễn này, ta có *Định lý 4.3* sau đây.

Định lý 4.3 (*Phân hoạch của các tập $\mathcal{FS}^M(ro)$ và $\mathcal{FS}^M$*). $\forall ro \in RO^M$, $\forall \sigma_i \in CS^M(ro)$, $\forall \gamma_j \in Genset(\sigma_i)$, ta có các biểu diễn sau:

$$\mathcal{FS}^M(ro) = \sum_{\sigma_i \in CS^M(ro)} \mathcal{FS}^{*M}(\sigma_i) \text{ và } \mathcal{FS}^{*M}(\sigma_i) = \sum_{\gamma_j \in Genset(\sigma_i)} \mathcal{FS}^{*M}(\sigma_i, \gamma_j).$$

Khi đó, ta có các phân hoạch mịn của $\mathcal{FS}^M(ro)$ và $\mathcal{FS}^M$ như sau:

$$\mathcal{FS}^M(ro) = \sum_{\sigma_i \in CS^M(ro)} \sum_{\gamma_j \in Genset(\sigma_i)} \mathcal{FS}^{*M}(\sigma_i, \gamma_j),$$
$$\mathcal{FS}^M = \sum_{ro \in RO^M} \sum_{\sigma_i \in CS^M(ro)} \sum_{\gamma_j \in Genset(\sigma_i)} \mathcal{FS}^{*M}(\sigma_i, \gamma_j).$$

4.3.2. Sinh đầy đủ và không trùng lặp các chuỗi trong $\mathcal{FS}^M(\sigma, \gamma)$ và $\mathcal{FS}^M$

Chú ý rằng biểu biễn của các chuỗi trong mỗi tập con $\mathcal{FS}^M(\sigma, \gamma)$ xuất hiện trong định nghĩa của $\mathcal{FS}^{*M}(\sigma, \gamma)$ không chỉ ra phương pháp cụ thể để sinh ra các chuỗi trong $\mathcal{FS}^M(\sigma, \gamma)$. Vì vậy, tương tự như biểu diễn tường minh $\mathcal{FS}'(\sigma, \gamma)$ của tập $\mathcal{FS}(\sigma, \gamma)$ trong *Mệnh đề 3.2*, luận án đưa ra biểu diễn tường minh $\mathcal{FS}'^M(\sigma, \gamma)$ của tập $\mathcal{FS}^M(\sigma, \gamma)$ như sau: $\forall ro \in RO^M$, $\forall \sigma \in CS^M(ro)$, $\forall \gamma \in GenSet(\sigma)$,

$$\mathcal{FS}'^M(\sigma, \gamma) \stackrel{\text{def}}{=} \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'(\sigma, \gamma) \mid lp \in LP(\sigma, \gamma) \wedge \delta(lp) \sqsubseteq \Delta(lp) \wedge C_M(\alpha)\}.$$

Mặc dù các chuỗi trong $\mathcal{FS}'^M(\sigma, \gamma)$ được sinh ra một cách rõ ràng, nhưng nhiều chuỗi trong đó vẫn bị trùng lặp. Để khắc phục, các điều kiện phát hiện sớm sự trùng lặp $DCond_{1-3}$ trong *Chương 3* được sử dụng trong một biểu diễn khác $\mathcal{FS}''^M(\sigma, \gamma)$ của $\mathcal{FS}'^M(\sigma, \gamma)$:

$$\mathcal{FS}''^M(\sigma, \gamma) \stackrel{\text{def}}{=} \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^M(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp))\}.$$

Khi đó, $\mathcal{FS}^{*M}(\sigma, \gamma) = \{\alpha \in \mathcal{FS}''^M(\sigma, \gamma) \mid not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))$.

Dựa trên tập con $\mathcal{FS}''^M(\sigma, \gamma)$, *Định lý 4.4* dưới đây được đưa ra.

Định lý 4.4 (*Biểu diễn tường minh và không trùng lặp của $\mathcal{FS}^M(\sigma, \gamma)$ và $\mathcal{FS}^M$*).

a) ($\mathcal{FS}''^M(\sigma, \gamma)$, *biểu diễn tường minh và không trùng lặp của $\mathcal{FS}^M(\sigma, \gamma)$*).

Tất cả các chuỗi trong $\mathcal{FS}''^M(\sigma, \gamma)$ được sinh ra một cách tường minh và không trùng lặp. Với $\forall ro \in RO^M$, $\forall \sigma \in CS^M(ro)$, $\forall \gamma \in GenSet(\sigma)$, khi đó $\mathcal{FS}^M(\sigma, \gamma) = \mathcal{FS}'^M(\sigma, \gamma) = \mathcal{FS}''^M(\sigma, \gamma)$ và $\mathcal{FS}^{*M}(\sigma, \gamma) = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^M(\sigma, \gamma) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, \gamma))\}$.

b) Do đó, tất cả các chuỗi trong tập $\mathcal{FS}^M = \sum_{ro \in RO^M} \sum_{\sigma_i \in CS^M(ro)} \sum_{\gamma \in GenSet(\sigma_i)} \mathcal{FS}^{*M}(\sigma_i, \gamma_j)$ cũng được sinh ra đầy đủ và không trùng lặp.

