DOI: https://doi.org/10.62985/j.huit_ojs.vol25.no2.181

NÂNG CAO HIỆU QUẢ KHAI THÁC N TẬP HỮU ÍCH CAO NHẤT TRONG CƠ SỞ DỮ LIỆU GIAO DỊCH

Vũ Văn Vinh, Mạnh Thiên Lý, Dương Thị Mộng Thùy,
Nguyễn Thị Thanh Thủy, Nguyễn Văn Lễ, Lâm Thị Họa Mi*
Trường Đại học Công Thương Thành phố Hồ Chí Minh
*Email: *milth@huit.edu.vn*

Ngày nhận bài: 20/3/2023; Ngày chấp nhận đăng: 22/5/2023

TÓM TẮT

Bài toán khai thác các tập mặt hàng có lợi nhuận cao đã có nhiều ứng dụng trong thực tế. Tuy nhiên trong quá trình khai thác cần xác định trước giá trị ngưỡng lợi nhuận tối thiểu dẫn đến khó khăn cho người dùng. Để giải quyết vấn đề này, nhiều nghiên cứu đã được đề xuất nhằm thay thế việc xác định ngưỡng lợi nhuận tối thiểu bằng việc đưa ra một số nguyên dương N trong các bài toán về khai thác N tập hữu ích cao nhất (top-N HUI). Trong nghiên cứu này, nhóm tác giả đề xuất thuật toán *topNEFIM* để khai thác hiệu quả top-N HUI trong cơ sở dữ liệu giao dịch cùng với chiến lược tăng ngưỡng RTU để xác định tự động giá trị ngưỡng ban đầu. Thuật toán cũng sử dụng một cấu trúc hàng đợi ưu tiên toàn cục (*priorityQueue*) để tối ưu quá trình tăng ngưỡng. Ngoài ra, nhóm tác giả còn đề xuất cấu trúc dữ liệu *ExtentionP_set* để hạn chế việc duyệt cơ sở dữ liệu nhiều lần trong quá trình khai thác top-N HUI. Kết quả thực nghiệm chỉ ra rằng thuật toán do nhóm tác giả đề xuất có thời gian thực thi tốt hơn trên cả cơ sở dữ liệu dày và thưa khi so sánh với hai thuật toán TKEH và THUI.

Từ khóa: Tập hữu ích cao, N tập hữu ích cao nhất, chiến lược tăng ngưỡng, khai thác dữ liệu.

1. MỞ ĐẦU

Bài toán khai thác tập hữu ích cao (HUIM – High Utility Itemset Mining) đã có nhiều ứng dụng trong thực tế và ngày càng trở nên phổ biến trong những năm gần đây. Tuy nhiên, việc khai thác tập hữu ích cao (HUI – High Utility Itemset) cần xác định trước giá trị ngưỡng độ hữu ích tối thiểu và việc lựa chọn giá trị này một cách phù hợp đôi khi là vấn đề gây khó khăn cho người dùng. Nếu giá trị ngưỡng được xác lập quá thấp sẽ dẫn đến thời gian thực thi cao, không gian lưu trữ lớn và số lượng tập hữu ích cao khai thác được cũng có thể rất lớn. Kết quả này sẽ rất khó để lựa chọn ra các tập hữu ích cao phù hợp trong việc xây dựng hay thay đổi chiến lược kinh doanh. Ngược lại, nếu chọn giá trị ngưỡng quá cao, có thể sẽ không tìm được bất kỳ tập hữu ích cao nào từ cơ sở dữ liệu. Từ đó, bài toán khai thác top-N HUI được đề xuất để giải quyết hạn chế này. Thay vì dựa vào ngưỡng độ hữu ích tối thiểu, bài toán khai thác top-N HUI chỉ quan tâm đến số lượng tập có lợi nhuận cao nhất mà người dùng mong muốn nhận được (giá trị N).

Trong bài báo này, nhóm tác giả nghiên cứu các phương pháp, chiến lược tỉa đã được đề xuất để tối ưu hóa việc khai thác HUI; các kỹ thuật, chiến lược tăng ngưỡng nhằm nâng cao hiệu suất trong bài toán khai thác top-N HUI. Những đóng góp chính của bài báo gồm: (1) Đề xuất chiến lược tăng ngưỡng RTU (Real Transaction Utility) dựa vào độ hữu ích của các giao dịch trong cơ sở dữ liệu; (2) Sử dụng cấu trúc hàng đợi ưu tiên toàn cục (priorityQueue) lưu trữ các giá trị độ hữu ích tìm được khi áp dụng các chiến lược tăng ngưỡng để có thể sử dụng lại các giá trị này trong các giai đoạn tăng ngưỡng sau đó; (3) Cải tiến cấu trúc lưu trữ P_set thành ExtentionP_set để loại bỏ việc duyệt cơ sở dữ liệu nhiều lần, giảm đáng kể thời gian thực thi trong quá trình khai thác top-N HUI; (4) Đề xuất thuật toán *topNEFIM* giải quyết hiệu quả bài toán top-N HUI về mặt thời gian trên cả cơ sở dữ liệu dày và thưa. Nghiên cứu đã tiến hành thực nghiệm và so sánh *topNEFIM* với hai thuật toán *TKEH* [1] và *THUI* [2] cho thấy thuật toán đề xuất có thời gian thực thi tốt hơn trên các cơ sở dữ liệu dày và thưa.

2. CÁC CÔNG TRÌNH LIÊN QUAN

2.1. Khai thác tập hữu ích cao

Hiện nay, nhiều nghiên cứu đã được thực hiện để khai thác các tập hữu ích cao trên cơ sở dữ liệu giao dịch và mang lại nhiều giá trị trong các ứng dụng thực tế. Năm 2004, Yao và cộng sự đề xuất thuật toán khai thác HUI đầu tiên là *UMining* [3], thuật toán này hoạt động dựa vào việc tính các cận trên của các tập mục. Sau đó, nhóm tác giả này đã đề xuất thuật toán *Umining – H* trên cơ sở cải tiến thuật toán trên bằng cách sử dụng kỹ thuật Heuristic để tỉa các tập ứng viên hiệu quả hơn. Năm 2005, Liu và cộng sự trình bày khái niệm trọng số hữu ích giao dịch – TWU (Transaction Weighted Utility) [4] – là một ngưỡng cận trên thỏa tính chất bao đóng giảm giúp thu gọn không gian tìm kiếm nhằm tìm nhanh hơn các HUI có trong cơ sở dữ liệu; đồng thời đề xuất thuật toán *Two-Phase* tích hợp TWU vào thuật toán Apriori [5]. Thuật toán *Two-Phase* thực hiện khai thác các HUI ở hai pha. Pha 1: tính TWU của các tập mục và so sánh với ngưỡng độ hữu ích tối thiểu δ, các tập mục có TWU không nhỏ hơn ngưỡng δ được xem là các tập ứng viên hữu ích cao. Pha 2: quét lại cơ sở dữ liệu nhiều lần để tính chính xác giá trị hữu ích của các tập ứng viên này và trả về kết quả là các tập mục có độ hữu ích không nhỏ hơn ngưỡng δ. Tiếp theo đó, nhiều nhóm tác giả đã lần lượt giới thiệu các thuật toán nhằm nâng cao tính hiệu quả của quá trình khai thác HUI, chẳng hạn như: *TWU-Mining* [6], *DTWU-Mining* [7] và *IHUP* [8].

Năm 2012, Liu và cộng sự đề xuất thuật toán một pha có tên là *HUI-Miner* [9] với cấu trúc dữ liệu Utility-List đã làm giảm đáng kể thời gian thực thi và bộ nhớ sử dụng. Năm 2014, Fournier-Viger và cộng sự đề xuất thuật toán *FHM* [10] với cấu trúc EUCS. Thuật toán *EFIM* [11] được Zida và cộng sự đề xuất vào năm 2017. Nhóm tác giả đã đề xuất 03 kỹ thuật gồm: (i) Phép chiếu trên cơ sở dữ liệu (HDP – High-utility Database Projection) giúp giảm kích thước cơ sở dữ liệu có chứa các giao dịch có độ dài lớn. (ii) Ghép giao dịch hữu ích cao (HTM) để thu nhỏ không gian tìm kiếm bằng cách kết hợp các giao dịch tương tự nhau trong cơ sở dữ liệu chiếu. (iii) Giới thiệu hai ngưỡng cận trên mới trên độ đo độ hữu ích của các tập mục là độ hữu ích trên cây con (su – sub-tree utility*)* và độ hữu ích cục bộ (lu – local utility*)*, hai ngưỡng chặt chẽ hơn so với TWU trong việc giảm không gian tìm kiếm. Cùng khoảng thời gian đó, Krishnamoorthy đã kết hợp các kỹ thuật cắt tỉa không gian tìm kiếm đã được giới thiệu trước đó như TWU-Prune [4], LA-Prune [12], EUCS-Prune [10], U-Prune [12] và một kỹ thuật mới do ông đề xuất là C-Prune [12] để phát triển một thuật toán có tên gọi là *HMiner* cho phép khai thác HUI hiệu quả hơn. Sau đó, Duong và cộng sự đề xuất một thuật toán mở rộng dựa trên *FHM* có tên gọi là *ULBMiner* [13] vào năm 2018. Thuật toán này có khả năng tái sử dụng lại bộ nhớ thông qua cấu trúc dữ liệu bộ đệm Utility-List, nhờ đó giảm bộ nhớ cần thiết để thực hiện thuật toán. Các hướng nghiên cứu mở rộng khác về HUIM cũng được quan tâm và phát triển, chẳng hạn HUIM trên cơ sở dữ liệu giao dịch mà các hạng mục có lợi ích động hay lợi ích âm.

