KHAI THÁC TOP-K TẬP HỮU ÍCH CAO CÓ TƯƠNG QUAN TRÊN CƠ SỞ DỮ LIỆU GIAO DỊCH

Mạnh Thiên Lý*, Nguyễn Thị Thanh Thủy, Nguyễn Văn Lễ
Trường Đại học Công nghiệp Thực phẩm TP.HCM
*Email: *lymt@hufi.edu.vn*
Ngày nhận bài: 10/6/2022; Ngày chấp nhận đăng: 31/10/2022

TÓM TẮT

Trong bài báo này, nhóm tác giả đề xuất một hướng nghiên cứu mới, đó là khai thác top-k tập hữu ích cao có tương quan (Top-k Correlated High Utility Itemset –TCHUI) trên cơ sở dữ liệu giao dịch. Kết hợp giữa bài toán khai thác tập hữu ích cao có tương quan và bài toán khai thác top-k nhằm tìm top-k tập mặt hàng có tính tương quan mà có độ hữu ích cao trong cơ sở dữ liệu giao. Để tìm kiếm tập $TCHUI$, chúng tôi kết hợp khai thác tập hữu ích cao có tương quan (Correlated High Utility Itemset - CoHUI) với các chiến lược nâng ngưỡng và đề xuất thuật toán TCH. Thuật toán này sử dụng cấu trúc dữ liệu Utility List để lưu trữ thông tin về độ hữu ích của các tập mặt hàng, sử dụng độ đo $Kulc$ để đo lường tính tương quan và áp dụng các chiến lược tỉa: *U-Prune, TWU-Prune, LA-Prune* giúp giảm không gian tìm kiếm. Đồng thời, các chiến lược nâng ngưỡng như *RIU, LIU-E, RUC* cũng được sử dụng để khai thác tập $TCHUI$ một cách hiệu quả. Thực nghiệm trên các bộ dữ liệu lớn gồm Chess, Mushroom, Retail, Chainstore và so sánh hiệu suất thực thi giữa thuật toán TCH với thuật toán gần đây là $THUI$. Kết quả thực nghiệm cho thấy thuật toán đề xuất TCH có hiệu suất thực thi tốt hơn thuật toán $THUI$ về thời gian thực thi và bộ nhớ sử dụng.

Từ khóa: Tập hữu ích cao, tính tương quan, top-k, cơ sở dữ liệu giao dịch, tập hữu ích cao có tương quan, top-k tập hữu ích cao có tương quan.

1. GIỚI THIỆU

Theo Kotler và Keller, nghiên cứu hành vi khách hàng là nghiên cứu hành vi lựa chọn, mua sắm, sử dụng sản phẩm của những cá nhân, tổ chức nhằm làm thỏa mãn nhu cầu của họ [1]. Hiểu được hành vi khách hàng, nhà quản lý có thể phân tích, lựa chọn và đưa ra các chiến lược phù hợp trong việc nhập kho, phân phối, sắp xếp sản phẩm để mang lại lợi nhuận cao nhất cho doanh nghiệp. Để tìm kiếm thông tin hữu ích từ hành vi mua hàng của khách, khai thác tập hữu ích cao (High utility itemsets mining - HUIM) là một trong các hướng nghiên cứu được sử dụng phổ biến trong những năm gần đây. Mặc dù các thuật toán khai thác HUIs đã mang lại hiệu quả trong nhiều lĩnh vực, tuy nhiên nhược điểm của nó là chưa xem xét tới các ràng buộc trong kết quả nhận được.

Gần đây, trong các nghiên cứu về HUI có ràng buộc thì khai thác tập hữu ích cao có tương quan (Correlated High Utility Itemset - CoHUI) là hướng nghiên cứu được nhiều tác giả quan tâm. Theo đó, khai thác CoHUIs là việc khám phá các tập mặt hàng có độ hữu ích cao mà vẫn đảm bảo tính tương quan chặt chẽ giữa các mặt hàng. Các thuật toán khai thác CoHUI gần đây nhất là CoHUIM [2], CoUPM [3]. Ngoài ra, một hướng tiếp cận khác cũng được nhiều nhà nghiên cứu quan tâm, đó là khai thác top-k tập hữu ích cao (Top-k High Utility Itemset – Top-k HUI). Mục đích của các nghiên cứu này là tìm ra top k tập mặt hàng mang lại độ hữu ích cao nhất, từ đó nhà quản lý có thể nắm bắt được tình hình kinh doanh tại cửa hàng

và đề xuất chiến lược bán hàng hợp lý. Một số thuật toán khai thác top-k HUIs nổi bật có thể kể đến như TKU và TKO [4], kHMC [5], THUI [6].

Tuy nhiên, cho đến nay, chưa có nghiên cứu nào thực hiện khai thác top-k tập mặt hàng vừa có độ hữu ích cao vừa có tương quan chặt chẽ giữa các mặt hàng. Trong bài báo này, chúng tôi xây dựng thuật toán TCH để khai thác top-k tập hữu ích cao có tương quan trên cơ sở dữ liệu giao dịch.

Những đóng góp chính của bài báo:

a) Áp dụng cấu trúc dữ liệu UList để lưu trữ dữ liệu trong quá trình khai thác tập hữu ích cao có tương quan trên cơ sở dữ liệu giao dịch.

b) Áp dụng nhiều chiến lược tỉa để giảm không gian tìm kiếm như: *TWU-Prune, LA-Prune, Kulc-Prune*.

c) Áp dụng các chiến lược nâng ngưỡng để khai thác top-k hiệu quả như: *RIU, LIU-E, RUC*.

d) Kết quả thực nghiệm so sánh với thuật toán $THUI$ cho thấy thuật toán TCH có thời gian thực hiện nhanh hơn thuật toán $THUI$ trên các cơ sở dữ liệu như Chess, Mushroom, Retail và Chainstore.

Cấu trúc bài báo được chia làm 5 phần. Phần 1 giới thiệu về bài báo; Phần 2 trình bày các công trình liên quan; Phần 3 trình bày các đinh nghĩa và ký hiệu; Phần 4 trình bày thuật toán đề xuất; Phần 5 trình bày kết quả thực nghiệm và đánh giá; Phần 6 trình bày kết luận và hướng phát triển.

2. CÁC CÔNG TRÌNH LIÊN QUAN

Khai thác tập hữu ích cao (HUI) đã được nghiên cứu rộng rãi và có nhiều ứng dụng trong thực tế để tìm ra tập các mặt hàng mang lại lợi nhuận cao, thỏa một ngưỡng tối thiểu cho trước do người dùng quy định. Các thuật toán khai thác HUI ban đầu được thực hiện trên hai pha như: Two-Phase [7], CTU-Mine [8], TWU-Mining [9], UP-Growth [10], UP-Growth+ [11]. Nhược điểm của những thuật toán hai pha này là tiêu tốn khá nhiều thời gian và bộ nhớ. Do đó, các nghiên cứu về sau đã đề xuất các thuật toán khai thác HUI chỉ trên một pha nhằm rút ngắn thời gian và bộ nhớ một cách hiệu quả. Các thuật toán gần đây nhất phải kể đến là: HUI-Miner [12], FHM [13], EFIM [14], HMiner [15], MAHI [16].