Như được thảo luận ở trên, trong khi các thuật toán trước đây thường rất khó để sử dụng hiệu quả ràng buộc C_M cho việc rút gọn không gian tìm kiếm, ưu điểm nổi bật của phương pháp được đề xuất là cho phép phát hiện và loại bỏ sớm nhiều tập con $\mathcal{FS}(\sigma, \gamma)$ (không cần sinh ra tất cả các chuỗi phổ biến của chúng) nếu chuỗi đóng σ đại diện của chúng không thỏa mãn ràng buộc C_M. Điều này được minh họa cụ thể trong *Ví dụ 4.5*.

4.3.3. Thuật toán *MFS-MC*

Dựa trên các kết quả lý thuyết trong các *Định lý 4.3 – 4.4*, luận án đề xuất thuật toán *MFS-MC* để khai thác tất cả các chuỗi phổ biến thỏa ràng buộc C_M. Từ cặp ($\mathcal{FCS}$, $\mathcal{FGS}$) ta có thể dễ dàng xác định được tập $\mathcal{FCGS} \stackrel{\text{def}}{=} \{(\mathcal{FCGS}(ro), |ro|) \mid ro \in RO\}$ (đầu vào của thuật toán *MFS-MC*), trong đó $\mathcal{FCGS}(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet(\sigma)) \mid \sigma \in \mathcal{FCS}, \rho(\sigma) = ro\}$.

Với mỗi $ro \in RO$, ta xác định tập $\mathcal{FCGS}^{*M}(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet(\sigma)) \in \mathcal{FCGS}(ro) \wedge C_M(\sigma)\}$. Đặt $RO^M \stackrel{\text{def}}{=} \{ro \in RO \mid \mathcal{FCGS}^{*M}(ro) \neq \emptyset\}$, $\mathcal{FCGS}^M(ro) \stackrel{\text{def}}{=} \{(\sigma, GenSet(\sigma)) \mid (\sigma, GenSet(\sigma)) \in \mathcal{FCGS}^{*M}(ro)\}$ và $\mathcal{FCGS}^M \stackrel{\text{def}}{=} \{(\mathcal{FCGS}^M(ro), |ro|) \mid ro \in RO^M\}$. Khi đó, ta có $CS^M(ro) = \{\sigma$

$|\ (\sigma, GetSet(\sigma)) \in \mathcal{FCGS}^M(ro)\}$ và $supp(\sigma) = |ro|, \forall \sigma \in CS^M(ro)$. Thuật toán *MFS-MC* được thiết kế dựa trên hai công thức sau:

$$\mathcal{FS}^{*M}(\sigma,\gamma) = \{\alpha = Ex(\gamma, lp) \oplus \delta(lp) \in \mathcal{FS}'^M(\sigma,\gamma) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha,\sigma)) \wedge not(DCondG(\alpha,\sigma,\gamma))\}$$ và

$$\mathcal{FS}^M = \sum_{ro \in RO^M} \sum_{\sigma_i \in CS^M(ro)} \sum_{\gamma_j \in GenSet(\sigma_i)} \mathcal{FS}^{*M}(\sigma_i, \gamma_j).$$

Khi đó, $supp(\alpha) = |ro|, \forall \alpha \in \mathcal{FS}^{*M}(\sigma,\gamma)$, với bất kỳ $\gamma \in GenSet(\sigma)$ và $(\sigma, GenSet(\sigma)) \in \mathcal{FCGS}^M(ro)$. Mã giả của *MFS-MC* được mô tả trong *Hình 4.4*.

```
FS^M MFS-MC(FCGS, C_M)
1. Tìm FCGS^M ≝ {FCGS^M(ro) | ro ∈ RO^M} từ FCGS;
2. for each (FCGS^M(ro), |ro|) ∈ FCGS^M do
3.    for each (σ_i, GenSet(σ_i)) ∈ FCGS^M(ro) do
4.       for each γ_k ∈ GenSet(σ_i) do {
5.          FS^*M(σ_i, γ_k) = MFS-MC-FromOnePair(σ_i, γ_k);
6.          FS^M = FS^M ∪ FS^*M(σ_i, γ_k);
7.       }
8. return FS^M;
```

Hình 4.4. Thuật toán *MFS-MC* khai thác các chuỗi với lớp ràng buộc C^M.

Đầu tiên, thuật toán xác định tập $\mathcal{FCGS}^M$ từ tập $\mathcal{FCGS}$. Sau đó, với mỗi tập $\mathcal{FCGS}^M(ro) \in \mathcal{FCGS}^M$ và kế tiếp là mỗi cặp chuỗi (σ_i, γ_k), thuật toán *MFS-MC* gọi thủ tục *MFS-MC-FromOnePair* trong *Hình 4.5* để khai thác các chuỗi phổ biến trong tập $\mathcal{FS}^{*M}(\sigma_i, \gamma_k)$. Cuối cùng, *MFS-MC* cho ra kết quả là tập $\mathcal{FS}^M$.