2.2. Khai thác N tập hữu ích cao nhất

Bài toán khai thác top-N HUI giúp cho người dùng giải quyết vấn đề khó khăn trong lựa chọn ngưỡng độ hữu ích tối thiểu phù hợp khi khai thác HUI. *TKU* [14] là thuật toán đầu tiên được giới thiệu để khai thác top-N HUI. *TKU* thuộc lớp bài toán hai pha được phát triển từ *UP-Growth* [15]: pha đầu, *TKU* thực hiện tìm các ứng viên tiềm năng; pha thứ hai, *TKU* sàng lọc lại một lần nữa để xác định chính xác các HUI nằm trong top-N HUI cần tìm. Về vấn đề tăng ngưỡng, *TKU* đã sử dụng các chiến lược như: PE (Pre-evaluation Step) (trong pha đầu) và MC (MUIs of Candidates), MD (MIU values of Descendents), NU (Node Utilities) và SE (Sorting and calculating Exact utility of candidates) (trong pha thứ hai). Ngoài ra, *TKU* cũng sử dụng 04 chiến lược tỉa không gian tìm kiếm dựa trên cây UP-Tree là: DGU (Discarding Global Unpromising items), DGN (Discarding Global Node utilities), DLN (Decreasing Local Node utilities) và DLU (Discarding Local Unpromising items). Cũng trong năm 2015, dựa trên cấu trúc cây UP-Tree, Ryang và Yun giới thiệu thuật toán *REPT* [16]. Nhóm tác giả này đã đề xuất 03 chiến lược tăng ngưỡng độ hữu ích tối thiểu ngay trong pha đầu để tính toán chính xác độ hữu ích của các tập mục chứa một mục (1-itemsets) và hai mục (2-itemsets) là: RIU (Real Item Utilities), PUD (Pre-evaluation with Utility Descending order) và RSD (Raising the threshold with items in Support Descending order). Trong pha thứ hai, một chiến lược tăng ngưỡng và tính chính xác độ hữu ích của các ứng viên SEP (Sorting candidates and raising the threshold by Exact and Pre-calculated utilities of candidates) được đề xuất làm tăng hiệu quả của *REPT* so với *TKU*. Tuy nhiên, các thuật toán hai pha thường tốn nhiều thời gian và lãng phí không gian lưu trữ trong quá trình thực thi do không kết hợp đồng thời được việc sinh ứng viên và tính chính xác độ hữu ích của chúng.

Để giải quyết các vấn đề này, nhiều thuật toán một pha được đề xuất cho bài toán top-N HUI hiệu quả hơn. Năm 2016, thuật toán *TKO* đã được đề xuất để giải quyết bài toán top-N chỉ với một pha bằng cách sử dụng cấu trúc lưu trữ Utility-List. *TKO* cũng áp dụng nhiều chiến lược tăng ngưỡng như: RUC (Raising threshold by Utility of Candidates), RUZ (Reducing estimated utility values by using Z-elements) và EPB (Exploring the most Promising Branches first). Kết quả thực nghiệm cho thấy, *TKO* hiệu quả hơn so với *TKU*. Sau đó, Duong và cộng sự đề xuất thuật toán *kHMC* [17] cũng chỉ sử dụng một pha để khai thác top-N HUI. Thuật toán đã giới thiệu hai chiến lược: chiến lược tăng ngưỡng EUCPT (Estimated Utility Co-occurrence Pruning Strategy with Threshold) dựa vào cấu trúc EUCST và chiến lược tỉa phần mở rộng bắc cầu TEP (Transitive Extension Pruning) để giảm không gian tìm kiếm; đồng thời tái sử dụng các chiến lược tăng ngưỡng như: RIU (Real Item Utilities), COV (COVerage with utility descending order) và CUD (Co-occurrence with Utility Descending order). Vào năm 2018, Liu và cộng sự đã đưa ra thuật toán *TONUP* (TOp-N High Utility Pattern mining) [18] - được phát triển dựa trên thuật toán *d2HUP* bằng cách cải tiến cấu trúc CAUL thành iCAUL. Thuật toán *TONUP* đã áp dụng các phương pháp tăng ngưỡng và sử dụng 05 chiến lược để duy trì danh sách các mẫu thu gọn, tính toán độ hữu ích và ước lượng một cách gần đúng cận trên nhằm giảm thiểu không gian tìm kiếm. Một thuật toán khác về top-N HUI cũng được đề xuất bởi Kuldeep Singh và cộng sự là *TKEH* [1], được phát triển trên nền tảng kỹ thuật chiếu cơ sở dữ liệu. Thuật toán này tận dụng tất cả tính ưu việt của phép chiếu cơ sở dữ liệu cũng như các chiến lược tỉa đã được giới thiệu trong thuật toán *EFIM* [11]. Vì vậy, *TKEH* có hiệu suất thực hiện rất tốt trên cả cơ sở dữ liệu dày và thưa. Năm 2019, thuật toán *THUI* được đề xuất bởi S. Krishnamoorthy [2] sử dụng hai cấu trúc để lưu trữ là Utility-List và ma trận LIU (Leaf Itemset Utility). LIU chính là giá trị độ hữu ích của một chuỗi liên tục các mục đã được sắp thứ tự (contiguous sequence) tồn tại trong cơ sở dữ liệu. *THUI* cũng đề xuất thêm các chiến lược tăng ngưỡng là LIU_E (LIU-Exact utilities) và LIU_LB (A novel utility lower bound estimation method) dựa trên giá trị LIU của các tập hợp đã được tính toán và lưu trữ trong ma trận LIU.

Để giải quyết những hạn chế của các thuật toán gần đây, bài báo này trình bày một thuật toán mới tên là *topNEFIM*.

3. ĐỊNH NGHĨA VÀ KÝ HIỆU

Phần này giới thiệu về các định nghĩa và các ký hiệu để giải quyết bài toán khai thác tập hữu ích cao và N tập hữu ích cao nhất trong cơ sở dữ liệu giao dịch.

Định nghĩa 1. (Cơ sở dữ liệu giao dịch) [9]. Tập mục $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ là một tập hữu hạn các hạng mục. Mỗi hạng mục $i \in \mathcal{I}$ được kết hợp với một giá trị dương $p(i)$, gọi là giá trị hữu ích ngoại của i. Một tập mục $X = \{i_1, i_2, \cdots, i_k\}$là một tập hữu hạn các hạng mục sao cho $X \subseteq \mathcal{I}$. Một cơ sở dữ liệu giao dịch $\mathcal{D} = \{T_1, T_2, \cdots, T_n\}$ gồm một tập các giao dịch sao cho với mỗi giao dịch T_c có một định danh TID và mỗi hạng mục i xuất hiện trong T_c được gắn với một giá trị dương $q(i, T_c)$ gọi là giá trị hữu ích nội của i.

Định nghĩa 2. (Độ hữu ích của một hạng mục) [9]. Độ hữu ích của hạng mục i trong giao dịch T_c được ký hiệu là $u(i, T_c)$ và $u(i, T_c) = p(i) \times q(i, T_c)$. Giá trị này biểu diễn lợi nhuận thu được khi bán hạng mục i trong giao dịch T_c.

Định nghĩa 3. (Độ hữu ích của một tập mục) [9]. Độ hữu ích của một tập mục X trong giao dịch T_c, ký hiệu $u(X, T_c)$ và được xác định như sau: $u(X, T_c) = \sum_{i \in X \wedge X \subseteq T_c} u(i, T_c)$.

Cho $g(X)$ là tập các giao dịch có chứa tập X. Khi đó, độ hữu ích của một tập mục X trong cơ sở dữ liệu $\mathcal{D}$, ký hiệu $u(X)$ và được định nghĩa là: $u(X) = \sum_{T_c \in g(X)} u(X, T_c)$.

Định nghĩa 4. [3] Độ hữu ích của một giao dịch T_c là tổng độ hữu ích của các hạng mục có trong T_c và được xác định như sau: $tu(T_c) = \sum_{i \in T_c} u(i, T_c)$. Nói cách khác, độ hữu ích của một giao dịch T_c chính là lợi nhuận phát sinh bởi giao dịch đó trong $\mathcal{D}$.

Bảng 1. Cơ sở dữ liệu giao dịch $\mathcal{D}$

TID	Tập mục	Giá trị hữu ích nội	Độ hữu ích trên từng hạng mục	Độ hữu ích của giao dịch
1	a, b, d, e	1, 2, 2, 3	4, 6, 4, 3	17
2	b, c, e, f	1, 5, 1, 2	3, 5, 1, 6	15
3	b, c, d, e	2, 1, 6, 2	6, 1, 12, 2	21
4	c, d, e	2, 2, 3	2, 4, 3	9
5	a, f	1, 1	4, 3	7
6	a, b, c, d, e	1, 1, 4, 4, 1	4, 3, 4, 8, 1	20
7	b, c, e, f	3, 2, 2, 3	9, 2, 2, 9	22

Bảng 2. Bảng giá trị hữu ích ngoại (giá trị lợi nhuận) của các hạng mục

Hạng mục	a	b	c	d	e	f
Giá trị lợi ích ngoại (Lợi nhuận)	4	3	1	2	1	3

Định nghĩa 5. [3] Một tập mục X được gọi là tập hữu ích cao (HUI) nếu độ hữu ích của X không nhỏ hơn ngưỡng độ hữu ích tối thiểu (δ) cho trước. Tức là, $u(X) \geq \delta$.

