Nâng cao hiệu quả khai phá tập hữu ích cao bằng giải pháp chiếu ngược P-set

Võ Đình Bảy[1*], Nguyễn Tấn Phúc[2]

[1]Khoa Công nghệ thông tin, Trường Đại học Công nghệ TP Hồ Chí Minh
[2]Trung tâm Ngoại ngữ - Tin học, Trường Đại học Khánh Hòa

Ngày nhận bài 3/7/2017; ngày chuyển phản biện 7/7//2017; ngày nhận phản biện 4/8/2017; ngày chấp nhận đăng 10/8/2017

Tóm tắt:

Trong khi khai phá tập phổ biến chỉ quan tâm đến sự xuất hiện của các mục trong giao dịch (nghĩa là chúng có hay không có trong các giao dịch) thì khai phá tập hữu ích cao (HUI - High Utility Itemset) lại quan tâm đến lợi nhuận thu được khi bán các tập mục cùng nhau. Đã có nhiều thuật toán được phát triển nhằm nâng cao hiệu quả khai phá HUI, trong đó EFIM (EFficient high-utility Itemset Mining) là thuật toán mới nhất áp dụng nhiều kỹ thuật để cải thiện tốc độ và không gian tìm kiếm. Tuy nhiên, EFIM vẫn còn tốn nhiều chi phí quét các dòng dữ liệu để xác định sự liên quan đến ứng viên đang xét làm giảm hiệu quả của thuật toán, đặc biệt là đối với cơ sở dữ liệu (CSDL) thưa. Bài báo này đề xuất giải pháp chiếu ngược P-set để giảm số lượng giao dịch cần xét trong thuật toán EFIM và vì vậy, làm giảm thời gian khai phá HUI. Một thuật toán cải tiến từ EFIM (IEFIM - Improve EFficient high-utility Itemset Mining) dựa trên P-set cũng được đề nghị. Kết quả thực nghiệm cho thấy, thuật toán IEFIM làm giảm đáng kể số lượng giao dịch cần xét và thời gian thực thi trên các CSDL thưa.

Từ khóa: ***Khai phá dữ liệu, khai phá tập hữu ích cao, tỉa ứng viên.***

Chỉ số phân loại: ***1.2***

Đặt vấn đề

Khai phá tập phổ biến (FIM - Frequent Itemset Mining) được Agrawal giới thiệu vào năm 1993 khi phân tích mô hình dữ liệu siêu thị [1], làm cơ sở để mở rộng thành các bài toán khác trong lĩnh vực khai phá dữ liệu.

Trong các nghiên cứu về thị trường, FIM trong CSDL giao dịch chính là tìm các tập (itemset) thường xuyên xuất hiện trong các giao dịch. Các thuật toán khai phá tập phổ biến thường áp dụng tính chất bao đóng giảm (downward closure property) [2] để tăng khả năng tỉa các tập ứng viên thừa. Cụ thể, nếu có một tập không phổ biến X thì thuật toán không xét các tập ứng viên chứa tập X, nghĩa là với một bộ dữ liệu chứa n phần tử và X chứa k phần tử, thuật toán sẽ không xét $2^{(n-k)}$ - 2 tập có chứa X.

Tuy nhiên, tập phổ biến chỉ quan tâm đến việc có mua hay không mua các mặt hàng mà không quan tâm đến lợi nhuận thu được đối với từng mặt hàng. Vì vậy, bài toán khai phá tập hữu ích cao được đặt ra. Chúng ta xét ví dụ như ở hình 1 về dữ liệu bán hàng [3] để hiểu rõ hơn về bài toán khai phá tập phổ biến và bài toán khai phá HUI. Trong đó, bảng (1A) là bảng chứa giá trị lợi nhuận trên từng đơn vị sản phẩm (item) và bảng (1B) chứa thông tin từng giao dịch với từng sản phẩm tương ứng với số lượng bán được trong giao dịch đó. Với khai phá tập phổ biến không quan tâm đến bảng (1A) và số lượng ở bảng (1B), nhưng tập phổ biến chưa chắc là tập có giá trị hữu ích cao. Cụ thể, độ phổ biến của {bc} là 3, hữu ích là 18, trong khi {de} có giá trị lần lượt là 2 và 22.