4.3.4. Kỹ thuật khử ràng buộc loại C_{SupS}

Mặc dù phương pháp phân hoạch được sử dụng trong thuật toán *MFS-MC* cho phép loại bỏ sớm nhiều lớp tương đương hoặc tập con mà các chuỗi đại diện của chúng không thỏa ràng buộc, nhưng do đặc điểm của lớp ràng buộc C^M, ta có thể vẫn phải thường xuyên kiểm tra ràng buộc trong quá trình sinh ra các chuỗi phổ biến của các lớp còn lại. Vì vậy, hiệu quả của thuật toán *MFS-MC* có thể vẫn chưa thật cao cho một số dạng ràng buộc đặc biệt, chẳng hạn như loại ràng buộc C_{SupS} (xem *Ví dụ 4.6*). Để khắc phục hạn chế này, luận án đề xuất một kỹ thuật khử ràng buộc C_{SupS} (chi tiết của loại ràng buộc này được trình bày trong *Mục 4.1*) để tránh việc kiểm tra C_{SupS} trong suốt quá trình khai thác, qua đó góp phần nâng cao hơn nữa tính hiệu quả của *MFS-MC* khi áp dụng nó cho dạng ràng buộc này.

4.3.4.1. Phân hoạch tập lời giải $\mathcal{FS}^{\Gamma_0}$ dựa trên kỹ thuật khử ràng buộc

Trước hết, ta nhận thấy rằng, với phương pháp sinh chuỗi trong *Mục 4.3.2*, $\forall \gamma \in GenSet(\sigma)$, $\forall lp \in LP(\sigma,\gamma)$, thì chuỗi $\Delta(lp) \stackrel{\text{def}}{=} \sigma \ominus Ex(\gamma, lp)$ càng lớn khi γ càng nhỏ, khi đó số lượng các chuỗi con $\delta(lp)$ của $\Delta(lp)$ và các chuỗi tương ứng $\alpha = Ex(\gamma, lp) \oplus \delta(lp)$ có thể rất lớn. Ngoài ra, $\forall \sigma \in CS^{\Gamma_0}(ro)$, $\forall \gamma \in GenSet(\sigma)$, ta có $\Gamma_0 \sqsubseteq \sigma$ và $\gamma \sqsubseteq \sigma$. Bởi vậy, với bất kỳ $lp \in LP(\sigma,\gamma)$, $lp_0 \in LP(\sigma,\Gamma_0)$, ta định nghĩa tổng trực tiếp của hai chuỗi mở rộng $Ex(\Gamma_0, lp_0)$ và $Ex(\gamma, lp)$, ký hiệu là $sumex(\Gamma_0, \gamma, \sigma, lp_0, lp)$ hay ngắn gọn là $sm = sumex(lp_0, lp)$ như sau:

$$sumex(lp_0, lp) \stackrel{\text{def}}{=} Ex(\Gamma_0, lp_0) \oplus Ex(\gamma, lp).$$

Khi đó, sau khi rút gọn, $\Gamma_0 = reduce(Ex(\Gamma_0, lp_0)) = Ex(\Gamma_0, lp_0) \sqsubseteq sm \sqsubseteq \sigma$ và $\gamma = Ex(\gamma, lp) \sqsubseteq sm$. Ngoài ra, $\forall lpu \in LP(\sigma, sm)$, $Ex(\gamma, lp) \sqsubseteq Ex(sm, lpu) = sm$ (sau khi rút gọn), nên $\Delta u(lpu) \stackrel{\text{def}}{=} \sigma \ominus Ex(sm, lpu) \sqsubseteq \Delta(lp) \stackrel{\text{def}}{=} \sigma \ominus Ex(\gamma, lp)$. Khi đó, với bất kỳ $\delta u(lpu) \sqsubseteq$

$\Delta u(lpu)$ và $\alpha = Ex(sm, lpu) \oplus \delta u(lpu)$, ta luôn có $\alpha \sqsupseteq Ex(sm, lpu) = sm \sqsupseteq \Gamma_0$. Vì vậy, chuỗi α được sinh ra theo cách này không cần phải kiểm tra ràng buộc, vì nó luôn chứa Γ_0.