Mặc dù việc khai thác tập hữu ích cao có thể giúp cho các nhà bán lẻ xác định được tập các mặt hàng mang lại lợi nhuận cao. Tuy nhiên, trong thực tế, một tập mặt hàng như {muối, iphone13} vẫn có thể là một tập hữu ích cao, do các nghiên cứu trước đây không xem xét đến tính tương quan giữa các mặt hàng trong một HUI. Để giải quyết vấn đề này, các nghiên cứu về khai thác tập hữu ích cao có tương quan đã được đề xuất. Năm 2018, Fournier-Viger và cộng sự [17] đã trình bày khái niệm "Correlation" dựa trên ba độ đo any-confidence, all-confidence và bond và đề xuất thuật toán CHIs để khai thác tập hữu ích cao có tương quan trên cơ sở dữ liệu giao dịch. Cũng vào năm này, một thuật toán khác được đề xuất dựa trên độ đo Kulc là thuật toán CoHUIM [2], do Lin và cộng sự thực hiện. Thuật toán khai thác CoHUI mới nhất là thuật toán CoUPM, do nhóm nghiên cứu của Wensheng Gan [3] đề xuất vào năm 2019. Thuật toán CoUPM sử dụng cấu trúc dữ liệu Utility-list để lưu trữ thông tin về độ hữu ích của các tập mặt hàng, kết hợp nhiều chiến lược tỉa khác nhau trong quá trình khai thác CoHUI. Kết quả cho thấy thuật toán này hiệu quả hơn so với thuật toán CoHUIM được đề xuất trước đó. Năm 2021, Mohammed Ali Fouad và cộng sự [18] nghiên cứu ứng dụng kỹ thuật phân cụm nhằm phát hiện ra các mặt hàng tương quan tiềm năng và áp dụng thuật toán khai thác mẫu cho mỗi cụm để tìm ra các tập mặt hàng tương quan và có lợi nhuận.

Một hạn chế khác trong bài toán khai thác tập hữu ích cao là khó xác định chính xác ngưỡng độ hữu ích tối thiểu thích hợp. Giá trị ngưỡng độ hữu ích nhỏ sẽ tạo ra một số lượng rất lớn các HUI không cần thiết, ngược lại giá trị ngưỡng độ hữu ích quá cao sẽ tạo ra số lượng HUI quá ít. Chính vì vậy, các thuật toán về khai thác top-k tập hữu ích cao (top-k HUI) ra đời. Các thuật toán này cho phép người dùng chỉ định số lượng tập hữu ích cao mà họ muốn lấy thông qua tham số k thay vì sử dụng độ hữu ích tối thiểu. Năm 2016, Q-H. Duong và cộng sự đã đề xuất thuật toán kHMC [5] để khai thác top-k tập hữu cao. Thuật toán sử dụng cấu trúc dữ liệu Utility-List với các chiến lược sinh ngưỡng độ hữu ích tối thiểu ban đầu như: RIU (Real Item Utilities), CUD (Co-occurrence with Utility Descending order) và COV (COVerage with utility descending order). Ngoài ra, trong quá trình khai thác, thuật toán còn áp dụng các chiến lược tỉa: EUCPT (với cấu trúc EUCS), TEP (Transitive Extension Pruning) và EA (Early Abandoning). Năm 2018, Kuldeep Singh và cộng sự đã đề xuất thuật toán TKEH [19] để khai thác top-k HUI một cách hiệu quả hơn. Trong thuật toán này, ngoài việc sử dụng ba chiến lược RIU, CUD và COV để tăng ngưỡng độ hữu ích tối thiểu, nhóm tác giả đã sử dụng kỹ thuật chiếu cơ sở dữ liệu, trộn giao dịch và hai chiến lược cắt tỉa là EUCP và Sub-tree utility để giảm không gian tìm kiếm trong quá trình khai thác top-k. Thuật toán gần đây nhất để khai thác top-k là THUI [6], do Srikumar Krishnamoorthy đề xuất vào năm 2019. Điểm nổi bật của thuật toán này là tác giả đã sử dụng bốn chiến lược nâng ngưỡng là RIU, RUC, LIU-E, LIU-LB để giảm đáng kể không gian tìm kiếm, kết quả thực nghiệm cho thấy thuật toán này hiệu quả hơn các thuật toán trước đó như TKO, kHCM. Bên cạnh các nghiên cứu được đề xuất để tìm ra các tập hữu ích cao top-k khi tập mặt hàng có độ hữu ích dương, năm 2021, Sun và cộng sự [20] cũng đề xuất một nghiên cứu mới về khai thác tập các mặt hàng hữu ích cao top-k với độ hữu ích âm.

1. CÁC ĐỊNH NGHĨA VÀ KÝ HIỆU

Cho $I = \{x_1, x_2, \ldots, x_m\}$ là tập hợp m mặt hàng khác nhau trong cơ sở dữ liệu giao dịch $D = \{T_1, T_2, \ldots, T_n\}$ gồm n giao dịch T_j $(j = 1..n)$, với $\forall T_j \in D$, $T_j = \{x_k | k = 1, 2, \ldots, N_j, x_k \in I\}$, trong đó: N_j là số lượng các mặt hàng trong giao dịch T_j. Mỗi mặt hàng x_i trong I có một giá trị lợi nhuận là $P(x_i)$. Số lượng mỗi mặt hàng x_k được mua trong giao dịch T_j là $Q(x_k, T_j)$. Xét ví dụ về cơ sở dữ liệu giao dịch D trong Bảng 1 với lợi nhuận của các mặt hàng được cho trong Bảng 2.

Bảng 1. Cơ sở dữ liệu giao dịch D

TID	Giao dịch (T)	Số lượng mua (Pq)	Độ hữu ích (U)	Độ hữu ích giao dịch (TU)
1	a, b, e, f	4, 4, 3, 1	16, 16, 3, 3	38
2	c, d, e	2, 2, 3	10, 4, 3	17
3	a, b, c, e	3, 4, 6, 2	12, 16, 30, 2	60
4	c, e, f	2, 4, 3	10, 4, 9	23
5	a, b, d, e, f	4, 5, 2, 5, 1	16, 20, 4, 5, 3	48
6	a, b, g	2, 3, 1	8, 12, 2	22
7	b, c, e, g	6, 1, 5, 2	24, 5, 5, 4	38
8	a, b, c, d, e, f	3, 2, 6, 5, 4, 6	12, 8, 30, 10, 4, 18	82

Bảng 2. Lợi nhuận của các mặt hàng

Mặt hàng	a	b	c	d	e	f	g
Lợi nhuận	4	4	5	2	1	3	2

Tần số xuất hiện của một tập mặt hàng (itemset) X trong D, ký hiệu: $SUP(X)$, là số lượng các giao dịch T_j trong cơ sở dữ liệu D có chứa X. Ví dụ: trong Bảng 1, $SUP(b) = 6$ và $SUP(bde) = 2$.