Item	a	b	c	d	e		g
Utility	1	2	1	5	4	3	1

(A) Bảng lợi nhuận.

Tid	Giao dịch	Số lượng
T1	{b,c,d,g}	{1,2,1,1}
T2	{a,b,c,d,e}	{4,1,3,1,1}
T3	{a,c,d}	{4,2,1}
T4	{a,b,d,e}	{5,2,1,2}
T5	{a,b,c,f}	{3,4,1,2}

(B) Bảng giao dịch.

Hình 1. Dữ liệu bán hàng.

Tương tự như tập phổ biến, một tập là HUI nếu giá trị hữu ích (chẳng hạn như lợi nhuận thu được khi bán itemset trong tất cả các giao dịch) phải đạt ngưỡng tối thiểu cho trước. Với tập hữu ích cao, tính chất bao đóng giảm không còn phù hợp, cụ thể: Các tập {a}, {ab}, {abc} có độ phổ biến lần lượt là 4, 3, 2 (thỏa mãn tính chất bao đóng giảm) nhưng giá trị hữu ích là 16, 26, 21. Nếu lấy ngưỡng là 20 thì

[*]Tác giả liên hệ: Email: bayvodinh@gmail.com

Efficient solution for mining High Utility Itemsets by reverse projection P-set

Dinh Bay Vo[1*], Tan Phuc Nguyen[2]

[1]*Faculty of Information Technology, Ho Chi Minh City University of Technology*
[2]*Foreign Languages and Informatics Center, Khanh Hoa University*

Received 3 July 2017; accepted 10 August 2017

Abstract:

Mining frequent itemsets just focuses on mining items which have the same importance (e.g., unit profit) and may not appear more than once in each transaction. On the contrary, mining High Utility Itemsets (HUIs) considers items which have different unit profits and may have non-binary purchase quantities in transactions. Basically, mining HUIs is to find the items that produce a higher profit than those bought frequently. There have been many algorithms developed for mining HUIs, among which EFIM is the latest algorithm which applies several techniques to improve the runtime and the search space. However, the cost of EFIM for scanning transactions to determine candidate relevance is high, which reduces the efficiency of the algorithm, especially on sparse databases. In this paper, the authors developed a P-set structure and proposed an improved algorithm of EFIM to reduce the number of transaction scans and thereby reduce the mining time. Experimental results showed that the improved algorithm reduced significantly the number of transaction scans and the mining time, especially on sparse databases.

Keywords: ***Data mining, high utility itemset mining, pruning candidates.***

Classification number: ***1.2***

ta chọn {ab}, {abc} và loại {a}, còn nếu lấy ngưỡng là 22 thì chỉ mỗi {ab} được chọn. Vì vậy, các phương pháp khai phá tập phổ biến không thể áp dụng vào khai phá tập hữu ích cao.

Từ khi bài toán được phát biểu vào năm 2004 [4] đến nay, đã có nhiều thuật toán khai phá tập hữu ích cao được phát triển nhằm nâng cao hiệu quả khai phá: UMining (2004) [4], UMining-H (2006) [5], Two-Phase (2005) [6], IHUP (2009) [7], TWU-Mining (2009) [8], UP-Growth (2010) [9], DTWU-Mining (2011) [10], EFIM (2015) [11] và một số hướng phát triển khác của tập hữu ích cao, điển hình như khai phá tập đóng có CHUD (2011) [12], AprioriCH, AprioriHC-D (2015) [13]; khai phá Top-k HUI có TKU (2012) [14], TKO (2016) [15]; khai phá HUI trên luồng dữ liệu có THUI-Mine (2008) [16], GUIDE (2012) [17], hay khai phá HUI trên dữ liệu không chắc chắn [18].