Ví dụ: Cho cơ sở dữ liệu cho trong bảng 1, 2: độ hữu ích của hạng mục a trong giao dịch T_1 là $u(a, T_1) = p(a) \times q(a, T_a) = 4 * 1 = 4$, độ hữu ích của tập mục $X = \{bc\}$ trong giao dịch T_2 là $u(bc, T_2) = u(b, T_2) + u(c, T_2) = 3 * 1 + 1 * 5 = 8$. Độ hữu ích của một tập mục $X = \{bc\}$ trong cơ sở dữ liệu là $u(bc) = u(bc, T_2) + u(bc, T_3) + u(bc, T_6) + u(bc, T_7) = 33$. Độ hữu ích của giao dịch T_2 là tổng độ hữu ích của các hạng mục có trong T_2 và được xác định như sau: $tu(T_2) = u(b, T_2) + u(c, T_2) + u(e, T_2) + u(f, T_2) = 15$. Với $\delta = 30$, $u(bc) = 33 > \delta$ nên X = {bc} là tập hữu ích cao.

Phát biểu bài toán khai thác N tập hữu ích cao nhất. Khai thác N tập hữu ích cao nhất (top-N HUI) là bài toán tìm N tập mục có độ hữu ích cao nhất trong cơ sở dữ liệu giao dịch $\mathcal{D}$.

Định nghĩa 6. [9] Cho tập mục X, giá trị trọng số hữu ích giao dịch của X được định nghĩa là tổng độ hữu ích của các giao dịch có chứa X, ký hiệu là $twu(X)$ và được tính theo công thức:

$$twu(X) = \sum_{T_c \in \mathcal{D} \wedge X \subseteq T_c} tu(T_c)$$

Tính chất 1. [9] Nếu $twu(X) < \delta$, thì X và tất cả các tập cha của X không phải là HUI.

Định nghĩa 7. [9] (Độ hữu ích của các phần tử phía sau). Cho $\succ$ là một thứ tự toàn phần các hạng mục từ $\mathcal{I}$ (ví dụ như thứ tự từ điển), X là một tập mục. Độ hữu ích của các phần tử phía sau của X trong một giao dịch T_c được định nghĩa:

$$ru(X, T_c) = \sum_{i \in T_c \wedge i > x \forall x \in X} u(i, T_c)$$

Khi đó, độ hữu ích của các phần tử phía sau X trong cơ sở dữ liệu $\mathcal{D}$ được định nghĩa:

$$ru(X) = \sum_{X \subseteq T_c \wedge T_c \in \mathcal{D}} ru(X, T_c)$$

Tính chất 2. [9] Cho tập mục X, nếu $u(X) + ru(X) < \delta$ thì X và tất cả các tập mở rộng của X không phải là HUI.

Định nghĩa 8. [11] (Tập những hạng mục có thể mở rộng một tập mục). Cho tập mục X và thứ tự toàn phần $\succ$. Ta gọi $E(X)$ là tập tất cả các hạng mục có thể sử dụng để mở rộng X, nghĩa là:

$$E(X) = \{z \mid z \in I \wedge z \succ x, \forall x \in X\}$$

Định nghĩa 9. [11] (Mở rộng của một tập mục). Một tập mục Z là một mở rộng của tập X nếu $Z = X \cup W$ với một tập mục $W \in 2^{E(X)}$ và $W \neq \emptyset$. Một tập mục Z là một mở rộng đơn của X nếu $Z = X \cup \{z\}$ với hạng mục $z \in E(X)$.

Định nghĩa 10. [11] (Phép chiếu của giao dịch trên tập mục). Phép chiếu của giao dịch T_c trên tập mục X được ký hiệu là $T_c \backslash X = \{z \in T_c \wedge z \in E(X)\}$.

Định nghĩa 11. [11] (Phép chiếu cơ sở dữ liệu trên tập mục). Phép chiếu cơ sở dữ liệu $\mathcal{D}$ trên tập mục X được ký hiệu là $\mathcal{D}\backslash X == \{T_c\backslash X \mid T_c \in \mathcal{D}\}$.

Định nghĩa 12. (Độ hữu ích tiền tố) [12] Cho tập mục X và một mở rộng $y \in I$ của X, độ hữu ích tiền tố của tập mục Xy trong giao dịch T_c được định nghĩa là: $pu(Xy, T_c) = u(X, T_c)$. Nếu $X = \emptyset$ thì $pu(Xy, T_c) = 0$.

Định nghĩa 13. [11] (Độ hữu ích trên cây con). Cho tập mục X và hạng mục $w \in E(X)$, độ hữu ích trên cây con của w đối với X được tính như sau:

$$su(X, w) = \sum_{T_c \in g(X \cup \{w\})} \left[u(X, T_c) + u(w, T_c) + \sum_{z \in T_c \wedge z \in E(X \cup \{w\})} u(z, T_c) \right]$$

Tính chất 3. [11] Nếu $su(X, w) < \delta$, thì tập mục $X \cup \{w\}$ và tất cả phần mở rộng của nó sẽ có độ hữu ích thấp. Nói cách khác, cây con của $X \cup \{w\}$ trong cây liệt kê có thể được cắt bớt, từ đó rút gọn không gian tìm kiếm.

Ví dụ: Giá trị trọng số hữu ích giao dịch của bc là $twu(bc) = tu(T_2) + tu(T_3) + tu(T_6) + tu(T_7) = 15 + 21 + 20 + 22 = 78$ và $twu(bde) = 37$. Các phần tử sau khi sắp xếp theo thứ tự $\succ$ là: $e \succ b \succ c \succ d \succ a \succ f$. Độ hữu ích của các phần tử sau bc trong T_2 là $ru(bc, T_2) = u(f, T_2) = 6$, độ hữu ích của các phần tử sau bc trong cơ sở dữ liệu là $ru(bc) = ru(bc, T_2) + ru(bc, T_3) + ru(bc, T_6) + ru(bc, T_7) = 6 + 12 + 12 + 9 = 39$. Tập tất cả các hạng mục có thể sử dụng để mở rộng bc là $E(bc) = \{d, a, f\}$.

Định nghĩa 14. [11] (Độ hữu ích cục bộ). Cho tập mục X và hạng mục $w \in E(X)$, độ hữu ích cục bộ của w đối với X được tính như sau:

$$lu(X, w) = \sum_{T_c \in g(X \cup \{w\})} [u(X, T_c) + re(X, T_c)]$$

Tính chất 4. [11] Nếu $lu(X, w) < \delta$, thì tất cả các phần mở rộng của X chứa w đều có độ hữu ích thấp. Hay nói cách khác, mục w có thể được bỏ qua khi khai phá tất cả các cây con của X.

Định nghĩa 15. [11] Cho tập mục X,

$$Primary(X) = \{z \mid z \in E(X) \wedge su(X, z) \geq \delta\}$$

$$Secondary(X) = \{z \mid z \in E(\alpha) \wedge lu(X, z) \geq \delta\}$$

Vì $lu(X, z) \geq su(X, z)$, nên $Primary(X) \subseteq Secondary(X)$.

Định nghĩa 16. [19] (Phép chiếu TIDs của tập mục trên cơ sở dữ liệu giao dịch). Cho cơ sở dữ liệu giao dịch $\mathcal{D}$ và tập mục X. Phép chiếu TIDs của X trên $\mathcal{D}$ được định nghĩa là tập hợp các định danh của các giao dịch chứa X, ký hiệu là P_set của X và

$$P\_set(X) = \{T_c \mid T_c \in \mathcal{D} \wedge X \subseteq T_c\}$$

4. THUẬT TOÁN ĐỀ XUẤT

Để cải thiện thời gian thực thi, nhóm tác giả đã tái sử dụng các chiến lược tỉa như TWU–Prune, EUCS-Prune, U–Prune, C-Prune, LA-Prune và các chiến lược nâng ngưỡng như RIU [16], LIU [2],LIU_LB [2],CUD [17], và COV [17] đã được đề xuất và chứng minh hiệu quả nhằm loại bỏ các mục không tiềm năng. Trong quá trình khai thác HUI, nếu áp dụng tốt các chiến lược tỉa và các chiến lược nâng ngưỡng, thuật toán sẽ loại bỏ được số lượng lớn các tập không thỏa mãn yêu cầu của HUI trong cơ sở dữ liệu.

Bên cạnh việc áp dụng các chiến lược nâng ngưỡng đã có, trong phần này nhóm tác giả đề xuất một chiến lược nâng ngưỡng mới có tên là RTU và đề xuất cấu trúc dữ liệu mới ExtentionP_set để hạn chế việc duyệt cơ sở dữ liệu nhiều lần trong quá trình khai thác top-N HUI.

4.1. Chiến lược tăng ngưỡng RTU

Hai vấn đề có ảnh hưởng lớn tới hiệu suất khai thác top-N HUI trong cơ sở dữ liệu giao dịch đó là việc tăng ngưỡng độ hữu ích tối thiểu và cắt tỉa không gian tìm kiếm trong quá trình khai thác. Nếu việc tăng ngưỡng được thực hiện tốt, sớm xác định được ngưỡng tối thiểu có giá trị cao sẽ giúp loại bỏ

được rất nhiều các ứng viên không tiềm năng, ngược lại quá trình khai thác làm tiêu tốn nhiều thời gian cho các ứng viên không phải các tập nằm trong top-N HUI cần tìm.