Độ hữu ích của một mặt hàng x_i trong một giao dịch T_j, ký hiệu: $U(x_i, T_j)$. Ví dụ: độ hữu ích của mặt hàng $\{b\}$ trong giao dịch T_1 được tính: $u(b, T_1) = P(b) * Q(b, T1) = 4 * 4 = 16$. Độ hữu ích của một tập mặt hàng $X = \{x_1, x_2, \dots, x_k\}$ trong giao dịch T_j, ký hiệu: $U(X, T_j)$ và được định nghĩa: $U(X, T_j) = \sum_{x_i \in X} U(x_i, T_j)$. Ví dụ: $U(c, T_3) = P(c) * Q(c, T_3) = 5 * 6 = 30$ và $U(ce, T_3) = 32$. Độ hữu ích của một tập mặt hàng X trong cơ sở dữ liệu giao dịch D, ký hiệu: $U(X)$ và được định nghĩa: $U(X) = \sum_{X \subseteq T_j \wedge T_j \in D} U(X, T_j)$. Ví dụ: $U(a) = U(a, T_1) + U(a, T_3) + U(a, T_5) + U(a, T_6) + U(a, T_8) = 64$ và $U(bcd) = 51$. Độ hữu ích của một giao dịch T_j trong cơ sở dữ liệu D, ký hiệu: $TU(T_j)$ và được định nghĩa: $TU(T_j) = \sum_{x_i \in T_j} U(x_i, T_j)$. Ví dụ: $TU(T_1) = U(a, T_1) + U(b, T_1) + U(e, T_1) + U(f, T_1) = 38$ và $TU(T_2) = 17$.

Độ hữu ích trọng số giao dịch của một tập mặt hàng X trong cơ sở dữ liệu D được kí hiệu là $TWU(X)$ và được định nghĩa: $TWU(X) = \sum_{X \subseteq T_j \in D} TU(T_j)$. Ví dụ: $TWU(cd) = TU(T_2) + TU(T_8) = 99$ và $TWU(acd) = 82$.

Dựa theo việc sắp xếp tăng dần theo tần số xuất hiện của các mặt hàng trong cơ sở dữ liệu D để xây dựng một thứ tự toàn phần $\prec$. Trong cơ sở dữ liệu ở Bảng 1, thứ tự sau khi sắp xếp toàn phần các mặt hàng là: $g \prec d \prec f \prec a \prec c \prec b \prec e$. Thứ tự SUP của các mặt hàng sau khi sắp tăng dần được thể hiện trong Bảng 3 và Bảng 4 thể hiện cơ sở dữ liệu sau khi sắp tăng dần theo tần số xuất hiện.

Bảng 3. Tần số xuất hiện của các mặt hàng sau khi được sắp tăng dần

Item	*g*	*d*	*f*	*a*	*c*	*b*	*e*
SUP	2	3	4	5	5	6	7

Bảng 4. Cơ sở dữ liệu sau khi sắp tăng dần theo tần số xuất hiện

TID	Giao dịch (T)	Số lượng mua (Pq)	Độ hữu ích (U)	Độ hữu ích giao dịch (TU)
1	f, a, b, e	1, 4, 4, 3	3, 16, 16, 3	38
2	d, c, e	2, 2, 3	4, 10, 3	17
3	a, c, b, e	3, 6, 4, 2	12, 30, 16, 2	60
4	f, c, e	3, 2, 4	9, 10, 4	23
5	d, f, a, b, e	2, 1, 4, 5, 5	4, 3, 16, 20, 5	48
6	g, a, b	1, 2, 3	2, 8, 12	22
7	g, c, b, e	2, 1, 6, 5	4, 5, 24, 5	38
8	d, f, a, c, b, e	5, 6, 3, 6, 2, 4	10, 18, 12, 30, 8, 4	82

Tập tất cả các mặt hàng sau X trong T_j, được ký hiệu là $T_j|X$. Ví dụ: trong Bảng 4, $T_3|\{ab\} = \{e\}$ và $T_1|\{a\} = \{b, e\}$. **Độ hữu ích sau** của tập mục X trong một giao dịch T_j, ký hiệu: $RU(X, T_j)$, là tổng độ hữu ích của tất cả các mục sau X trong T_j, và được định nghĩa: $RU(X, T_j) = \sum_{x_i \in (T_j|X)} U(x_i, T_j)$. Ví dụ: $RU(a, T_1) = U(b, T_1) + U(e, T_1) = 19$.

Một số định nghĩa về độ tương quan giữa các mặt hàng trong tập mặt hàng để khai thác tập hữu ích cao có tính tương quan:

Định nghĩa 1. Sự tương quan giữa các mặt hàng trong một tập mặt hàng $X = \{x_1, x_2, \dots, x_k\}$ được định nghĩa là $kulc(X) = \frac{1}{k}\left(\sum_{x_i \in X} \frac{SUP(X)}{SUP(x_i)}\right)$, với $0 \leq kulc(X) \leq 1$ [2].

Ví dụ: $kulc(ab) = \frac{1}{2}\left(\frac{SUP\ (ab)}{SUP(a)} + \frac{SUP\ (ab)}{SUP\ (b)}\right) = \frac{1}{2}\left(\frac{5}{5} + \frac{5}{6}\right) = 0{,}917$

Xét thứ tự toàn phần $x_1 \prec x_2 \prec \cdots \prec x_k \prec x_{k+1}$ của các mặt hàng trong cơ sở dữ liệu D, khi đó, giá trị $kulc$ thỏa tính chất bao đóng giảm (*downward closure*) là $kulc(x_1, \dots, x_{k+1}) \leq kulc(x_1, \dots, x_k)$ [2].

Định nghĩa 2. Tập mặt hàng X được gọi là tập hữu ích cao có tương quan (*Correlated High Utility Itemset - CoHUI*) nếu $kulc(X) \geq minCor$ và $U(X) \geq minUtil$. Trong đó minUtil là giá trị ngưỡng độ hữu ích tối thiểu và minCor là giá trị ngưỡng tương quan tối thiểu.

$$CoHUIs = \{X \subseteq I \mid U(X) \geq minUtil \wedge kulc(X) \geq minCor\}.$$

Ví dụ, trong Bảng 4, $U(de) = 30, kulc(de) = 0{,}714$. Với $minUtil = 50$ và $minCor = 0{,}5$ thì tập mục $\{de\}$ là một CoHUI.