Trong số các thuật toán khai phá tập hữu ích cao, EFIM được xem là thuật toán nhanh nhất với nhiều giải pháp để cải thiện không gian tìm kiếm và thời gian như kỹ thuật chiếu trên CSDL (Database Projection), trộn các giao dịch (Transaction Merging) và tính lại biên cận trên. Mặc dù cải thiện đáng kể về thời gian khai phá và bộ nhớ sử dụng so với các thuật toán trước đó (UP-Growth, HUI-Miner [3], UP-Growth [19], HUP-Miner [20]) nhưng EFIM vẫn quét thừa giao dịch dẫn đến: Tìm kiếm vị trí tập ứng viên trong giao dịch chưa hiệu quả; tăng thời gian tạo vùng dữ liệu để mở rộng ứng viên; duyệt qua cả những giao dịch không chứa ứng viên để tính giá trị hữu ích của tập ứng viên; hiệu quả về tốc độ tìm kiếm tập hữu ích không cao do thuật toán thực hiện đồng thời 3 công việc với mỗi giao dịch, kể cả giao dịch không chứa ứng viên (tìm kiếm vị trí ứng viên, thực hiện phép chiếu ứng viên trên giao dịch và tính độ hữu ích ứng viên).

Dựa trên các nhận xét trên, bài báo có một số đóng góp như sau: i) Đề xuất cấu trúc P-set với mục đích hạn chế số giao dịch tham gia trực tiếp vào quá trình khai phá tập hữu ích cao; ii) Đề xuất phương pháp chiếu ngược trên P-set giữa tập ứng viên và vùng dữ liệu đang xét nhằm hạn chế số giao dịch tham gia thực hiện phép chiếu tạo vùng dữ liệu mới cho việc mở rộng tập ứng viên và tính giá trị hữu ích tập ứng viên; iii) Đề xuất thuật toán IEFIM, cải tiến từ thuật toán EFIM dựa trên P-set và phương pháp chiếu ngược.

Các nghiên cứu liên quan

Bài toán khai phá tập hữu ích cao do Yao và Hamilton đưa ra vào năm 2004 [4]. Các tác giả cũng đề xuất thuật toán UMining dựa vào chặn trên (upper bound) của độ hữu ích để khai phá HUI. Sau đó thêm thuật toán UMining-H, một dạng heuristic của UMining do thay đổi cách tính chặn trên độ hữu ích để tỉa ứng viên. Cả UMining và UMining-H đều có khả năng tỉa nhầm các tập HUI. Năm 2005, Liu và các đồng sự đề xuất một chặn trên mới có tên là TWU (Transaction Weighted Utilization) dùng cho khai phá HUI [6]. TWU của các itemset thỏa tính chất bao đóng giảm nên có thể dựa vào đó để tỉa ứng viên. Vì vậy, các tác giả đề xuất thuật toán Two-Phase dựa trên TWU để tỉa ứng viên. Two-Phase được chia làm hai giai đoạn bao gồm: (1) Khai phá tất cả các itemset có TWU lớn hơn hay bằng minutil (là ngưỡng tối thiểu do người sử dụng đưa vào); (2) Từ tập các itemset có TWU thỏa mãn minutil, Two-Phase quét CSDL để tính độ hữu ích của từng itemset và lọc ra các itemset có độ hữu ích thỏa mãn minutil. Do Two-Phase tốn khá nhiều lần quét

CSDL và sinh nhiều ứng viên trong phase 1 nên không hiệu quả trên các CSDL lớn.

Sau Two-Phase, hầu hết các thuật toán đều vận dụng phương pháp tỉa dựa trên TWU và áp dụng những chiến lược riêng để nâng cao hiệu quả tỉa ứng viên. TWU-Mining và DTWU-Mining của Le và các đồng sự vận dụng, phát triển cấu trúc IT-Tree của Zaki [21] thành cấu trúc WIT-Tree [8] để giảm số lần duyệt CSDL. Cùng vận dụng FP-Growth [22], IHUP của Ahmed và các đồng sự đề xuất, tạo ứng viên trên IHUP-Tree [7], còn UP-Growth và UP-Growth+ của Tseng và các đồng sự thì thực hiện tạo ứng viên trên UP-Tree [9] bên cạnh các chiến lược bổ trợ: Giảm độ hữu ích của tập không triển vọng trên UP-Tree toàn cục (DGU - Discarding Global Unpromising item), giảm độ hữu ích của nút trên UP-Tree toàn cục (DGN - Discarding Global Node utilities), loại bỏ tập không triển vọng cục bộ (DLU - Discarding Local Unpromising item), giảm độ hữu ích của nút trên UP-Tree cục bộ (DLN - Decreasing Local Node utilities), giảm độ hữu ích của tập không triển vọng cục bộ trên UP-Tree cục bộ (DNU - Discarding local unpromising items and their estimated Node Utilities) và giảm độ hữu ích của nút không triển vọng cục bộ trong UP-Tree cục bộ (DNN - Decreasing local Node utilities for the nodes of local UP-Tree). Sau khi tạo danh sách ứng viên IHUP, UP-Growth và UP-Growth+ đều quét lại CSDL để tính giá trị hữu ích và xem xét việc ứng viên có phải là tập hữu ích cao hay không.