Đặt: $UE(\sigma, \gamma, \Gamma_0) \stackrel{\text{def}}{=} \{reduce(sumex(\Gamma_0, \gamma, \sigma, lp_0, lp)) \mid lp_0 \in LP(\sigma, \Gamma_0), lp \in LP(\sigma, \gamma)\}$ và

$$MUE^{\Gamma_0}(\sigma) \stackrel{\text{def}}{=} \{minimal(sm) \mid sm \in UE(\sigma, \gamma, \Gamma_0), \gamma \in GenSet(\sigma)\}.$$

Khi đó, $MUE^{\Gamma_0}(\sigma)$ có vai trò giống như $GenSet(\sigma)$. Ngoài ra, lực lượng của tập $MUE^{\Gamma_0}(\sigma)$ thường không lớn hơn tập *GenSet*(σ), do có thể tồn tại hai chuỗi sinh $\gamma_1, \gamma_2 \in GenSet(\sigma)$ sao cho hai chuỗi tương ứng $sm_1 \in UE(\sigma, \gamma_1, \Gamma_0)$, $sm_2 \in UE(\sigma, \gamma_2, \Gamma_0)$ thỏa điều kiện $sm_2 \sqsubset sm_1$. Khi đó, tất cả các chuỗi được sinh bởi $sm_1 \sqsubseteq \alpha \sqsubseteq \sigma$ sẽ trùng lặp với tất cả các chuỗi được sinh bởi $sm_2 \sqsubseteq \alpha \sqsubseteq \sigma$. Do đó, hàm *minimal* được sử dụng trong định nghĩa của $MUE^{\Gamma_0}(\sigma)$ nhằm loại bỏ các chuỗi như sm_1. Nhắc lại rằng, hàm *reduce* được sử dụng trong định nghĩa của UE nhằm loại bỏ các sự kiện rỗng trong chuỗi.

Dựa trên tập $MUE^{\Gamma_0}(\sigma)$, thay vì khai thác tập $\mathcal{FS}^{\Gamma_0}(\sigma, \gamma)$ (có kiểm tra ràng buộc C_{SupS} tại bước hậu xử lý), ta nên khai thác tập tương đương $\mathcal{FS}(\sigma, sm)$ (không cần kiểm tra loại ràng buộc Γ_0 vì các chuỗi trong tập $\mathcal{FS}(\sigma, sm)$ luôn chứa Γ_0) bằng cách thay tập $GenSet(\sigma)$ bởi tập $MUE^{\Gamma_0}(\sigma)$. Vì vậy, cách khai thác này được gọi là phương pháp *khử ràng buộc* (Γ_0 cho các tập con $\mathcal{FS}(\sigma_i, sm_j)$ ở mức hai). Khi đó, $\forall ro \in RO^{\Gamma_0}$, $\forall \sigma_i \in CS^{\Gamma_0}(ro)$, $\forall sm_j \in MUE^{\Gamma_0}(\sigma_i)$, luận án đưa ra các định nghĩa sau: $\mathcal{FS}^{\Gamma_0}(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \sqsubseteq \sigma_i \mid C_{SupS}(\alpha) \wedge \rho(\alpha) = \rho(\sigma_i)\}$, $\mathcal{FS}^{*\Gamma_0}(\sigma_i) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}^{\Gamma_0}(\sigma_i) \mid not(DCondC(\alpha, \sigma_i))\}$, $\mathcal{FS}(\sigma_i, sm_j) \stackrel{\text{def}}{=} \{\alpha \mid sm_j \sqsubseteq \alpha \sqsubseteq \sigma_i\}$ và $\mathcal{FS}^*(\sigma_i, sm_j) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{FS}(\sigma_i, sm_j) \mid not(DCondC(\alpha, \sigma_i)) \wedge not(DCondG(\alpha, \sigma_i, sm_j))\}$. Ta có biểu diễn của lớp $FS^{\Gamma_0}(ro)$ như sau:

$$\mathcal{FS}^{\Gamma_0}(ro) = \bigcup_{\sigma_i \in CS^{\Gamma_0}(ro)} \mathcal{FS}^{\Gamma_0}(\sigma_i) = \bigcup_{\sigma_i \in CS^{\Gamma_0}(ro)} \bigcup_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}(\sigma_i, sm_j),$$

trong đó, $\mathcal{FS}^{\Gamma_0}(\sigma_i) = \bigcup_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}(\sigma_i, sm_j)$.

Dựa trên các biểu diễn này, ta có *Định lý 4.5* dưới đây.

Định lý 4.5 (*Phân hoạch các tập $\mathcal{FS}^{\Gamma_0}(ro)$ và $\mathcal{FS}^{\Gamma_0}$*).

$\forall ro \in RO^{\Gamma_0}$, $\forall \sigma_i \in CS^{\Gamma_0}(ro)$, $\forall sm_j \in MUE^{\Gamma_0}(\sigma_i)$, ta có các biểu diễn sau:

$\mathcal{FS}^{\Gamma_0}(ro) = \sum_{\sigma_i \in CS^{\Gamma_0}(ro)} \mathcal{FS}^{*\Gamma_0}(\sigma_i)$ và $\mathcal{FS}^{*\Gamma_0}(\sigma_i) = \sum_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}^*(\sigma_i, sm_j)$.