Việc xác định ngưỡng độ hữu ích tối thiểu thường phụ thuộc vào kinh nghiệm của người dùng, số CoHUI nhận được có thể quá lớn hoặc quá bé nên không phù hợp với nhu cầu sử dụng. Trong một số trường hợp, các nhà quản lý cần chỉ định số lượng tập hữu ích cao có tương quan mà họ muốn nhận được thông qua tham số k thay vì sử dụng độ hữu ích tối thiểu. Vì vậy, việc khai thác top-k tập hữu ích cao có tương quan (Top-k CoHUI) được nghiên cứu để giải quyết vấn đề trên. Trong bài toán khai thác Top-k CoHUI, ngưỡng độ hữu ích tối thiểu sẽ thay đổi trong quá trình khai thác CoHUI để đạt được ngưỡng độ hữu ích tối ưu.

Định nghĩa 3. Độ hữu ích tối ưu là độ hữu ích cao thứ K của tập các mặt hàng thỏa CoHUI, ký hiệu: minUtil và được định nghĩa:

$$minUtil = min\{U(X) | X \in Top - k\ CoHUI\}$$

Định nghĩa 4. Tập mặt hàng X được gọi là một Top-k tập hữu ích cao có tương quan (*Top-k Correlated High Utility Itemsets – TCHUI*) nếu $kulc(X) \geq minCor$ và $U(X) \geq minUtil$. Trong đó $minUtil$ là giá trị ngưỡng độ hữu ích tối ưu và $minCor$ là giá trị ngưỡng tương quan tối thiểu.

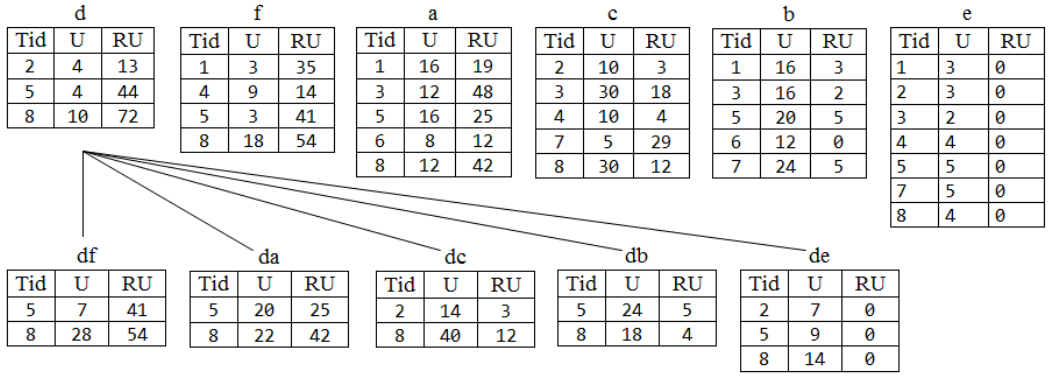
$$TCHUIs = \{X \subseteq I \mid U(X) \geq minUtil \wedge kulc(X) \geq minCor\}$$

Ví dụ, trong Bảng 4, $U(abe) = 130, kulc(abe) = 0{,}679$. Với $k = 3$, $minUtil = 80$ và $minCor = 0{,}5$ thì tập mục $\{abe\}$ là một tập $TCHUI$.

4. THUẬT TOÁN ĐỀ XUẤT

4.1. Cấu trúc dữ liệu UList

Thuật toán TCH sử dụng cấu trúc dữ liệu $UList$ [12] để biểu diễn các tập mặt hàng phát sinh trong quá trình khai thác. Cấu trúc $UList$ của một tập mặt hàng X gồm 3 thành phần là Tid, U và RU. Trong đó, Tid: thứ tự của giao dịch chứa X, U: độ hữu ích của tập mặt hàng X trong giao dịch Tid, RU: độ hữu ích của tất cả các mặt hàng sau X trong giao dịch Tid. Hình 1 trình bày danh sách các $UList$ 1 và 2 phần tử sau khi đã loại bỏ các mặt hàng x có $TWU(x) < minUtil$ và sắp theo thứ tự toàn phần. Trong đó, $minUtil$ được xác định dựa vào chiến lược tăng ngưỡng RIU (trình bày trong phần 4.3). Xét danh sách $UList(d)$ gồm 3 bộ tương ứng với các giao dịch T_2, T_5, T_6. Bộ T_2 có $U(d, T_2) = 4$, $RU(d, T_2) = 13$; bộ T_5 có $U(d, T_5) = 4$, $RU(d, T_5) = 44$; bộ T_8 có $U(d, T_8) = 10$, $RU(d, T_8) = 72$. Tương tự cho các $UList$ khác.



Hình 1. UList của các tập mặt hàng 1 phần tử và 2 phần tử

4.2. Các chiến lược tỉa

Trong bài toán khai thác tập hữu ích cao có tương quan, ngoài cấu trúc dữ liệu lưu trữ thì các chiến lược tỉa cũng có ảnh hưởng lớn đến hiệu suất thực thi của thuật toán. Việc áp dụng các chiến lược tỉa giúp thu hẹp không gian tìm kiếm làm tối ưu thời gian thực thi và bộ nhớ sử dụng. Trong bài báo này, chúng tôi sử dụng các chiến lược tỉa như: U-Prune [12], LA-Prune [21], EUCS-Prune [13], Kulc-Prune [2] để tăng hiệu suất thực thi của thuật toán.

Chiến lược 1 (U-Prune): Nếu tổng độ hữu ích của tập mặt hàng X và tổng độ hữu ích của các mặt hàng sau X nhỏ hơn $minUtil$ thì mọi tập mở rộng từ X đều không phải là $TCHUI$. Nghĩa là nếu $U(X) + RU(X) < minUtil$ thì $\forall\, Y \supseteq X,\, Y \notin TCHUI$. Khi đó ngừng mở rộng với tập mặt hàng X.

Chiến lược 2 (LA-Prune): Xét 2 tập X và Y, nếu $U(X) + RU(X) - \sum_{X \subseteq T_j \in D \wedge Y \not\subset T_j} U(X, T_j) + RU(X, T_j) < minUtil$ thì tập mặt hàng XY không phải là $TCHUI$ và mọi tập mở rộng từ XY cũng không phải là $TCHUI$ (nghĩa là $\forall\, Z \supseteq XY$ thì Z không phải là $TCHUI$).

Chiến lược 3 (EUCS-Prune): Xét X, Y là hai tập mặt hàng 1 phần tử, Nếu $TWU(X, Y) < minUtil$ thì tập mặt hàng XY không phải là $TCHUI$ và mọi tập mở rộng từ XY cũng không phải là $TCHUI$. Khi đó dừng mở rộng với tập mục XY.

Chiến lược 4 (Kulc-Prune): Nếu $kulc(X) < minCor$ thì mọi tập mở rộng từ X không phải lài một $TCHUI$. Giả sử rằng y là phần tử mở rộng từ tập X để có tập Xy, theo tính chất 2, ta có $kulc(Xy) < kulc(X) < minCor$. Do đó mọi tập mở rộng từ X không phải là $TCHUI$.