Với HUI-Miner của Liu và Qu đi theo hướng mới, chỉ duyệt CSDL một lần và lưu vào cấu trúc do nhóm đề xuất Utility-list [3], khai phá và tỉa ứng viên trên cấu trúc đó. Tuy nhiên, số lượng Utility-list do HUI-Miner tạo ra khá nhiều nên Fournier-Viger và các đồng sự đề xuất thuật toán FHM (2014) [23] và cấu trúc EUCS (Estimated Utility Co-occurrence Structure) [23] với phương án tỉa EUCP (Estimated Utility Co-occurrence Pruning) [23] để hạn chế việc tạo Utility-list nhằm tăng tốc độ thuật toán. Cùng mục đích với FHM, HUP-Miner [20] của Krishnamoorthy áp dụng thêm 2 chiến lược tỉa theo phân vùng (PA - PArtitioned utility) [20] và tỉa trước (LA - LookAhead utility) [20] bên cạnh chiến lược tỉa theo Utility-list.

Mỗi thuật toán đều phát huy hiệu quả chiến lược tỉa ứng viên của mình và đẩy nhanh tốc độ tìm kiếm tập hữu ích cao. Tuy nhiên, trong quá trình khai phá, các thuật toán vẫn quét các giao dịch rỗng và chưa có phương án xử lý các dòng dữ liệu tương đồng với nhau (giống các phần tử xuất hiện trong giao dịch và chỉ khác số lượng). Vì vậy, EFIM đã đề xuất 3 chiến lược: Chiếu trên CSDL (HDP - High utility Database Projection) [11] để tìm kiếm các phần trùng nhau; chiến lược trộn các giao dịch (HTM - High utility Transaction Merging) [11] để giảm không gian tìm kiếm và các phương pháp tỉa bằng các chặn trên theo giá trị hữu ích cục bộ (Local utility) [11] và giá trị hữu ích trên nhánh phụ (Sub-tree utility) [11] để loại các tập ứng viên không mong đợi.

Thuật toán IEFIM

Các khái niệm liên quan

Cho $I = \{i_1, i_2, \ldots, i_n\}$ là tập các phần tử và CSDL D gồm bảng hữu ích (Utility table) và bảng giao dịch (Transaction table) như hình 1. Mỗi phần tử trong I có giá trị hữu ích nhất định chứa trong bảng hữu ích. Một giao dịch T trong bảng giao dịch được xác định duy nhất bằng *tid* và chứa tập con của I có liên kết với số lượng tương ứng.

Định nghĩa 1: Giá trị hữu ích mở rộng của phần tử i, ký hiệu *eu(i),* là những giá trị hữu ích của i trong bảng hữu ích của D [6].

Định nghĩa 2: Giá trị hữu ích nội bộ của phần tử i trong giao dịch T, ký hiệu *iu(i,T)*, là đếm giá trị kết hợp của phần tử i thuộc T trong bảng giao dịch của D [6].

Định nghĩa 3: Giá trị hữu ích của phần tử i trong giao dịch T, ký hiệu *u(i,T)*, là phép nhân giữa *iu(i,T)* và *eu(i)* hay $u(i,T) = iu(i,T) \times eu(i)$ [6]. Ví dụ: $eu(a) = 1$, $iu(a,T2) = 4$ và $u(a,T2) = iu(a,T2) \times eu(a) = 4 \times 1 = 4$.

Định nghĩa 4: Giá trị hữu ích của tập X trong giao dịch T, ký hiệu *u(X,T)*, là tổng giá trị hữu ích của các phần tử thuộc X có trong giao dịch T hay $u(X,T) = \Sigma_{i \in X \wedge X \subseteq T} u(i,T)$ [6].