Mỗi giao dịch trong cơ sở dữ liệu chính là một tập mục cần được xem xét, do đó độ hữu ích của chúng cũng có thể sử dụng trong quá trình tăng ngưỡng ban đầu cho thuật toán. Trong lần duyệt cơ sở dữ liệu đầu tiên, độ hữu ích của các giao dịch sẽ được xác định. Độ hữu ích của một giao dịch T_c trong cơ sở dữ liệu $\mathcal{D}$ được ký hiệu là $\mathrm{tu}(T_c)$ và được xác định như sau: $\mathrm{tu}(T_c) = \sum_{x \in T_c} u(x, T_c)$.

Tính chất 5. Gọi $\mathrm{RTU} = \{\mathrm{tu}(T_1), \mathrm{tu}(T_2), \mathrm{tu}(T_3), \dots, \mathrm{tu}(T_n)\}$ là tập hợp độ hữu ích của các giao dịch trong $\mathcal{D}$ và $\mathrm{tu}(t_N)$ là giá trị cao nhất thứ N trong RTU. $\mathrm{tu}(T_N)$ không lớn hơn độ hữu ích của các tập trong top-N HUI.

Chứng minh:

Gọi $\{i_1, i_2, \cdots, i_k\}$ là các hạng mục trong giao dịch T_j. Khi đó, $X = \{i_1, i_2, \cdots, i_k\}$ cũng chính là một tập mục và $\mathrm{tu}(T_j)$ (độ hữu ích của giao dịch T_j) cũng là độ hữu ích của X trong giao dịch T_j.

Giả sử cơ sở dữ liệu $\mathcal{D}$ có n giao dịch thì sẽ phát sinh n tập mục $\{X_1, X_2, \dots, X_n\}$ với tập độ hữu ích $\{\mathrm{tu}(T_1), \mathrm{tu}(T_2), \mathrm{tu}(T_3), \dots, \mathrm{tu}(T_n)\}$ tương ứng. Do mỗi tập mục X_i là một phần tử trong không gian tìm kiếm top-N nên giá trị độ hữu ích thứ N trong tập $\{\mathrm{tu}(T_1), \mathrm{tu}(T_2), \mathrm{tu}(T_3), \dots, \mathrm{tu}(T_n)\}$ có thể dùng làm ngưỡng độ hữu ích tối thiểu (δ) trong việc chọn ra N tập mục có độ hữu ích cao nhất trong cơ sở dữ liệu $\mathcal{D}$. Vì vậy, giá trị độ hữu ích cao thứ N của $\mathcal{D}$ sẽ không bé hơn $\mathrm{tu}(T_N)$.

Từ tính chất trên, thuật toán có thể dựa vào độ hữu ích của giao dịch để xác định giá trị ngưỡng δ trong quá trình khai thác top-N HUI.

4.2. Cấu trúc hàng đợi ưu tiên toàn cục (priorityQueue)

Trong quá trình thực hiện khai thác top-N HUI, nhiều chiến lược tăng ngưỡng độ hữu ích tối thiểu δ đã được áp dụng. Khi sử dụng các chiến lược tăng ngưỡng, các chiến lược này sẽ cập nhật lại giá trị của δ khi đã tìm được ít nhất N độ hữu ích và giá trị độ hữu ích thứ N phải lớn hơn giá trị δ hiện tại. Một cách hiệu quả và được áp dụng trong các chiến lược đó là sử dụng cấu trúc hàng đợi ưu tiên chứa N phần tử và khi hàng đợi đầy thì bắt đầu cập nhật lại giá trị cho δ.

Tuy nhiên, với nhiều chiến lược tăng ngưỡng được sử dụng trong quá trình tăng ngưỡng, các hàng đợi ưu tiên này chỉ được dùng một cách cục bộ. Khi áp dụng một chiến lược tăng ngưỡng bất kỳ, một hàng đợi mới được khởi tạo và phải thực hiện lại việc lấp đầy N giá trị trước khi tăng ngưỡng và không tái sử dụng các giá trị độ hữu ích đã được tìm thấy và đưa vào hàng đợi trong chiến lược tăng ngưỡng đã sử dụng trước đó. Khi N có giá trị càng lớn, thời gian tiêu tốn càng nhiều và trong nhiều trường hợp sẽ không tăng ngưỡng được cho δ (ví dụ minh họa trong bước 16 phần 4.5). Từ đó, nghiên cứu này đề xuất sử dụng một hàng đợi ưu tiên toàn cục, đặt tên là priorityQueue, được sử dụng chung cho tất cả các chiến lược tỉa áp dụng trong toàn bộ quá trình tăng ngưỡng để giải quyết vấn đề trên.

4.3. Cấu trúc ExtentionP_set

Để cải tiến hiệu suất thực thi của thuật toán EFIM, Nguyen và cộng sự [19] đã đề xuất cấu trúc P_set(X) nhằm lưu lại các TID của các giao dịch chứa X. Chính nhờ cấu trúc P_set đã giúp cho thuật toán *MEFIM* không cần quét lại toàn bộ cơ sở dữ liệu khi chiếu trên tập X nhằm xác định cơ sở dữ liệu chiếu $\mathcal{D}\backslash X$.

Tuy nhiên, P_set(X) chỉ lưu lại các TID chứa tập mục X để tránh phải duyệt các giao dịch mà không chứa X trong cơ sở dữ liệu. Khi mở rộng X, thuật toán vẫn phải duyệt và tìm các mục mở rộng của X trong các giao dịch để tính được độ hữu ích và độ hữu ích còn lại của các mục mở rộng đó. Từ đó thuật toán sẽ tạo ra các bản sao của giao dịch X để chuẩn bị cho việc mở rộng X ở các bước tiếp theo. Vì vậy bộ nhớ cần sử dụng, thời gian tiêu tốn cũng tương tự như thuật toán *EFIM*. Để giải quyết vấn đề này, nhóm tác giả đề xuất cấu trúc mở rộng được phát triển dựa trên ý tưởng của cấu trúc P_set được gọi là ExtentionP_set. Cấu trúc này là một mảng mà mỗi phần tử là một bộ gồm 04 thông tin $<$ tid, pu, u, ru $>$ trong đó:

- TID: là định danh của giao dịch chứa tập mục X.
- pu: là độ hữu ích tiền tố của tập mục X trong giao dịch.
- u: là độ hữu ích của tập mục X trong giao dịch.

– ru: là độ hữu ích còn lại của tập mục X trong giao dịch.

Với cách lưu trữ này, thuật toán sẽ không phải quét lại cơ sở dữ liệu để tìm vị trí của tập mục X trong mỗi giao dịch cũng như xác định lại các giá trị ru và u. Vì ExtentionP_set đã lưu lại cái các giá trị u, ru và pu nên khi mở rộng X thuật toán không cần phải duyệt lại cơ sở dữ liệu mà dựa vào các tham số này để thực hiện. Do đó giảm thời gian cũng như không cần tạo lại các giao dịch mới khi chiếu.

Ví dụ: Từ cơ sở dữ liệu $\mathcal{D}$ trong Bảng 1, ta có

$$\text{ExtentionP\_set}\ (\emptyset, a) = \{< 1,0,4,13 >, < 5,0,4,3 >, < 6,0,4,16 >\}$$

$$\text{ExtentionP\_set}\ (\emptyset, f) = \{< 2,0,6,9 >, < 5,0,3,0 >, < 7,0,9,13 >\}$$

Từ đó việc xác định $\text{ExtentionP\_set}\ (a, f)$ sẽ được thực hiện bằng thao tác kết hợp $\text{ExtentionP\_set}\ (\emptyset, a)$ và $\text{ExtentionP\_set}\ (\emptyset, f)$. Ta thấy f và a có chung giao dịch T_5 nên kết quả thu được là một bộ duy nhất với $\text{TID} = 5$, $u = u(a) + u(f) - pu(a) = 3 + 4 - 0 = 7$, và $ru = ru(f) = 0$ và $pu = u(a) = 3$. Khi đó,

$$\text{ExtentionP\_set}\ (a, f) = \{< 5,4,7,0 >\}$$

4.4. Thuật toán đề xuất

Dựa trên cơ sở thuật toán khai thác tập hữu ích cao EFIM, nghiên cứu cải tiến và áp dụng các đề xuất ở trên để đưa ra phương pháp khai thác một cách hiệu quả N tập hữu ích cao nhất trong cơ sở dữ liệu giao dịch có tên *topNEFIM*.