4.3. Các chiến lược nâng ngưỡng

Chiến lược RIU (Real Item Utilities): Khi quét cơ sở dữ liệu giao dịch lần đầu sẽ tính được độ hữu ích của từng mặt hàng. Mỗi giá trị độ hữu ích của mặt hàng x ký hiệu là $RIU(x)$. Tập tất cả các RIU ký hiệu là $R = \{riu_1, riu_2.., riu_m\}$. Giá trị khởi đầu của $minUtil$ được đặt $minUtil = riu_k$ là giá trị lớn nhất thứ k trong R (thay vì giá trị 0) [5, 6]

Chiến lược LIU-E (Leaf itemset utility exact): Chiến lược này được áp dụng để sinh ngưỡng $minUtil$ ban đầu trước quá trình khai thác tập $TCHUI$. Mỗi phần tử LIU được định nghĩa $LIU(x_i, x_j) = \sum_{X=\{x_i, x_{i+1}, x_{i+2} .. x_j\} \subset T_s \in D} U(X, T_s)$ [6]

Xét $PQ\_LIU = \{LIU(x, y) \,|LIU(x, y) > minUtil\}$. Nếu $PQ\_LIU > K$ thì $minUtil = PQ\_LIU_K$ (nghĩa là giá trị $minUtil$ sẽ được nâng giá trị lên thành PQ_LIU_K.

Chiến lược RUC (Raising threshold by Utility of Candidates) [4]: RUC thực hiện cập nhật K tập mặt hàng trong tập kết quả $TCHUI$ trong quá trình khai thác bằng cách thay thế tập mặt

hàng có độ hữu ích thấp nhất trong $TCHUI$ bằng tập mặt hàng có độ hữu ích cao hơn. Xét $TCHUI = \{X_1, X_2, \dots X_k\}$. Nếu tồn tại tập mặt hàng Y có $U(Y) > U(X_i)$ và $Kulc(Y) > Kulc(X_i)$ thì thay thế $TCHUI = (TCHUI - \{X_i\}) \cup Y$.

4.4. Thuật toán TCH

Trong bài báo này, nhóm tác giả đề xuất thuật toán TCH (Top-K Correlated High Utility Itemsets để khai thác K tập hữu ích cao nhất có tính tương quan. Thuật toán chính TCH có dữ liệu đầu vào là cơ sở dữ liệu giao dịch D, giá trị ngưỡng tương quan tối thiểu $minCor$ và K: số tập mặt hàng cần khai thác. Kết quả thực hiện của thuật toán là top-K tập hữu ích cao có tính tương quan. Thuật toán thực hiện quét cơ sở dữ liệu D ở bước đầu tiên để tính các giá trị $SUP(i), TWU(i), U(i)$ cho mỗi mặt hàng i có trong cơ sở dữ liệu. Tại dòng 2 và 3, thuật toán áp dụng các chiến lược nâng ngưỡng là RIU và $LIU\text{-}E$ để xác định giá trị $minUtil$ ban đầu. Với giá trị minUtil vừa tính được, tại dòng 4, tính tập I^* bằng cách loại bỏ các mặt hàng i có $TWU(i) < minUtil$ khỏi tập I, đồng thời loại mặt hàng i khỏi cơ sở dữ liệu D. Sau đó sắp xếp tập I^* tăng dần theo SUP và sắp các mặt hàng trong D theo thứ tự của I^* (dòng 5). Tại dòng 6, khởi tạo giá trị cho cấu trúc $EUCS$. Dòng 7 tạo danh sách các $UList$ cho mỗi phần tử $i \notin I^*$ bằng cách quét D lần 2, sau đó gọi thuật toán $MiningTCHUI$ để khai thác $top\ K$ tập hữu ích cao có tính tương quan.

Thuật toán 1: *Thuật toán chính* $-$ TCH

Vào: D: Cơ sở dữ liệu giao dịch, $minCor$: Ngưỡng tương quan tối thiểu, K: Số lượng tập mặt hàng có tính tương quan và độ hữu ích cao nhất cần khai thác

Ra: $TopK$ tập mặt hàng có độ hữu ích cao có tính tương quan ($TCHUIs$)

1. Quét cơ sở dữ liệu D để tính $SUP(i), TWU(i), U(i)$ cho mỗi mặt hàng i có trong I.
2. Sinh ngưỡng $minUtil = RIU_K$ (chiến lược tăng ngưỡng RIU)
3. Sinh ngưỡng $minUtil = PQ\_LIU_K$ (chiến lược tăng ngưỡng LIU-E)
4. Tính $I^* = \{i \in I \mid TWU(i) \geq minUtil\}$, loại khỏi D các mặt hàng $i \notin I^*$.
5. Sắp xếp I^* tăng theo độ hỗ trợ SUP, sắp xếp các mặt hàng trong D theo thứ tự của I^*.
6. Khởi tạo cấu trúc $EUCS$
7. Quét cơ sở dữ liệu D để tạo danh sách *UList* cho mỗi phần tử $i \notin I^*$ là ULs
8. $MiningTCHUI(\varnothing, ULs, minUtil, minCor, K, EUCS)$

Thuật toán 2 ($MiningTCHUI$) có dữ liệu đầu vào gồm một $UList\ P$ với vai trò tiền tố, một danh sách các $UList$ là ULs có tiền tố P và các giá trị $minCor, minUtil, EUCS$. Thuật toán khai thác top-K tập hữu ích cao tương quan bằng hình thức đệ quy. Từ dòng 1 đến dòng 5, thuật toán duyệt qua tất cả các $UList\ X$ trong danh sách ULs và kiểm tra điều kiện nếu $U(X) \geq minUtil \wedge kulc(X) \geq minCor$ thì áp dụng chiến lược RUC bằng cách đưa X vào tập $TCHUIs$, cập nhật lại giá trị $minUtil$, đồng thời loại bỏ tập mặt hàng Y với $U(Y) < minUtil$ ra khỏi tập $TCHUIs$. Từ dòng 6 đến 12, kiểm tra điều kiện mở rộng tập bằng cách áp dụng chiến lược tỉa U-Prune và Kulc-Prune. Dòng 9 áp dụng chiến lược tỉa EUCS, dòng 10 gọi hàm $ULConstruct$ để kết hợp 2 UList X, Y thành $UList\ XY$. Dòng 13 thực hiện gọi đệ quy để khai thác tập $TCHUI$

Thuật toán 2: $MiningTCHUI$

Vào: P: *UList* với vai trò là tiền tố; ULs: Danh sách các *UList* có tiền tô là *UList* P, $minUtil$: Ngưỡng độ hữu ích tối thiểu, $minCor$: Ngưỡng tương quan tối thiểu, $EUCS$: Cấu trúc EUCS, K: Số lượng tập mặt hàng có tính tương quan và độ hữu ích cao nhất cần khai thác

Ra: $TopK$ tập mặt hàng có độ hữu ích cao có tính tương quan ($TCHUIs$)