Định nghĩa 5: Giá trị hữu ích của tập X, ký hiệu *u(X)*, là tổng giá trị hữu ích của X trong tất cả giao dịch T có chứa X trên DB hay $u(X) = \Sigma_{T \in D \wedge X \subseteq T} u(X,T)$ [6].

Định nghĩa 6: Cho trước ngưỡng hữu ích tối thiểu *minutil*, tập X được gọi là tập hữu ích cao nếu giá trị hữu ích của X không nhỏ hơn ngưỡng hay $u(X) \geq minutil$ [6]. Ví dụ: $u(\{a,b\}, T2) = u(a, T2) + u(b, T2) = 4 \times 1 + 1 \times 2 = 6$, và $u(\{a,b\}) = u(\{a,b\}, T2) + u(\{a,b\}, T4) + u(\{a,b\}, T5) = 6 + 9 + 11 = 26$. Nếu $minutil = 20$ thì {a,b} là tập hữu ích cao, ngược lại với $minutil = 30$ thì {a,b} không phải là tập hữu ích cao.

Định nghĩa 7: Giá trị hữu ích của giao dịch T, ký hiệu *tu(T)*, là tổng giá trị hữu ích của các phần có trong T hay $tu(T) = \Sigma_{i \in T} u(i,T)$ và giá trị hữu ích của DB là tổng giá trị hữu ích các giao dịch trong DB [6]. Ví dụ: $tu(T3) = u(\{a\}, T3) + u(\{c\}, T3) + u(\{d\}, T3) = 4 + 2 + 5 = 11$.

Định nghĩa 8: Trọng số giao dịch hữu ích của tập X, ký hiệu *TWU(X)*, là tổng giá trị hữu ích của tất cả các giao dịch có chứa X trên DB hay $TWU(X) = \Sigma_{T \in DB \wedge X \subseteq T} tu(T)$ [6]. Ví dụ: $TWU(\{e\}) = tu(T2) + tu(T4) = 18 + 22 = 40$.

Định nghĩa 9: Gọi $\succ$ là phép sắp xếp thứ tự theo TWU của các phần tử trong *I*. Giá trị hữu ích còn lại của *X* trong giao dịch T, ký hiệu *ru(X,T),* là tổng giá trị hữu ích các phần tử sau *X* trong T, hay là $ru(X,T) = \Sigma_{i \in T \wedge i \succ x \forall x \in X} u(i,T)$ [3]. Ví dụ: *ru({a},T3) = u({c}, T3) + u({d},T3)* = 1+ 5 = 6.

Định nghĩa 10: Cho tập các phần tử *I* được xếp thứ tự theo $\succ$, và tập X, tập các phần tử mở rộng của *X* được định nghĩa như sau $E(X) = \{z \mid z \in I \wedge Z \succ X \forall X \in X\}$ [11].

Định nghĩa 11: Cho giao dịch T và tập *X*, phép chiếu của tập *X* trên giao dịch T được xác định là $T_x = \{ i \mid i \in T \wedge i \in E(X)\}$ [11]. Ví dụ: cho *X = {b}*, xét phép thứ tự $a \succ b \succ c \succ d \succ e$ thì $T1_x = \emptyset$, $T2_x = \{a\}$.

Định nghĩa 12: Cho CSDL *D* và tập *X*, phép chiếu của tập *X* trên *D* được định nghĩa như sau $D_X = \{T_X \mid T \in D \wedge T_X \neq \emptyset\}$ [11]. Ví dụ: cho $X = \{c\}$, xét phép thứ tự $a \succ b \succ c \succ d \succ e$, $D_X = \{T1_X, T2_X, T3_X, T5_X\} = \{\{b\},\{a,b\},\{a\},\{a,b\}\}$.

Định nghĩa 13: Cho tập *X*, phần tử $z \in E(X)$ và giá trị hữu ích cục bộ của *(X,z)* được tính như sau $lu(X,z) = \sum_{T \supset (X \cup \{z\})} [u(X,T) + ru(X,T)]$ [11]. Ví dụ: cho *X* = {a}, *lu(X,c) = (u(X,T2) + ru(X,T2)) + (u(X,T3) + ru(X,T3)) + (u(X,T5) + ru(X,T5))* = 18 + 11 + 22 = 51.