Thuật toán 1: Thuật toán *topNEFIM*

Input: $\mathcal{D}$: Cơ sở dữ liệu giao dịch, N: Số lượng tập HUI muốn khai thác

Output: topN: tập hợp chứa N tập mục có độ hữu ích cao nhất trong $\mathcal{D}$

1	Gán $\delta = 1$
2	Gán $\alpha = \emptyset$
3	Duyệt $\mathcal{D}$ để tính RTU của cơ sở dữ liệu, tính $lu(\alpha, i)$ và $riu(i)$ của tất cả các $i \in I$
4	Khởi tạo hàng đợi ưu tiên toàn cục priorityQueue
5	Cập nhật giá trị δ sử dụng chiến lược tăng ngưỡng TU;
6	Cập nhật δ sử dụng chiến lược RIU;
7	Xác định $\text{Secondary}(\alpha) = \{i \mid i \in I \wedge lu(\alpha, i) \geq \delta\}$;
8	Xây dựng thứ tự toàn cục $\succ$ tăng dần theo $lu(\alpha, i)$ cho tất cả các hạng mục trong $\text{Secondary}(\alpha)$
9	Loại bỏ các hạng mục i mà $i \notin \text{Secondary}(\alpha)$ trong tất cả giao dịch, và loại bỏ các giao dịch rỗng
10	Sắp xếp các hạng mục trong các giao dịch theo thứ tự $\succ$
11	Tính LIU và cập nhật δ sử dụng chiến lược LIU
12	Tính LIU_LB và cập nhật δ sử dụng chiến lược LIU_LB
13	Tính $su(\alpha, i)$ và $\text{ExtentionP_set}(\alpha, i)$của tất cả các $i \in \text{Secondary}(\alpha)$;
14	$\text{Primary}(\alpha) = \{i \mid i \in \text{Secondary}(\alpha) \wedge su(\alpha, i) \geq \delta\}$;
15	Xây dựng EUCS và CUDM;
16	Cập nhật δ sử dụng chiến lược CUD;
17	$\text{topN} = \textbf{ExTopNEFIM}(\alpha, \text{Secondary}(\alpha), \text{Primary}(\alpha), \text{ExtentionP\_set}, 0, N)$;

topNEFIM là hàm xử lý chính của thuật toán nghiên cứu đề xuất. Đầu vào của *topNEFIM* là cơ sở dữ liệu $\mathcal{D}$ và số nguyên N. Đầu ra của thuật toán là N tập có độ hữu ích cao nhất trong cơ sở dữ liệu. Trong dòng 1 – 2, ngưỡng độ hữu ích được khởi tạo $\delta = 1$ và khởi tạo tập tiền tố $\alpha = \emptyset$. Dòng 3, thuật toán duyệt cơ sở dữ liệu để xác định các giá trị trong *RTU* của cơ sở dữ liệu, đồng thời thuật toán cũng xác định giá trị $\mathrm{lu}(\alpha, i)$ và $\mathrm{riu}(i)$ của tất cả các hạng mục. Cấu trúc hàng đợi ưu tiên priorityQueue được khởi tạo để sử dụng cho các chiến lược tỉa của thuật toán tại dòng 4. Dòng 5 – 6, thuật toán áp dụng chiến lược tăng ngưỡng để cập nhật lại giá trị mới cho ngưỡng δ. Từ dòng 7 – 10, thuật toán xác định tập Secondary, xây dựng thứ tự toàn cục và điều chỉnh lại cơ sở dữ liệu ban đầu để loại bỏ các hạng mục không tiềm năng và các giao dịch rỗng.

Dòng từ 11 – 12, chiến lược tăng ngưỡng LIU và LIU_LB được vận dụng để cập nhật giá trị cho δ. Tại dòng 13, thuật toán tiến hành xác định ExtentionP_set của tất cả các hạng mục trong tập Secondary và lưu trữ trong một danh sách. Từ dòng 14–16, thuật toán xác định các hạng mục cần được khai phá, áp dụng chiến lược tăng ngưỡng CUD và xây dựng bảng EUCS. Để xác định các tập thuộc top-N HUI, thuật toán sử dụng hàm ExTopNEFIM tại bước 17.

Thuật toán 2: Thuật toán ExTopNEFIM

Input:α: tập mục trong $\mathcal{D}$;
SeSet, PriSet: danh sách các hạng mục trong Secondary và Primary của α;
lstExPset: Danh sách ExtentionP_set của α;
len: số lượng phần tử của α;
N: số tập HUI muốn khai thác
Output: topHUI: tập các HUI thuộc topN_HUIs được mở rộng từ α

```
1   Foreach hạng mục i ∈ PriSet do
2       Gán β = α ∪ {i};
3       Tính u(β) và ru(β) sử dụng ExtentionP_set(α, i)
4       If u(β) ≥ δ then Thêm β vào topN_HUIs và cập nhật δ
5       If u(β) + ru(β) ≥ δ then
6           P = ∅, S = ∅, lstExPsetβ = ∅
7           Foreach hạng mục z ∈ SeSet do
8               If i ≻ z and EUCS[i, z] ≥ δthen
9                   Xác định ExtentionP_set(β, z), lu(β, z), su(β, z) dựa trên
                    ExtentionP_set(α, i) và ExtentionP_set(α, z)
10                  If su(β, z) ≥ δ then P = P ∪ {z};
11                  If lu(β, z) ≥ δ then
12                      S = S ∪ {z};
13                      Thêm ExtentionP_set(β, z), vào lstExPsetβ
14                  EndIf
15              EndIf
16          EndFor
17          ExTopNEFIM(β, S, P, lstExPsetβ, len + 1, N)
18      EndIf
19  EndFor
```

Thuật toán ExTopNEFIM được thực hiện theo cơ chế đệ quy để phát hiện tất cả các tập thuộc top-N HUI trong $\mathcal{D}$. Thuật toán lần lượt xét tất cả các hạng mục i có khả năng mở rộng của α trong Primary.

Dòng 2–3, gán $\beta = \alpha \cup \{i\}$ và xác định độ hữu ích cũng như độ hữu ích còn lại của β dựa vào ExtentisonP_set(α, i). Nếu u(β) thỏa ngưỡng hiện tại thì β được đưa vào tập topN_HUIs và giá trị δ được cập nhật lại tại dòng 4. Dòng 5 thuật toán xem xét khả năng mở rộng của β, nếu β thỏa điều kiện mở rộng thì tiến hành việc xác định các tập Primary và Secondary của β. Từ dòng 7–16, khi điều kiện mở rộng của β đã được kiểm tra bằng cách xét tất cả các hạng mục z đứng sau I và thỏa chiến lược tỉa EUCP trong tập Secondary của α, thuật toán kết hợp ExtentionP_set(α, i) và ExtentionP_set(α, z) để xây dựng ExtentionP_set(β, z). Trong quá trình thực hiện, thuật toán vận dụng chiến LA-Prune và U-Prune để xác định giá trị lu(β, z) và su(β, z). Dòng 11–14, thuật toán xác định việc cập nhật z vào Secondary và Primary của β, thêm ExtentionP_set(β, z) và danh sách các ExtentionP_set của β. Cuối cùng, thuật toán gọi đệ quy với các tham số vừa được xác định để kiểm tra các mở rộng của β có thuộc topN_HUIs hay không?

4.5. Ví dụ minh họa

Trong phần này, thuật toán được minh họa trên cơ sở dữ liệu $\mathcal{D}$ (trong Bảng 1, 2) với N = 4.

Bước 1–2: Gán $\delta = 1$, $\alpha = \emptyset$.

Bước 3: Tính RTU, lu(α, i) và riu(i) của tất cả các hạng mục i ∈ I trong $\mathcal{D}$ (Kết quả được trình bày trong các Bảng 3, 4)

Bảng 3. Giá trị RTU của các hạng mục

Tid	1	2	3	4	5	6	7
RTU	17	15	21	9	7	20	22

Bảng 4. Giá trị lu và riu của các hạng mục

Hạng mục	a	b	c	d	e	f
lu	44	95	87	67	104	44
riu	12	27	14	28	12	18

Bước 4: Khởi tạo priorityQueue rỗng.

Bước 5: Sử dụng chiến lược tăng ngưỡng RTU. Trong bảng 3, với $N = 4$, có 4 giá trị RTU cao nhất là $\{17, 21, 20, 22\}$. Vì vậy, thuật toán cập nhật $\delta = 17$ và priorityQueue $= \{17, 21, 20, 22\}$.

Bước 6: Cập nhật ngưỡng bằng chiến lược RIU. Sử dụng priorityQueue ở bước 5 kết hợp với giá trị riu trong bảng 4, thuật toán thu được priorityQueue $= \{21, 22, 27, 28\}$ và $\delta = 21$.

Bước 7: Xác định Secondary(α) = {a, b, c, d, e, f}, vì tất cả các hạng mục này đều có $lu > \delta =$ 21 (bảng 4).

Bước 8: Sắp xếp tăng dần các hạng mục trong Secondary(α) theo lu và xây dựng thứ tự toàn cục : a ≻ f ≻ d ≻ c ≻ b ≻ e.

Bước 9-10: Loại bỏ các hạng mục không thuộc Secondary và sắp xếp lại các hạng mục trong mỗi giao dịch (trong ví dụ này, không có hạng mục nào bị loại bỏ). Ta được cơ sở dữ liệu đã chỉnh sửa như Bảng 5.

Bảng 5. Cơ sở dữ liệu sau điều chỉnh

TID	Giao dịch	Số lượng	Độ hữu ích	Độ hữu ích của giao dịch
1	a d b e	1 2 2 3	4 4 6 3	17
2	f c b e	2 5 1 1	6 5 3 1	15
3	d c b e	6 1 2 2	12 1 6 2	21
4	d c e	2 2 3	4 2 3	9
5	a f	1 1	4 3	7
6	a d c b e	1 4 4 1 1	4 8 4 3 1	20
7	f c b e	3 2 3 2	9 2 9 2	22

Bước 11: Tính LIU (Bảng 6) và sử dụng LIU để cập nhật δ.