1. **for** each $X \in ULs$ **do**
2. **if** $U(X) \geq minUtil \wedge kulc(X) \geq minCor$ **then**
3. $TCHUIs \leftarrow X$
4. $minUtil = min\{U(X) | X \in TCHUIs\}$ //Chiến lược RUC
5. **end if**
6. **if** $(U(X) + RU(X) \geq minUtil \wedge kulc(X) \geq minCor)$ //U-Prune và Kulc-Prune
7. $exULs = \emptyset$ //Khởi tạo danh sách *UList* mở rộng từ X
8. **for** each $Y \in ULs \wedge Y\ after\ X$ **do**
9. **if** $EUCS(X,Y) \geq minUtil$ **then** //Chiến lược tỉa EUCS
10. $exULs \leftarrow UL\text{Construct}(P, X, Y)$;
11. **end if**
12. **end for**
13. $MiningTCHUI(X,\ exULs,\ minUtil,\ minCor,\ EUCS, K)$;
14. **end if**
15. **end for**

Thuật toán 3 ($ULConstruct$) có dữ liệu đầu vào gồm 1 UList tiền tố P và 2 UList Px và Py. Thuật toán thực hiện kết hợp Px, Py thành $UList$ mới Pxy. Tại dòng 1, khởi tạo giá trị ban đầu cho $U_{LA} = U(P) + RU(P)$. Dòng 2 duyệt qua từng bộ $ex \in Px$ và kiểm tra nếu có bộ $ey \in Py$ mà $ex.tid = ey.tid$ (dòng 3) thì tạo một bộ mới là exy trong các trường hợp: nếu $UList$ tiền tố $P \neq \emptyset$ (dòng 4) nghĩa là Px, Py là các $UList$ có từ 2 phần tử trở lên. Khi đó thực hiện tìm bộ $e \in P$ sao cho $e.tid = ex.tid$ và tạo giá trị bộ $exy =< ex.tid,\ ex.u + ey.u - e.u, ey.ru >$ (dòng 5,6). Trường hợp ngược lại ($P = \emptyset$) nghĩa là Px, Py là các $UList$ một phần tử không có $UList$ tiền tố, khi đó thực hiện tạo giá trị bộ $exy =< ex.tid,\ ex.u + ey.u,\ ey.ru >$ (dòng 8). Tại dòng 11, trường hợp ngược lại điều kiện dòng 3, nghĩa là không tồn tại bất kỳ bộ ey nào thuộc Py thì giảm U_{LA} một lượng $ex.u + ex.ru$ (áp dụng chiến lược tỉa $LA - Prune$). Dòng 18 trả về kết quả là $UList\ Pxy$ được kết hợp từ hai $UList\ Px$ và Py.

Thuật toán 3: ULConstruct

Vào: P: *UList* là tiền tố; $Px,\ Py$: Hai *UList* cần kết hợp; $minUtil$: Ngưỡng độ hữu ích tối thiểu.

Ra: Pxy: *UList* sau khi kết hợp Px và Py.

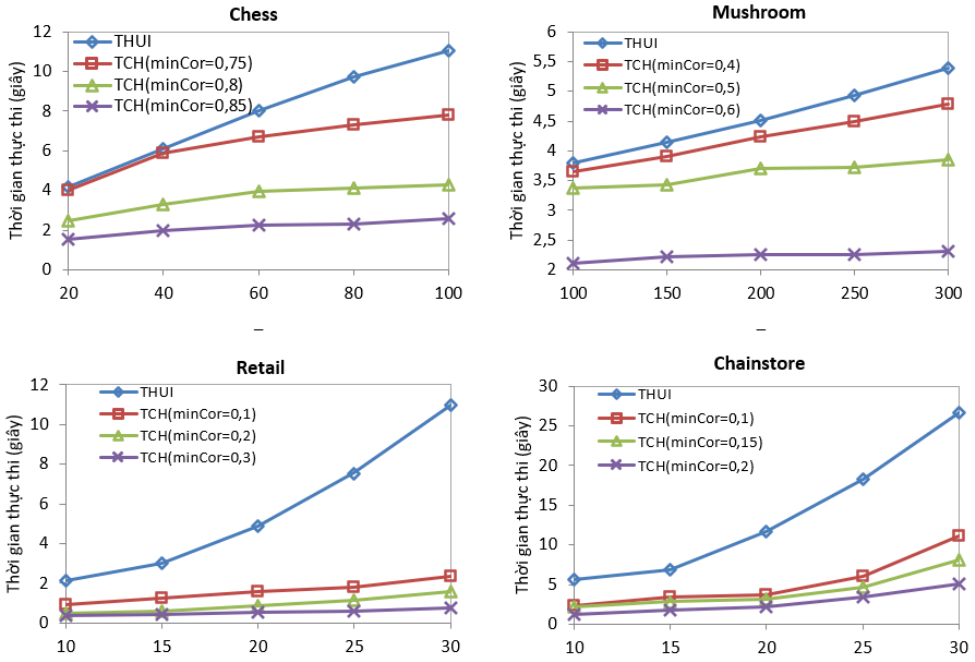
1. Đặt $U_{LA} = U(P) + RU(P)$;
2. **for** bộ $ex \in Px$ **then**
3. **if** $\exists\ ey \in Py \wedge ex.tid = ey.tid$ **then**
4. **if** $P \neq \emptyset$ **then**
5. Tìm $e \in P$ sao cho $e.tid = ex.tid$;
6. $exy =< ex.tid,\ ex.u + ey.u - e.u, ey.ru >$;
7. **else**
8. $exy =< ex.tid,\ ex.u + ey.u,\ ey.ru >$;
9. **end if**
10. $Pxy \leftarrow exy$;
11. **else**
12. $U_{LA} = U_{LA} - (ex.u + ex.ru)$;
13. **if** $U_{LA} < minUtil$ **then** // Chiến lược tỉa LA-Prune
14. return null;
15. **end if**
16. **end if**
17. **end for**
18. **return** Pxy;

5. THỰC NGHIỆM

Thuật toán TCH được cài đặt bằng ngôn ngữ Java, chạy thực nghiệm trên máy tính Dell Precision, Intel Core i3-2310 CPU @2,1GHz, bộ nhớ RAM 4GB trên hệ điều hành Windows 10. Các cơ sở dữ liệu dùng thử nghiệm được tải từ thư viện SPMF [22] gồm: Chess, Mushroom, Retail và Chainstore (Bảng 5). Kết quả thực nghiệm của thuật toán TCH được so sánh với thuật toán gần đây cùng khai thác top-K tập hữu ích cao là THUI [6] dựa trên đánh giá về thời gian thực thi và dung lượng bộ nhớ sử dụng.