Tính chất 1: Cho tập $X, z \in E(X)$, nếu $lu(X,z) < minutil$ thì tất cả các tập mở rộng của tập *X* với *z* đều không thể là tập hữu ích cao [11].

Định nghĩa 14: Cho tập *X* và phần tử $z \in E(X)$, giá trị hữu ích trên nhánh phụ z và tập *X* là $su(X,z) = \sum_{T \supset (X \cup \{z\})} [u(X,T) + u(z,T) + \sum_{i \in T \wedge i \in E(X \cup \{z\})} u(i,T)]$ [11]. Ví dụ, cho *X = {a}, su(X,c) = (u({a},T2) + u({c},T2)+ u({d},T2) + u({e},T2)) + (u({a},T3) + u({c},T3)+ u({d},T3)) + (u({a},T5) + u({c},T5) + u({f},T5))* = 16 + 11 + 7 = 34.

Tính chất 2: Cho tập *X* và $z \in E(X)$, nếu $su(X,z) < minutil$ thì tất cả các tập mở rộng của tập *X* với *z* đều không thể là tập hữu ích cao [11].

Định nghĩa 15: Cho tập *X*, phần tử chính và phần tử phụ (Primary, Secondary item) được định nghĩa như sau: $Primary(X) = \{z \mid z \in E(X) \wedge su(X,z) \geq minutil\}$ và $Secondary(X) = \{z \mid z \in E(X) \wedge lu(X,z) \geq minutil\}$ [11]. Tiếp tục các ví dụ tại định nghĩa 12 và 13, nếu xét minutil = 40 thì *X = {a}* là 1 phần tử phụ, không phải là phần tử chính, nhưng với minutil = 30 thì *X = {a}* vừa là phần tử chính vừa là phần tử phụ.

Định nghĩa 16: Cho 2 giao dịch T_a, T_b chứa các phần tử tương ứng $\{i_1,i_2,\ldots,i_m\}$ và $\{j_1,j_2,\ldots,j_n\}$. T_a và T_b được gọi là đồng nhất hay $T_a = T_b$ nếu thỏa mãn các điều kiện: n = m và $\forall k \in [1,n]$, $i_k = j_k$ [11]. Xét tiếp ví dụ ở định nghĩa 12, thì $T2_X$ và $T5_X$ được xem là đồng nhất vì có cùng kết quả là $\{a,b\}$.

Định nghĩa 17: Cho các giao dịch đồng nhất $Tr_1 = Tr_2 = \ldots = Tr_m$ trên *D*, các giao dịch trên được trộn lại bằng T_m trong đó $\forall i \in T_m, iu(i,T_m) = \sum_{k=1\ldots m} iu(i,T_k)$ [11]. Ví dụ: Giả sử $T2_X$ và $T5_X$ ở định nghĩa 12 là 2 giao dịch độc lập, thì hai giao dịch này được thay bằng $T2'_X$ có giá trị hữu ích nội bộ $iu(\{a\},T2'_X) = iu(\{a\},T2_X) + iu(\{a\},T5_X) = 4 + 3 = 7$ và $iu(\{b\},T2'_X) = iu(\{b\},T2_X) + iu(\{b\},T5_X) = 1 + 4 = 5$.

Định nghĩa 18 (về phép chiếu kết hợp trộn các giao dịch đồng nhất): Khi chiếu tập *X* lên *D*, các giao dịch đồng nhất được trộn bằng một giao dịch mới, ký hiệu cD_X [11]. Phép chiếu kết hợp phép trộn theo ví dụ ở định nghĩa 12 được thể hiện trên hình 2.

Tid	Giao dịch	Số lượng
$T1_X$	{b}	{1}
$T2_X$	{a,b}	{4,1}
$T3_X$	{a}	{4}
$T5_X$	{a,b}	{3,4}

(A) Dữ liệu đầy đủ của D_X với X = {c}.

Tid	Giao dịch	Số lượng
$T1_X$	{b}	{1}
$T2'_X$	{a,b}	{7,5}
$T3_X$	{a}	{4}

(B) Dữ liệu của cD_X với X = {c}.