Khi đó, ta có các phân hoạch mịn của $\mathcal{FS}^{\Gamma_0}(ro)$ và $\mathcal{FS}^{\Gamma_0}$ như sau:

$$\mathcal{FS}^{\Gamma_0}(ro) = \sum_{\sigma_i \in CS^{\Gamma_0}(ro)} \sum_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}^*(\sigma_i, sm_j) \text{ và}$$

$$\mathcal{FS}^{\Gamma_0} = \sum_{ro \in RO^{\Gamma_0}} \sum_{\sigma_i \in CS^{\Gamma_0}(ro)} \sum_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}^*(\sigma_i, sm_j).$$

4.3.4.2. Sinh đầy đủ và không trùng lặp các chuỗi trong $\mathcal{FS}(\sigma, sm)$ và $\mathcal{FS}^{\Gamma_0}$

Tương tự như trong *Mục 4.3.2*, biểu diễn của các chuỗi trong mỗi tập $\mathcal{FS}(\sigma, sm)$ xuất hiện trong định nghĩa của tập $\mathcal{FS}^*(\sigma, sm)$ là không tường minh. Vì vậy, luận án đề xuất một biểu diễn tường minh $\mathcal{FS}'(\sigma, sm)$ dưới đây cho tập $\mathcal{FS}(\sigma, sm)$. Với $\forall ro \in RO^{\Gamma_0}$, $\forall \sigma \in CS^{\Gamma_0}(ro)$, $\forall sm \in MUE^{\Gamma_0}(\sigma)$, $\forall lpu \in LP(\sigma, sm)$, $\Delta u(lpu) \stackrel{\text{def}}{=} \sigma \ominus Ex(sm, lpu)$, tập $\mathcal{FS}'(\sigma, sm)$ được định nghĩa như sau:

$$\mathcal{FS}'(\sigma, sm) \stackrel{\text{def}}{=} \{\alpha = Ex(sm, lpu) \oplus \delta u(lpu) \mid lpu \in LP(\sigma, sm) \wedge \delta u(lpu) \sqsubseteq \Delta u(lpu)\}.$$

Tuy nhiên, mặc dù các chuỗi trong tập $\mathcal{FS}'(\sigma, sm)$ được sinh ra một cách tường minh, nhiều chuỗi trong đó vẫn bị trùng lặp. Để tránh điều này, luận án tích hợp các điều kiện phát hiện sớm sự trùng lặp $DCond_{1-3}$ trong *Chương 3* vào trong một biểu diễn khác $\mathcal{FS}''(\sigma, sm)$ của $\mathcal{FS}'(\sigma, sm)$ như sau:

$$\mathcal{FS}''(\sigma, sm) \stackrel{\text{def}}{=} \{\alpha = Ex(sm, lpu) \oplus \delta u(lpu) \in \mathcal{FS}'(\sigma, sm) \mid not(DCond_{1-3}(\alpha, lpu))\}.$$

Dựa trên tập $\mathcal{FS}''(\sigma, sm)$, ta có *Định lý 4.6* sau đây.

Định lý 4.6 (*Biểu diễn tường minh và không trùng lặp các chuỗi trong* $\mathcal{FS}(\sigma, sm)$ *và* $\mathcal{FS}^{\Gamma_0}$).

a) ($\mathcal{FS}''(\sigma, sm)$, *biểu diễn tường minh, không trùng lặp các chuỗi trong* $\mathcal{FS}(\sigma, sm)$).

$\forall ro \in RO^{\Gamma_0}$, $\forall \sigma \in CS^{\Gamma_0}(ro)$, $\forall sm \in MUE^{\Gamma_0}(\sigma)$, khi đó: $\mathcal{FS}(\sigma, sm) = \mathcal{FS}'(\sigma, sm) = \mathcal{FS}''(\sigma, sm)$ và $\mathcal{FS}^*(\sigma, sm) = \{\alpha \in \mathcal{FS}'(\sigma, sm) \mid not(DCond_{1-3}(\alpha, lp)) \wedge not(DCondC(\alpha, \sigma)) \wedge not(DCondG(\alpha, \sigma, sm))\}$.

b) Tất cả các chuỗi phổ biến trong tập $\mathcal{FS}''(\sigma, sm)$ được sinh ra không trùng lặp. Do đó, tất cả các chuỗi trong tập $\mathcal{FS}^{\Gamma_0} = \sum_{ro \in RO^{\Gamma_0}} \sum_{\sigma_i \in CS^{\Gamma_0}(ro)} \sum_{sm_j \in MUE^{\Gamma_0}(\sigma_i)} \mathcal{FS}^*(\sigma_i, sm_j)$ cũng được sinh ra đầy đủ và không trùng lặp.