Bảng 6. Ma trận LIU

7	0	0	0	0
	0	0	0	0
		31	34	37
			33	39
				36

Bảng 7. Giá trị của priorityQueue

Priority	δ
22, 27, 28, 31	22
27, 28, 31, 34	27
28, 31, 34, 37	28
31, 34, 37, 33	31
33, 34, 37, 39	33
34, 37, 39, 36	34

Bằng cách sử dụng priorityQueue, quá trình cập nhật giá trị của δ được thể hiện trong Bảng 7. Cuối cùng, giá trị của $\delta = 34$. Thuật toán cập nhật lại giá trị của $\delta = 34$.

Bước 12: Thuật toán sử dụng LIU_LB. Tuy nhiên trong ví dụ này, khi áp dụng LIU_LB giá tri của $\delta = 34$ không thay đổi.

Bước 13: Thuật toán xác định $\mathrm{su}(\alpha, i)$ và $\mathrm{ExtentionP\_set}(\alpha, i)$.

Xét hạng mục a: có 3 giao dịch có chứa a là T_1, T_5, T_6. Với mỗi giao dịch này, tính pu(a), u(a), ru(a) trong giao dịch và thêm bộ $< \mathrm{tid}, \mathrm{pu}, \mathrm{u}, \mathrm{ru} >$ này vào ExtentionP_set. Vì $\mathrm{pu}(a, T_1) = 0$; $\mathrm{u}(a, T_1) = 4$; $\mathrm{ru}(a, T_1) = 13$ nên bộ đầu tiên được thêm vào $\mathrm{ExtentionP\_set}(\emptyset, a)$ là $< 1,0,4,13 >$. Thực hiện tương tự cho các giao dịch T_5, T_6. Kết quả: với mục mở rộng $i = a$, tập $\mathrm{ExtentionP\_set}(\emptyset, a)$ $=\{< 1,0,4,13 >, < 5,0,4,3 >, < 6,0,4,16 >\}$.

Tiếp tục thực hiện cho các hạng mục còn lại, kết quả được trình bày trong Bảng 8:

Bảng 8. Giá trị ExtentionP_set và su

i	ExtentionP_set(α, i)	su
a	$< 1,0,4,13 >, < 5,0,4,3 >, < 6,0,4,16 >$	44
f	$< 2,0,6,9 >, < 5,0,3,0 >, < 7,0,9,13 >$	40
d	$< 1,0,4,9 >, < 3,0,12,9 >, < 4,0,4,5 >, < 6,0,8,8 >$	58
c	$< 2,0,5,4 >, < 3,0,1,8 >, < 4,0,2,3 >, < 6,0,4,4 >, < 7,0,2,11 >$	44
b	$< 1,0,6,3 >, < 2,0,3,1 >, < 3,0,6,2 >, < 6,0,3,1 >, < 7,0,9,2 >$	36
e	$< 1,0,3,0 >, < 2,0,1,0 >, < 3,0,2,0 >, < 4,0,3,0 >< 6,0,1,0 >, < 7,0,2,0 >$	12

Bước 14: Tính $\mathrm{Primary}(\alpha) = \{a, f, d, c, b\}$; loại {e} vì $\mathrm{su}(e) = 12 < \delta = 34$.

Bước 15: Xây dựng EUCS và CUDM (kết quả như Bảng 9)

Bước 16: Sử dụng chiến lược CUD để cập nhật ngưỡng δ. Từ kết quả của bước 11, $\mathrm{priorityQueue} = \{34, 37, 39, 36\}$ và $\delta = 34$. Khi duyệt ma trận CUD, chiến lược tăng ngưỡng CUD chỉ xét các giá trị lớn hơn δ để đưa vào priorityQueue đồng thời cập nhật lại δ. Khi xét tới giá trị $\delta < 39$, thì 39 được thêm vào priorityQueue và cập nhật δ, do đó ta có priorityQueue ={36, 37, 39, 39} và $\delta = 36$. Xét $37 > \delta$, thực hiện tương tự thì priorityQueue ={37, 39, 39, 37} và $\delta = 37$. Do không còn giá trị nào lớn hơn δ nên kết thúc chiến lược CUD với $\mathrm{priorityQueue} = \{37, 39, 39, 37\}$ và $\delta = 37$.

Giả sử, nếu thuật toán không sử dụng priorityQueue toàn cục mà khởi tạo một priorityQueue mới. Với $\delta = 34$ (kết quả bước 11), thuật toán CUD sẽ xét và thêm các giá trị lớn hơn δ vào priorityQueue cho tới khi đủ $k = 4$ thì mới cập nhật được giá trị của δ. Do đó khi duyệt hết ma trận CUD chỉ có 3 giá trị mới được thêm vào priorityQueue là 39, 37, 36 nên không cập nhật được giá trị mới cho δ, nên δ sẽ giữ nguyên giá trị là 34.

Bước 17: Gọi ExTopNEFIM để xác định topN_HUIs với $\alpha = \emptyset$, $N = 4$, ExtentionP_set như Bảng 9, $Secondary(\alpha) = \{a, f, d, c, b, e\}$, $Primary(\alpha) = \{a, f, d, c, b\}$.

Xét hạng mục $i = \{a\}$ và $\beta = \alpha \cup \{i\} = \{a\}$; Tính $u(\beta) = 12, ru(\beta) = 32$; $u(\beta) < \delta$ nên β không được thêm vào topN_HUIs. Xét $u(\beta) + ru(\beta) = 44 \geq \delta$ nên thuật toán xét mở rộng của β.

Tính $lu(\beta, z)$, $su(\beta, z)$ và $ExtentionP\_set(\beta, z)$ của tất cả $a \succ z$ trong Secondary. Kết quả được hiện thực trong Bảng 10.

Bảng 9. Ma trận CUD

7	20	8	17	12
0	22	27	18	
31	39	37		
33	23			
36				

Bảng 10. Giá trị ExtentionP_set, lu và su của tiền tố β

z	ExtentionP_set(β, z)	lu	su
f	$< 5,4,7,0 >$	7	
d	$< 1,4,8,9 >, < 6,4,12,8 >$	37	37
c	$< 6,4,8,4 >$	20	
b	$< 1,4,10,3 >, < 6,4,7,1 >$	37	21
e	$< 1,4,7,0 >, < 6,4,5,0 >$	37	12

$P = \{d\}$, $S = \{d, b, e\}$,

$lstExPset\beta = \{ExtentionP\_set(\beta, d), ExtentionP\_set(\beta, b), ExtentionP\_set(\beta, e)\}$.

Gọi đệ quy ExTopNEFIM với $\beta = \{a\}$, $N = 4$, $P = \{d\}$, $S = \{d, b, e\}$ và $lstExPset\beta$.

Thuật toán tiếp tục thực hiện đến khi kết thúc. Kết quả cuối cùng thuật toán trả về như sau: $top - N\ HUIs = \{\{c, d, e\}: 37, \{b, d\}: 39, \{b, c, e\}: 39, \{b, d, e\}: 45\}$.

4.6. Độ phức tạp của thuật toán

Độ phức tạp của thuật toán *EFIM* về mặt thời gian là $O(lnw)$ với l, n, w lần lượt là số lượng tập mục trong không gian tìm kiếm, số lượng giao dịch của cơ sở dữ liệu và độ dài trung bình của các giao dịch. Thuật toán $topNEFIM$ là mở rộng của thuật toán EFIM và để giảm chi phí cho việc quét cơ sở dữ liệu bằng cách xây dựng cấu trúc ExtentionP_set. Trước hết, việc điều chỉnh này cũng không thay đổi độ phức tạp của thuật toán bởi vì cấu ExtentionP_set được xây dựng dựa trên việc quét cơ sở dữ liệu.

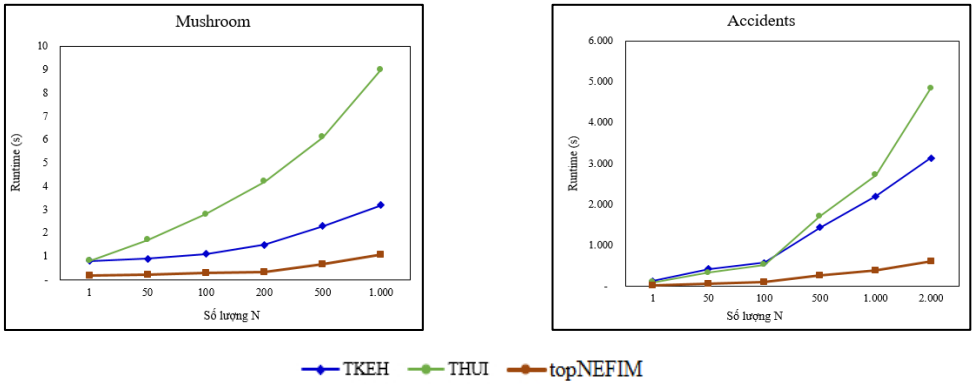
Hơn thế nữa, việc thêm và truy cập vào các phần tử trong ExtentionP_set là $O(1)$ vìExtentionP_set lưu cấu trúc mảng. Giá trị l là tổng số giao dịch chứa trong ExtentionP_set do đó $l = |ExtentionP\_set|$. Nếu $\mathcal{D}$ là cơ sở dữ liệu thưa, l sẽ rất nhỏ nên thời gian thực thi sẽ giảm nhiều. Trong trường hợp khi $\mathcal{D}$ là cơ sở dữ liệu rất dày ($l = |D|$) và độ phức tạp của $topNEFIM$ chính là độ phức tạp của thuật *EFIM*. Độ lớn của không gian tìm kiếm trong thuật toán khai thác top-HUI còn phụ thuộc vào hiệu quả của các chiến lược tăng ngưỡng. Nếu các chiến lược tăng ngưỡng hiệu quả, giá trị ngưỡng cao thì số lượng tập mục trong không gian tìm kiếm sẽ giảm nhưng về mặt độ phức tạp của thuật toán sẽ không thay đổi.