Bảng 5. Đặc điểm các cơ sở dữ liệu thực nghiệm

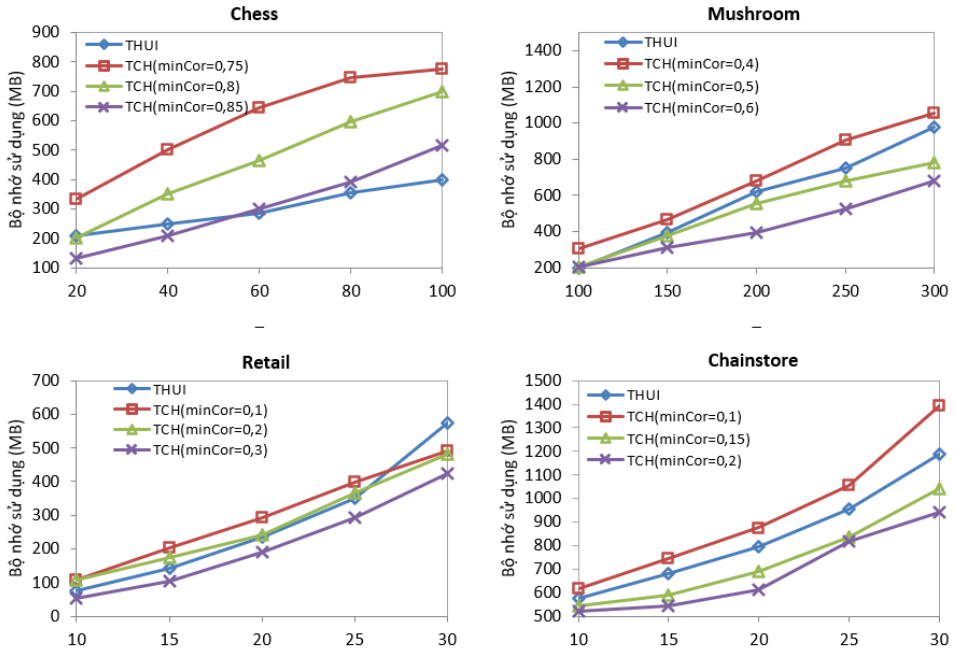
Cơ sở dữ liệu	Số lượng giao dịch	Số lượng mục (I)	Độ dài trung bình (A)	Độ dày (A/I) %
Chess	3.196	75	37	49,3333
Mushroom	8.124	119	23	19,3277
Retail	88.162	16.470	10,3	0,0625
Chainstore	1.112.949	46.086	7,3	0,0158



Hình 2. So sánh thời gian thực thi

Hình 2 trình bày so sánh hiệu suất của thuật toán đề xuất TCH và thuật toán $THUI$ dựa trên độ đo là thời gian thực thi trên 4 cơ sở dữ liệu $Chess, Mushroom, Retail$ và $Chainstore$. Với cơ sở dữ liệu có độ dày cao (Chess) và dày trung bình (Mushroom) thì thời gian thực thi của thuật toán TCH tốt hơn thuật toán $THUI$ tại mọi ngưỡng $minCor$ và chỉ số K. Cụ thể với cơ sở dữ liệu $Chess$, tại ngưỡng $minCor = 0{,}75$ và $K = 20$, thời gian thực thi của thuật toán TCH thấp hơn không đáng kể so với thuật toán THUI. Tuy nhiên, khi ngưỡng minCor tăng lên đến 0,85 thì thời gian thực hiện của thuật toán TCH thấp hơn từ 2,5 đến 4 lần nhiều so với thuật toán THUI. Với cơ sở dữ liệu Mushroom, khi chỉ số K tăng từ 100 đến 300 thì thuật toán TCH thực hiện tốt hơn thuật toán THUI tại tất cả các ngưỡng minCor. Tại $K = 100$, thuật toán THUI thực hiện 3,8 giây, trong khi thuật toán TCH thực hiện thấp nhất là 2,1 giây tại ngưỡng $minCor = 0{,}6$. Tại $K = 300$, thuật toán TCH thực hiện 4,7 giây tại ngưỡng $minCor = 0{,}4$,

3,8 giây tại ngưỡng $minCor = 0{,}5$ và 2,3 giây tại ngưỡng $minCor = 0{,}6$, trong khi thuật toán THUI thực hiện 5,4 giây. Với cơ sở dữ liệu thưa Retail, thời gian thực thi của thuật toán TCH nhanh hơn thuật toán THUI tại các ngưỡng K từ 10 đến 30. Tại ngưỡng $minCor = 0{,}3$, thuật toán TCH nhanh hơn thuật toán THUI từ 5 đến 15 lần, tại ngưỡng $minCor = 0{,}2$, thuật toán TCH nhanh hơn thuật toán THUI từ 4 đến 7 lần, với $minCor = 0{,}1$, thuật toán TCH nhanh hơn thuật toán THUI từ 2 đến 5 lần. Với cơ sở dữ liệu thưa Chainstore, thuật toán TCH có thời gian thực hiện trung bình từ 1,9 đến 8 giây tại các chỉ số K từ 10 đến 30. Trong khi thuật toán THUI có thời gian thực hiện cao hơn từ 5,5 đến 27 giây. Kết quả này cho thấy tính hiệu quả của các chiến lược tỉa và cấu trúc dữ liệu áp dụng trong thuật toán TCH đã giúp thu hẹp đáng kể không gian tìm kiếm, từ đó làm tăng hiệu suất thực thi của thuật toán.



Hình 3. So sánh bộ nhớ sử dụng

Hình 3 so sánh bộ nhớ sử dụng giữa 2 thuật toán TCH và THUI trên 4 cơ sở dữ liệu là Chess, Mushroom, Retail và Chainstore. Dữ liệu thực nghiệm cho thấy bộ nhớ sử dụng của 2 thuật toán tăng tuyến tính với độ lớn của ngưỡng K. Riêng đối với thuật toán TCH thì bộ nhớ sử dụng còn tỉ lệ nghịch với ngưỡng minCor. Với cơ sở dữ liệu Chess, thuật toán THUI sử dụng bộ nhớ tốt hơn thuật toán TCH tại các ngưỡng K. Với cơ sở dữ liệu Mushroom, thuật toán TCH sử dụng bộ nhớ ít hơn thuật toán THUI tại các ngưỡng K và ngưỡng $minCor = 0{,}5$ và 0,6. Tuy nhiên với ngưỡng $minCor = 0{,}4$, thuật toán THUI có mức độ sử dụng bộ nhớ tốt hơn thuật toán TCH với giá trị chênh lệch không đáng kể. Với cơ sở dữ liệu Retail, thuật toán TCH sử dụng bộ nhớ tốt hơn thuật toán THUI tại ngưỡng $minCor = 0{,}3$ trên các ngưỡng K từ 10 đến 30, dung lượng bộ nhớ sử dụng của thuật toán TCH từ 54 đến 423 MB trong khi thuật toán THUI sử dụng bộ nhớ từ 75 đến 574 MB. Tuy nhiên, so với thuật toán TCH tại 2 ngưỡng $minCor = 0{,}1$ và 0,2 thì thuật toán THUI có dung lượng bộ nhớ sử dụng tốt hơn tại 4 ngưỡng K đầu tiên. Với cơ sở dữ liệu Chainstore, tại 2 ngưỡng $minCor = 0{,}15$ và 0,2, thuật toán TCH có dung lượng sử dụng bộ nhớ tốt hơn so với thuật toán THUI. Cụ thể tại ngưỡng $minCor = 0{,}2$ và $K = 10$, thuật toán TCH sử dụng 519 MB so với 576 MB của thuật toán THUI. Khi K tăng lên đến 30, thuật toán TCH sử dụng 942 MB, ít hơn khoảng 246 MB so với thuật toán THUI. Kết quả thực nghiệm cho thấy giá trị ngưỡng tương quan minCor và ngưỡng

K càng cao thì thuật toán TCH sử dụng bộ nhớ càng hiệu quả hơn thuật toán THUI. Điều này chứng tỏ các chiến lược tỉa và cấu trúc dữ liệu áp dụng trong thuật toán TCH có ảnh hưởng tốt đến việc sử dụng bộ nhớ trong quá trình khai thác top-K tập hữu ích cao có tương quan.