Ưu điểm của phương pháp phân hoạch dựa trên kỹ thuật khử ràng buộc được minh họa cụ thể trong *Ví dụ 4.7*. Để giải quyết hiệu quả bài toán khai thác tập $\mathcal{FS}^{\Gamma_0}$ với ràng buộc C_{SupS} thuộc lớp C^M, luận án đề xuất thuật toán *MFS-SupSC* dựa trên các *Định lý 4.5 – 4.6*. Mã giả của *MFS-SupSC* tương tự như thuật toán tổng quát *MFS-MC* nhưng trong đó tập $GenSet(\sigma_i)$ được thay thế bởi tập $MUE^{\Gamma_0}(\sigma_i)$. Ngoài ra, lưu ý thêm rằng $\mathcal{FCGS}^{\Gamma_0}(ro) \stackrel{\text{def}}{=} \{(\sigma, MUE^{\Gamma_0}(\sigma)) \mid (\sigma, GenSet(\sigma)) \in \mathcal{FCGS}^{*\Gamma_0}(ro)\}$ và $CS^{\Gamma_0}(ro) = \{\sigma \mid (\sigma, MUE^{\Gamma_0}(\sigma)) \in \mathcal{FCGS}^{\Gamma_0}(ro)\}$, trong đó $MUE^{\Gamma_0}(\sigma) \stackrel{\text{def}}{=} \{minimal(sm) \mid sm \in UE(\sigma, \gamma, \Gamma_0), \gamma \in GenSet(\sigma)\}$ và $UE(\sigma, \gamma, \Gamma_0) \stackrel{\text{def}}{=} \{reduce(sumex(\Gamma_0, \gamma, \sigma, lp_0, lp)) \mid lp_0 \in LP(\sigma, \Gamma_0), lp \in LP(\sigma, \gamma)\}$.

4.4. Đánh giá thực nghiệm

Hiệu quả của các thuật toán được đề xuất *MFS-SubSC*, *MFS-IC* (hai phiên bản cụ thể của thuật toán tổng quát *MFS-AC* ứng với hai ràng buộc C_{SubS} và C_I), và *MFS-SupSC* (phiên bản cụ thể của *MFS-MC* ứng với ràng buộc C_{SupS}) được so sánh với các thuật toán tiêu biểu trước đây *CM-SPADE*, *CM-SPAM* [17] và *PrefixSpan* [63] thông qua việc tích hợp các ràng buộc C_{SubS}, C_I và C_{SubS} vào trong quá trình khai thác của chúng. Hai thực nghiệm đã được thực hiện là: *Thực nghiệm 1* (thay đổi giá trị của tham số $minsupp$) và *Thực nghiệm 2* (thay đổi số lượng ràng buộc được sử dụng). Những kết quả trong cả hai thực nghiệm đã cho thấy rằng, các thuật toán *MFS-IC*, *MFS-SubSC* và *MFS-SupSC* hiệu quả hơn nhiều về thời gian khai thác và bộ nhớ sử dụng so với các phiên bản được sửa đổi của *CM-SPAM*, *CM-SPADE* và *PrefixSpan*, đặc biệt trên các *SDB* lớn với các giá trị $minsupp$ bé, hoặc khi số lượng ràng buộc lớn. Ngoài ra, các thực nghiệm cũng chỉ ra rằng mô hình được đề xuất trong luận án không chỉ có hiệu quả cao trên lớp ràng buộc C^A, nhưng đồng thời trên lớp ràng buộc phức tạp C^M. Chẳng hạn, trong *Thực nghiệm 1*, trên *SDB BMS* với $minsupp$ = 0.057%, hai thuật toán *MFS-IC* and *MFS-SubSC* (khai thác chuỗi phổ biến với các loại ràng buộc thuộc lớp C^A) chạy nhanh hơn khoảng từ 6 đến 71 lần so với các thuật toán trước đây. Trong khi đó, thuật toán *MFS-SupSC* (khai thác chuỗi phổ biến với loại ràng buộc thuộc lớp C^M) chạy nhanh hơn các thuật toán đã có trước đây khoảng từ 109 đến 984 lần (do kỹ thuật khử ràng buộc được sử dụng thêm). Điều này cho thấy rằng, những kết quả lý thuyết mà luận án đã đề xuất trong chương này là thật sự có ý nghĩa và hiệu quả.

Chương 5 - Kết luận

Luận án đã đề xuất một mô hình mới để khai thác nhanh tập $\mathcal{FS}^C$ từ các tập chuỗi đóng phổ biến $\mathcal{FCS}$ và chuỗi sinh phổ biến $\mathcal{FGS}$, thay vì khai thác nó trực tiếp từ SDB như các phương pháp truyền thống. Để triển khai hiệu quả mô hình được đề xuất, luận án cần giải quyết tốt ba bài toán chính sau đây.

Bài toán đầu tiên ($\mathcal{BT}_1$ trong *Chương 2*) luận án cần giải quyết là khai thác hiệu quả hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ từ SDB. Chúng là các biểu diễn súc tích và không mất thông tin của tập $\mathcal{FS}$ bao gồm tất cả các chuỗi phổ biến có trong SDB, nghĩa là hai tập này mặc dù có lực lượng thường bé hơn rất nhiều so với $\mathcal{FS}$ nhưng chúng có thể được kết hợp với nhau để sinh ra tập $\mathcal{FS}$ mà không cần đọc lại SDB. Ngoài ra, chúng đồng thời có ý nghĩa quan trọng trong nhiều ứng dụng thực tế như được trình bày trong *Chương 2*. Cho đến nay đã có nhiều thuật toán được đề xuất để khai thác $\mathcal{FCS}$ và $\mathcal{FGS}$. Tuy nhiên, sau khi nghiên cứu và phân tích, luận án nhận thấy rằng hiệu quả khai thác của chúng vẫn còn chưa cao do còn tồn tại nhiều hạn chế. Vì vậy, luận án đã tập trung nghiên cứu bài toán $\mathcal{BT}_1$ và đã có một số đóng góp chính như sau:

- Luận án đã chỉ ra sự không chính xác trong trường hợp tổng quát của một số kết quả lý thuyết đã được đưa ra trong [91] năm 2003 và đã được trích dẫn và sử dụng nhiều trong các thuật toán sau đó để khai thác các chuỗi đóng và sinh phổ biến.
- Một số đo SE mới cùng với các kết quả lý thuyết chính xác và tổng quát hơn những kết quả lý thuyết trong [91] đã được đề xuất trong luận án, và chúng là cơ sở lý thuyết tin cậy để phát triển các chiến lược tỉa địa phương nhằm loại bỏ sớm các chuỗi ứng viên không là đóng và/hoặc sinh trên cây tìm kiếm tiền tố.
- Luận án đã đề xuất bốn thuật toán mới được gọi là FGenSM (khai thác $\mathcal{FGS}$), FCloSM (khai thác $\mathcal{FGS}$), FGenCloSM (khai thác tuần tự cả hai $\mathcal{FCS}$ và $\mathcal{FGS}$) và Par-GenCloSM (khai thác song song cả hai $\mathcal{FCS}$ và $\mathcal{FGS}$). Sự chính xác và hiệu quả của các thuật toán này đã đóng góp nhiều vào việc giải quyết hiệu quả bài toán khai thác các chuỗi phổ biến với ràng buộc trong *Chương 4* ($\mathcal{BT}_3$).

Những đóng góp trên của luận án cho bài toán $\mathcal{BT}_1$ đã được công bố trong [8,35,34].

Như được đề cập trong $\mathcal{BT}_1$ ở trên, hai tập $\mathcal{FCS}$ và $\mathcal{FGS}$ là những biểu diễn súc tích và không mất thông tin của $\mathcal{FS}$, và về mặt lý thuyết chúng có thể được kết hợp với nhau để sinh ra đầy đủ các chuỗi trong tập $\mathcal{FS}$. Tuy nhiên, cho đến nay chưa có nghiên cứu nào đề xuất một phương pháp tường minh để sinh ra nhanh $\mathcal{FS}$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$. Vì vậy, trong bài toán thứ hai ($\mathcal{BT}_2$ trong *Chương 3*), công việc của luận án là phát triển một phương pháp để sinh nhanh tập $\mathcal{FS}$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$, thay vì khai thác $\mathcal{FS}$ trực tiếp từ SDB như các phương pháp truyền thống. Phương pháp này sẽ là cơ sở lý thuyết quan trọng để luận án phát triển một mô hình mới giải quyết hiệu quả bài toán khai thác tập $\mathcal{FS}^{\mathcal{C}}$ trong *Chương 4*.

Để giải quyết bài toán $\mathcal{BT}_2$, luận án đã nghiên cứu và đưa ra *cấu trúc* của các chuỗi phổ biến trong tập $\mathcal{FS}$ dựa trên phương pháp phân hoạch. Cụm từ cấu trúc ở đây cũng như trong tên của luận án có nghĩa là mối quan hệ tường minh giữa các chuỗi trong $\mathcal{FS}$ và các chuỗi (đại diện của chúng) trong $\mathcal{FCS}$ và $\mathcal{FGS}$. Dựa trên mối quan hệ này, ta có thể biểu diễn một chuỗi α bất kỳ trong $\mathcal{FS}$ thông qua một chuỗi đóng σ trong $\mathcal{FCS}$ và một chuỗi sinh γ trong $\mathcal{FGS}$. Nói cách khác, α có thể được sinh ra từ cặp (σ, γ). Khó khăn lớn nhất khi giải bài toán $\mathcal{BT}_2$ là mỗi chuỗi phổ biến α có thể được sinh ra từ nhiều cặp (σ, γ) khác nhau và α có thể xuất hiện (chứa trong) σ tại nhiều vị trí khác nhau. Điều này có thể dẫn đến một số lượng rất lớn các chuỗi trùng lặp được sinh ra trong tập $\mathcal{FS}$, làm giảm hiệu quả về thời gian và bộ nhớ của quá trình khai thác. Để khắc phục, luận án đã đưa ra các điều kiện trùng lặp nhằm phát hiện và loại bỏ sớm tất cả các chuỗi trùng lặp mà không cần sinh ra chúng. Kết hợp tất cả các kết quả lý thuyết được đưa ra, luận án đã đề xuất một thuật toán mới tên là FS-Miner để sinh ra nhanh, đầy đủ và không trùng lặp tập $\mathcal{FS}$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$.

Những đóng góp trên cho bài toán $\mathcal{BT}_2$ đã được chấp nhận xuất bản trong một kỷ yếu hội nghị quốc tế uy tín [36].