5. THỰC NGHIỆM

Để đánh giá tính hiệu quả, thuật toán *topNEFIM* đã được cài đặt trên ngôn ngữ Java sử dụng máy tính Dell Latutide 7490 có bộ xử lý Core i7 1.90 GHz với bộ nhớ chính 32GB, trên nền hệ điều hành Windows 10. Thuật toán đã chạy thử nghiệm trên các bộ dữ liệu đã được công bố trên webstie SPMF, cụ thể trên các bộ dữ liệu bao gồm cả dữ liệu thưa và dữ liệu dày [20]. Đặc điểm chi tiết của các bộ dữ liệu này được thể hiện trong Bảng 11. Trong quá trình thực nghiệm, kết quả sẽ được so sánh với các thuật toán mới nhất đã được công bố gần đây về cùng chủ đề khai thác top-N tập hữu ích cao, cụ thể là hai thuật toán *TKEH* và *THUI*. Thuật toán đã đề xuất được hiện thực trên nền ngôn ngữ lập trình Java và triển khai trên nền Java SDK 8.0.

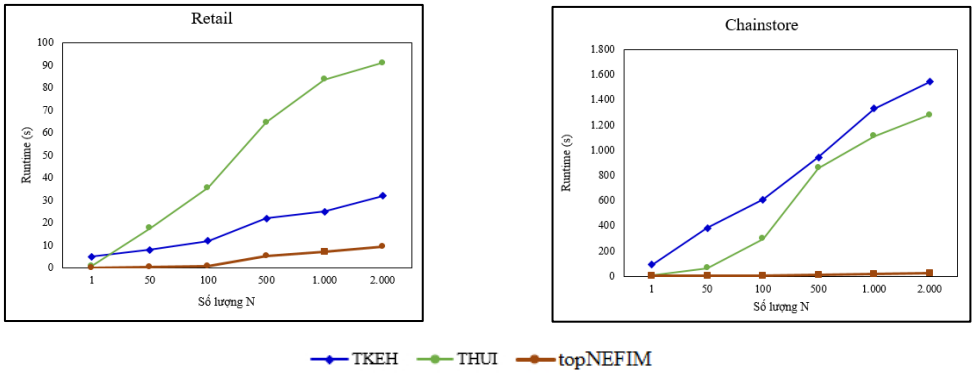
Bảng 11. Đặc trưng của các cơ sở dữ liệu

Tên cơ sở dữ liệu	Số lượng giao dịch	Hạng mục	Độ dài trung bình của giao dịch	Mật độ
Chainstore	1.112.949	46.086	7,3	0,00016
Retail	88.162	16.470	10,3	0,00063
Accidents	340.183	468	33,8	0,0722
Mushroom	8.124	119	23,0	0,1933



Hình 1. Thời gian thực thi trên các cơ sở dữ liệu dày

Biểu đồ Hình 1 cho thấy kết quả so sánh thời gian thực thi giữa thuật toán *topNEFIM*, *TKEH* và *THUI* trên cơ sở dữ liệu dày Mushroom và Accidents. Trong các trường hợp, với tất cả các giá trị N thực nghiệm, thời gian thực thi của thuật toán *topNEFIM* ổn định và nhanh hơn rất nhiều lần so với 02 thuật toán còn lại. Cụ thể, với cơ sở dữ liệu Mushroom: tốc độ thực thi của *topNEFIM* luôn nhanh hơn *TKEH* từ 3 đến 5 lần và nhanh hơn *THUI* từ 4 đến 13 lần; với cơ sở dữ liệu Accidents, thời gian thực thi của *topNEFIM* ít hơn từ 5 đến 7 lần so với *TKEH* và ít hơn 5 đến 8 lần so với *THUI*. Thời gian thực thi của thuật toán *TKEH* và *THUI* tăng nhiều và nhanh khi giá trị N ngày càng lớn. Điều này cũng là hiển nhiên bởi vì thuật toán *topNEFIM* đã tiến hành nâng ngưỡng sớm ngay lần đầu tiên duyệt cơ sở dữ liệu bằng chiến lược tăng ngưỡng RTU. Điều đó đã giúp cho giá trị ngưỡng ban đầu được khởi tạo sớm và cao hơn thuật toán *TKEH* và *THUI*. Giá trị ngưỡng độ hữu ích cao đã giúp chiến lược tỉa được áp dụng có thể loại được nhiều ứng viên hơn, làm giảm không gian cũng như thời gian thuật toán tiêu tốn. Thêm vào đó, việc áp dụng kỹ thuật tái sử dụng các giá trị hữu ích của các chiến lược tăng ngưỡng trước để tiếp tục sử dụng trong các chiến lược tăng ngưỡng sau làm cho việc tăng giá trị của δ được hiệu quả hơn, từ đó các ứng viên sẽ bị loại nhiều và nhanh hơn trong quá trình khai thác tập hữu ích cao.



Hình 2. Thời gian thực thi trên các cơ sở dữ liệu thưa

Hình 2 cho thấy thuật toán *topNEFIM* thực thi đặc biệt hiệu quả trên các cơ sở dữ liệu thưa. Cụ thể, với cơ sở dữ liệu Retail: tốc độ thực thi của *topNEFIM* nhanh hơn *TKEH* từ 3lần trở lên, thậm chí nhanh hơn 25 lần (với $N = 1$), 29 lần (với $N = 50$) và thuật toán đề xuất đã thực hiện nhanh hơn *THUI* tới 47 lần (với $N = 100$) và 65 lần (với $N = 50$); với cơ sở dữ liệu Chainstore (đây là cơ sở dữ liệu thưa

nhất trong các các dữ liệu thử nghiệm) tốc độ của *topNEFIM* nhanh *TKEH* từ 58 đến 134 lần, và nhanh hơn *THUI* từ 53 tới 76 lần khi từ N được chọn từ 100 trở lên. Kết quả thực nghiệm này càng chứng tỏ được tính hiệu quả của việc tăng ngưỡng sớm cũng như việc sử dụng cấu trúc toàn cục để tái sử dụng các giá trị độ hữu ích đã tìm được trong các chiến lược tăng ngưỡng. Do trong cơ sở dữ liệu thưa, số lượng các hạng mục trong mỗi giao dịch là rất ít nên cấu trúc ExtentionP_set lại càng hiệu quả hơn, giúp giảm thiểu rất nhiều thời gian duyệt cơ sở dữ liệu khi xem xét các ứng viên. Chính điều đó đã tạo nên hiệu quả về tốc độ thực thi của thuật toán *topNEFIM* như Hình 2.

Bảng 12. Bảng đánh giá hiệu quả của chiến lược RTU

N	Mushroom				Accidents				Retail			
	δ		HMTN		δ		HMTN		δ		HMTN	
	RTU	NRTU	RTU	NRTU	RTU	NRTU	RTU	NRTU	RTU	NRTU	RTU	NRTU
100	807	119	229	119	276.143	276.143	229	229	13.840	13.840	3.847	3.847
500	537	119	379	119	849	1	379	468	5.115	5.115	6.668	6.668
1.000	504	119	300	119	826	1	300	468	2.956	2.956	8.266	8.266
2.000	466	119	380	119	803	1	380	468	1.552	1.552	10.275	10.275
3.000	438	119	382	119	789	1	382	486	964	964	11.712	11.712
5.000	395	119	386	119	769	1	386	486	448	444	13.833	13.854
10.000	1	119	391	119	741	1	391	486	349	88	14.369	16.027

Các thuật toán *TKEH* và *THUI* chỉ áp dụng chiến lược nâng ngưỡng RUI để xác định giá trị độ hữu ích tối thiểu δ trước khi tiến hành loại bỏ các hạng mục không tiềm năng. Thuật toán *topNEFIM*, sử dụng cả chiến lược tăng ngưỡng RIU và chiến lược tăng ngưỡng RTU trước khi loại bỏ các hạng mục không tiềm năng (hạng mục không thuộc Secondary tại bước 9) trong cơ sở dữ liệu. Để đánh giá tính hiệu quả của chiến lược tăng ngưỡng RTU, nghiên cứu đã tiến hành thực nghiệm để đo giá trị δ đạt được, số lượng hạng mục tiềm năng (HMTN) mà thuật toán giữ lại để khai thác trong thuật toán *topNEFIM* trong 02 trường hợp: có sử dụng chiến lược RTU và không sử dụng chiến lược RTU (NRTU). Kết quả trong Bảng 13 cho thấy: với cơ sở dữ liệu Mushroom do số lượng các hạng mục ít, dù số lượng hạng mục không tiềm năng không giảm nhưng giá trị ngưỡng khi sử dụng RTU lớn hơn so với không sử dụng RTU. Đối với cơ sở dữ liệu Accidents, khi áp dụng RTU giá trị ngưỡng đã cao hơn so với không áp dụng RTU khi N nhận giá trị lớn hơn 500. Điều đó đã giúp cho thuật toán loại bỏ được một lượng khá lớn các hạng mục không tiềm năng (không có trong các HUI) giúp cho không gian tìm kiếm và số lượng ứng viên được thu gọn. Điều đó cũng được thể hiện khi thực nghiệm trên cơ sở dữ liệu Retail. Tuy nhiên với cơ sở dữ liệu Chainstore, số lượng hạng mục rất lớn và các giao dịch ngắn nên chiến lược RTU chưa mang lại hiệu quả cao.