6. KẾT LUẬN

Qua nghiên cứu này, nhóm tác giả đã đề xuất thuật toán TCH để khai thác top-k tập hữu ích cao có tính tương quan (TCHUI) trên cơ sở dữ liệu giao dịch. Thuật toán sử dụng cấu trúc dữ liệu UList với các chiến lược hiệu quả như: các chiến lược nâng ngưỡng (RIU, LIU-E, RUC) và các chiến lược tỉa (U-Prune, TWU-Prune, LA-Prune) để tối ưu hiệu suất thực thi trong quá trình khai thác. Kết quả thực nghiệm cho thấy thuật toán TCH có thời gian thực thi nhanh hơn thuật toán $THUI$ và bộ nhớ sử dụng của thuật toán TCH cũng tốt hơn thuật toán $THUI$ ở một số ngưỡng minCor cao.

Hướng phát triển tiếp theo của nghiên cứu này là cải tiến chiến lược nâng ngưỡng để tăng hiệu suất thực thi trên các cơ sở dữ liệu lớn có độ dày cao và ứng dụng trên cơ sở dữ liệu tăng trưởng.

TÀI LIỆU THAM KHẢO

1. Kotler P. and Keller K.L. - Marketing Management, Pearson (2016).
2. Gan W., Lin J. C. W., Fournier-Viger P., Chao H. C. and Fujita H. - Extracting non-redundant correlated purchase behaviors by utility measure. Knowledge-Based Systems **143** (2018) 30-41.
3. Gan W.; Lin J. C. W.; Chao H. C.; Fujita H.; Philip S. Y. - Correlated utility-based pattern mining. Information Sciences **504** (2019) 470-486.
4. Tseng, V. S.; Wu, C.-W.; Fournier-Viger, P.; Philip, S. Y. - Efficient algorithms for mining top-k high utility itemsets. IEEE Transactions on Knowledge and Data Engineering **28** (1) (2016) 54–67.
5. Duong Q.-H., Liao B., Fournier-Viger P. and Dam T.-L. - An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies. Knowledge-Based Systems **104** (2016) 106-122.
6. Krishnamoorthy S. - Mining top-k high utility itemsets with effective threshold raising strategies. Expert Systems With Applications **117** (2019) 148-165.
7. Liu Y., Liao W. K. and Choudhary A. - A two-phase algorithm for fast discovery of high utility itemsets, In Pacific-Asia Conference on Knowledge Discovery and Data Mining (2005) 689-695.
8. Erwin A., Gopalan R. P. and Achuthan N. R. - CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach. IEEE International Conference on Computer and Information Technology (2007) 71-76.
9. Le B., Nguyen H. and Vo B. - An efficient strategy for mining high utility itemsets. International Journal of Intelligent Information and Database Systems **5** (2) (2011) 164-176.

10. Tseng V. S., Wu C. W., Shie B. E. and Yu P. S. - UP-Growth: an efficient algorithm for high utility itemset mining, In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2010) 253-262.

11. Tseng V. S., Shie B. E., Wu C. W. and Philip S. Y. - Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering **25** (2012) 1772-1786.

12. 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 (2012) 55-64.

13. Fournier-Viger P., Wu C. W., Zida, S. and Tseng V. S. - FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. International Symposium on Methodologies for Intelligent Systems **8502** (2014) 83-92.

14. 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. Knowledge and Information Systems **51** (2) (2017) 595-625.

15. Krishnamoorthy S. - HMiner: Efficiently mining high utility itemsets. Expert Systems with Applications **90** (2017) 168-183.

16. Sohrabi M.K - An efficient projection-based method for high utility itemset mining using a novel pruning approach on the utility matri. Knowledge and Information Systems **62** (2020) 4141-4167.

17. Fournier-Viger P., Zhang Y., Lin J. C. W., Dinh D. T. and Le H. B. - Mining correlated high-utility itemsets using various measures. Logic Journal of the IGPL **28** (1) (2018) 19-32.

18. Fouad M. A., Hussein W., Rady S., Yu P. S. and Gharib T. F. - Correlated High Utility Itemset Mining Based on Item Decomposition, in 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt (2021).

19. 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.

20. Sun Rui, Han Meng, Zhang Chunyan, Shen Mingyao and Du Shiyu - Mining of Top-k High Utility Itemsets with Negative Utility. Journal of Intelligent & Fuzzy Systems **40** (3) (2021) 5637-5652.

21. Lin J. C. W., Gan W., Fournier-Viger P., Hong T. P. and Chao H. C. - FDHUP: Fast algorithm for mining discriminative high utility patterns. Knowledge and Information Systems **51** (2017) 873-909.

22. Fournier-Viger P., Gomariz A., Soltani A. and Lam H. - An Open-Source Data Mining Library (2014) [Online]. Available: http://www.philippe-fournier-viger.com.

ABSTRACT

MINING TOP-K CORRELATED HIGH UTILITY ITEMSETS ON TRANSACTION DATABASE

Manh Thien Ly*, Nguyen Thi Thanh Thuy, Nguyen Van Le
Ho Chi Minh City University of Food Industry
*Email: *lymt@hufi.edu.vn*

In this paper, we propose a new research direction, which is to exploit the top-k Correlated High Utility Itemset (TCHUI) on the transaction database. Combination of mining Correlated High Utility Itemsets and mining Top-K High Utility Itemsets to find the top-k correlated high utility itemsets on the transaction database. To address this issue, we combine the Correlated High Utility Itemset (CoHUI) mining with the threshold raising strategies and propose the TCH algorithm. This algorithm uses the Utility List data structure to store data about the utility of itemsets, uses the Kulc threshold to measure correlation, and applies several pruning strategies such as U-Prune, TWU -Prune, LA-Prune to reduce the search space. Besides that, the threshold raising strategies are applied such as RIU, LIU-E, and RUC to exploit the TCHUI set effectively. Our experiment results on large datasets including Chess, Mushroom, Retail, and Chainstore and compare with the state-of-the-art THUI algorithm. The results show that the proposed algorithm has better performance than the THUI algorithm in terms of execution time and memory usage.

Keywords: High utility itemsets, correlation, top-k, transaction database, correlated high utility itemsets, top-k correlated high utility itemsets.