Trong bài toán thứ ba ($\mathcal{BT}_3$ trong *Chương 4*), luận án tập trung vào cách tiếp cận khai thác tập $\mathcal{FS}^C$ từ $\mathcal{FCS}$ và $\mathcal{FGS}$, thay vì khai thác $\mathcal{FS}^C$ trực tiếp từ SDB như các tiếp cận truyền thống, trong đó C thuộc vào lớp ràng buộc C^A (các ràng buộc có tính đơn điệu giảm) hoặc C^M (các ràng buộc có tính đơn điệu tăng). Đây là hai lớp ràng buộc thường gặp trong thực tế. Tương ứng với hai lớp ràng buộc C^A và C^M là hai bài toán con tổng quát $\mathcal{BT}_{3.1}$ và $\mathcal{BT}_{3.2}$ cần được giải quyết trong chương này.

Tiếp cận chung được sử dụng trong luận án để giải quyết hai bài toán con $\mathcal{BT}_{3.1}$ và $\mathcal{BT}_{3.2}$ là dựa trên phương pháp phân hoạch. Với một quan hệ tương đương dựa trên toán tử ρ, tập lời giải $\mathcal{FS}^C$ được phân hoạch thành các lớp tương đương, mỗi lớp bao gồm các chuỗi xuất hiện trong cùng một tập con của các chuỗi đầu vào trong SDB (cùng ρ) và được đại diện bởi một số chuỗi đóng σ trong $\mathcal{FCS}$ và chuỗi sinh γ trong $\mathcal{FGS}$. Nói cách khác, mỗi lớp tương đương có thể được sinh ra từ các tập con được đại diện bởi các cặp (σ, γ) khác nhau. Do tính chất đơn điệu giảm hoặc đơn điệu tăng của các lớp ràng buộc C^A hoặc C^M, tính tối tiểu và tối đại của các phần tử đại diện γ và σ, ta có thể rút gọn nhanh không gian tìm kiếm bằng việc loại bỏ sớm nhiều tập con trong mỗi lớp tương đương thông qua việc kiểm tra ràng buộc chỉ trên các chuỗi đại diện (σ, γ) của lớp mà không cần sinh ra mọi chuỗi trong mỗi lớp. Sử dụng tính chất này và mối quan hệ tường minh được đề xuất trong *Chương 3*, luận án đã phát triển các phương pháp mới để giải quyết hiệu quả hai bài toán con tổng quát $\mathcal{BT}_{3.1}$ và $\mathcal{BT}_{3.2}$. Ngoài ra, luận án cũng đã đề xuất một kỹ thuật khử ràng buộc nhằm loại bỏ việc kiểm tra một ràng buộc phức tạp thuộc lớp C^M trong suốt quá trình khai thác, qua đó rút ngắn được đáng kể thời gian khai thác. Lưu ý là, các phương pháp trước đây thường có hiệu quả thấp với lớp ràng buộc C^M do cách tiếp cận truyền thống khó có thể được áp dụng hiệu quả để rút gọn nhanh không gian tìm kiếm với C^M.

Những điểm khác biệt và ưu điểm nổi trội của các phương pháp được đề xuất trong *Chương 4* so với các phương pháp truyền thống là:

- Tập $\mathcal{FS}^C$ được khai thác nhanh từ hai tập các chuỗi đóng và chuỗi sinh phổ biến (thường có kích thước khá bé), trong khi các phương pháp trước đây khai thác $\mathcal{FS}^C$ trực tiếp từ cơ sở dữ liệu (thường có kích thước lớn).
- Hai thuật toán *MFS-AC* và *MFS-MC* cho phép khai thác hiệu quả các chuỗi phổ biến với nhiều loại ràng buộc khác nhau thuộc hai lớp ràng buộc thông dụng C^A và C^M, trong khi đa số các thuật toán trước đây thường được thiết kế cho một loại ràng buộc cụ thể và chúng thường gặp nhiều khó khăn trong việc rút gọn không gian tìm kiếm khi đối mặt với lớp ràng buộc C^M.
- Việc kiểm tra ràng buộc được thực hiện trên một số lượng bé các phần tử đặc biệt (các chuỗi đóng và chuỗi sinh phổ biến đại diện cho mỗi lớp tương đương) nhưng có thể loại bỏ một số lượng rất lớn các chuỗi không thỏa ràng buộc mà không cần sinh ra chúng.
- Các phương pháp được đề xuất cho phép thiết kế dễ dàng hơn các thuật toán song song để khai thác hiệu quả hơn nữa các chuỗi phổ biến có hoặc không có ràng buộc trên những cơ sở dữ liệu lớn.
- Tính hiệu quả của các phương pháp được đưa ra ít chịu ảnh hưởng khi ràng buộc thường xuyên bị thay đổi bởi nhiều người dùng.

Một số đóng góp chính của luận án cho bài toán $\mathcal{BT}_3$ đã được xuất bản trong [32].