6. KẾT LUẬN VÀ KIẾN NGHỊ

Qua nghiên cứu này, nhóm tác giả đã đề xuất chiến lược tăng ngưỡng RTU để xác định ngưỡng tối thiểu ban đầu áp dụng trong bài toán khai thác top-N HUI. Nghiên cứu xem xét mỗi giao dịch trong cơ sở dữ liệu là một tập mục do đó độ hữu ích của các giao dịch cũng chính là một giá trị có thể sử dụng trong quá trình tăng ngưỡng của thuật toán. Vì lý do đó, độ hữu ích của giao dịch (TU) được tính toán và sử dụng để tăng ngưỡng ngay khi duyệt cơ sở dữ liệu lần đầu tiên. Hơn nữa, trong các thuật toán khác, khi các chiến lược nâng ngưỡng được áp dụng, chúng phải xác định lại N giá trị hữu ích thỏa mãn ngưỡng hiện tại. Nếu N là một giá trị lớn, công việc này cũng tiêu tốn nhiều thời gian và làm cho hiệu suất thực thi của thuật toán không được tối ưu. Để khắc phục vấn đề này, nhóm tác giả đã xây dựng cấu trúc lưu trữ toàn cục cho các giá trị hữu ích và sử dụng chung trong tất cả các chiến lược nâng ngưỡng được áp dụng trong quá trình khai thác. Với kỹ thuật này, hiệu suất của thuật toán trở nên tốt hơn vì ngưỡng tối thiểu tăng nhanh hơn.

Cuối cùng, nghiên cứu đã đề xuất thuật toán *topNEFIM* để khai thác các top-N HUI một cách hiệu quả về thời gian. Thuật toán *topNEFIM* đã hạn chế việc duyệt lại cơ sở dữ liệu bằng cách phát triển cấu

trúc lưu trữ ExtentionP_set dựa trên cấu trúc P_set. Cùng với việc kết hợp nhiều chiến lược tăng ngưỡng như RTU, RUI, LIU, COD, COV và vận dụng các chiến lược cắt tỉa không gian tìm kiếm như TWU-Prune, C-Prune, U-Prune, LA-Prune, EUCS-Prune đã giúp cho thuật toán đề xuất tối ưu hơn các thuật toán TKEH và THUI rất nhiều lần về thời gian thực thi trên cả cơ sở dữ liệu dày và thưa.

Trong thời gian tới, nghiên cứu sẽ tập trung nhiều hơn vào khai thác top-N HUI trên cơ sở dữ liệu động và cơ sở dữ liệu chứa các hạng mục có lợi ích âm.

Lời cảm ơn: Nghiên cứu này do Trường Đại học Công nghiệp Thực phẩm Thành phố Hồ Chí Minh (nay là Trường Đại học Công Thương Thành phố Hồ Chí Minh) bảo trợ và cấp kinh phí theo Hợp đồng số 07/HĐ-DCT ngày 05 tháng 01 năm 2021.

TÀI LIỆU THAM KHẢO

1. Singh K., Singh S. S., Kumar A., and Biswas B. - TKEH: an efficient algorithm for mining top-k high utility itemsets, Applied Intelligence **49** (3) (2019) 1078–1097. https://doi.org/10.1007/s10489-018-1316-x.
2. Krishnamoorthy S. - Mining top-k high utility itemsets with effective threshold raising strategies, Expert Syst Appl **117** (2019) 148–165. https://doi.org/10.1016/ j.eswa.2018.09.051.
3. Yao H., Hamilton H. J., and Butz G. J. - A foundational approach to mining itemset utilities from databases, Proc West Mark Ed Assoc Conf (2004) 482–486. https://doi.org/10.1137/1.9781611972740.51.
4. Liu Y., Liao W. K., and Choudhary A. - A two-phase algorithm for fast discovery of high utility itemsets, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **3518** LNAI(2005) 689–695. https://doi.org/10.1007/11430919_79.
5. Agrawal R. and Srikant R. - Fast Algorithms for Mining Association Rules in Large Databases, in Proceedings of the 20th International Conference on Very Large Data Bases (1994) 478–499.
6. Bac Le, Huy Nguyen, Tung Anh Cao, and Bay Vo - A novel algorithm for mining high utility itemsets, Proceedings - 2009 1st Asian Conference on Intelligent Information and Database Systems, ACIIDS (2009) 13–17. https://doi.org/10.1109/ACIIDS.2009.55.
7. Bac Le, Huy Nguyen, and Bay Vo - An efficient strategy for minings high utility itemsets, International Journal of Intelligent Information and Database Systems **5** (2) (2011) 64–176.
8. Ahmed C. F., Tanbeer S. K., Jeong B. S., and Lee Y. K. - Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans Knowl Data Eng **21** (12) (2009) 1708–1721. https://doi.org/10.1109/TKDE.2009.46.
9. Liu M. and Qu J. - Mining high utility itemsets without candidate generation, in Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM ’12 (2012) 55-64. https://doi.org/10.1145/2396761.2396773.
10. Fournier-Viger P., Wu C. W., Zida S., and Tseng V. S. - FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **8502** (2014) 83–92. https://doi.org/10.1007/978-3-319-08326-1_9.
11. Zida S., Fournier-Viger P., Lin J. C. W., Wu C. W., and Tseng V. S. - EFIM: a fast and memory efficient algorithm for high-utility itemset mining, Knowl Inf Syst **51** (2) (2017) 595–625. https://doi.org/10.1007/s10115-016-0986-0.
12. Krishnamoorthy S. - HMiner: Efficiently mining high utility itemsets, Expert Syst Appl, **90** (2017) 168–183. https://doi.org/10.1016/j.eswa.2017.08.028.
13. Quang-Huy Duong, Fournier-Viger P., Ramampiaro H., Nørvåg K., and Thu-Lan Dam - Efficient high utility itemset mining using buffered utility-lists, Applied Intelligence **48** (7) (2018) 1859–1877. https://doi.org/10.1007/s10489-017-1057-2.

14. Tseng V. S., Wu C. W., Fournier-Viger P., and Yu P. S. - Efficient Algorithms for Mining Top-K High Utility Itemsets, IEEE Trans Knowl Data Eng **28** (1) 54–67. https://doi.org/10.1109/TKDE.2015.2458860.
15. Tseng V. S., Wu C. W., Shie B. E., and Yu P. S. - UP-Growth: An efficient algorithm for high utility itemset mining, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2010) 253–262. https://doi.org/10.1145/1835804.1835839.
16. Ryang H., and Yun U. - Top-k high utility pattern mining with effective threshold raising strategies, Knowl Based Syst **76** (2015) 109–126. https://doi.org/10.1016/j.knosys.2014.12.010.
17. Quang-Huy Duong, Liao B., Fournier-Viger P., and Thu-Lan Dam - An efficient algorithm for mining the top- k high utility itemsets, using novel threshold raising and pruning strategies, Knowl Based Syst **104** (2016) 106–122. https://doi.org/10.1016/j.knosys.2016.04.016.
18. Liu J., Zhang X., Fung B. C. M., Li J., and Iqbal F. - Opportunistic mining of top-n high utility patterns, Inf Sci (N Y) **441** (2018) 171–186. https://doi.org/10.1016/ j.ins.2018.02.035.
19. Loan T.T. Nguyen, Phuc Nguyen, Trinh D.D. Nguyen, Bay Vo, Fournier-Viger P., and Tseng V. S. - Mining high-utility itemsets in dynamic profit databases, Knowl Based Syst **175** (2019) 130–144. https://doi.org/10.1016/j.knosys.2019.03.022.
20. Fournier-Viger P., Gomariz A., Gueniche T., Soltani A., Wu C. W., and Tseng V.S. - SPMF: a Java open-source pattern mining library, The Journal of Machine Learning Research **15** (2014) 3389–3393.

ABSTRACT

ENHANCE THE EFFICIENCY OF MINING TOP N HIGH UTILITY ITEMSETS IN TRANSACTION DATABASE

Vu Van Vinh, Manh Thien Ly, Duong Thi Mong Thuy,
Nguyen Thi Thanh Thuy, Nguyen Van Le, Lam Thi Hoa Mi*
Ho Chi Minh City University of Industry and Trade
*Email: *milth@huit.edu.vn*

The problem of mining high utility itemsets has many practical applications. However, in the mining process, the minimum utility threshold must be determined in advance, leading to difficulties for users. To solve this problem, many studies have been proposed to replace the determination of the threshold value by giving a positive integer N in top-N high utility itemset (top-N HUI) mining. In this study, the authors propose the topNEFIM algorithm to efficiently exploit the top-N HUI in the transaction database and the RTU threshold- strategy to automatically specify the initial threshold value. The algorithm uses a global priority queue (priorityQueue) structure to optimize the threshold-raising process. In addition, the authors also propose a data structure called ExtentionP_set to limit the database being browsed many times during top-N HUI mining. The experimental results show that the proposed algorithm has better execution time on both dense and sparse databases when it is compared with the two algorithms TKEH and THUI.

Keywords: High utility itemset, top N high utility itemsets, threshold-raising strategy, data mining.