Hình 2. Minh họa phép chiếu X = {c} trên CSDL và phép trộn kết hợp.

Cấu trúc P-set và thuật toán

EFIM tốn nhiều chi phí cho việc tạo phép chiếu trên tập *X* trên vùng giao dịch đang xét để dự toán sự triển vọng của các tập mở rộng. Với một tập *X* đang xét thì số phần tử cần mở rộng chính bằng lực lượng của tập phần tử phụ $Secondary(X)$.

Xét tập *X* và $z \in Secondary(X)$, và vùng dữ liệu cD_X là các giao dịch cần xét khi mở rộng phần tử. Xét phép chiếu *z* lên vùng cD_X, EFIM buộc phải quét lại toàn bộ cD_X một lần nữa, trong khi có thể xác định được vùng chiếu này khi tìm tập phần tử phụ $Secondary(X)$.

Định nghĩa 19 (phép chiếu ngược của tập X trên D): Cho CSDL *D* và tập X, *P-set* phép chiếu ngược của tập X trên *D* được xác định như sau: $P-set(X) = \{T.id \mid T \in D \wedge X \subseteq T\}$. Ví dụ, xét *X={e}, P-set(X) ={T2,T4}*.

Định nghĩa 20 (phép chiếu ngược mở rộng của tập X với i trên D): Cho CSDL *D* và tập X, phép chiếu ngược của tập *X* với *I* trên *D* được xác định như sau:

$Pex - set(X,i) = \{T'.id \mid T' \in cD_X \wedge i \subseteq T'\}$.

Mệnh đề 1: Giá trị hữu ích của tập X không đổi khi áp dụng *P-set* và *Pex-set(X,i)* trên D.

Giả sử chia CSDL D, theo định nghĩa 5 ta có: $u(X) = \sum_{T \in D1 \wedge X \subseteq T} u(X,T)$ hay $u(X) = \sum_{T \in D \wedge T.id \in P-set(X)} u(X,T)$ (theo định nghĩa 19). Vì vậy, khi áp dụng *P-set*, giá trị hữu ích các tập không thay đổi. Áp dụng thêm định nghĩa 18 và 20, ta chứng minh tương tự với *Pex-set(X,i)*. (1)

Ngoài ta, theo định nghĩa 12 và 18, ta có: $|cD_X| \le |D_X| \le |D|$ (2)

Áp dụng định nghĩa 20, ta có $|Pex\text{-}set(X,i)| \le |cD_X|$ (3)

Kết hợp (2) và (3) suy ra $|Pex\text{-}set(X,i)| \le |D_X|$ (4)

Từ (1) và (4) cho thấy hiệu quả của *P-set* và *Pex-set* tỷ lệ nghịch với độ phổ biến của các tập X và phần tử mở rộng I trên vùng dữ liệu tương ứng D hay cD_X.

Ví dụ, xét $X = \{e\}$, *P-set(X)* = *{T2,T4},* khi cần tính độ hữu ích của X, ta trực tiếp đến T2 và T4 để tính thay vì duyệt cả 5 giao dịch, và hiển nhiên hiệu quả khi sử dụng *P-set({a})* thấp hơn của *P-set({e})* do {a} xuất hiện trong nhiều giao dịch hơn {e}.

Với việc sử dụng *Pex-set*, thuật toán IEFIM thay đổi tại dòng 7 tính *Pex-set(X,i)* song song với *su*(*X, i*) và tại dòng 3, 5 của thủ tục Search (hình 3 và 4).

Thuật toán IEFIM

Input : D: CSDL cần khai phá, *minutil*: Ngưỡng tối thiểu

Output: Các tập hữu ích cao

1. $X = \emptyset$;

2. Duyệt D tính *lu*(*X, i*) cho tất cả $i \in I$;

3. *Secondary*(*X*) = {*i*|*i* ∈ *I* ∧ *lu*(*X, i*) ≥ *minutil*};

4. Sắp xếp tăng dần *Secondary*(*X*) theo giá trị TWU;

5. Duyệt D để xóa các phần tử $i \notin$ *Secondary*(*X*) ra khỏi các giao dịch và xóa các giao dịch rỗng;

6. Sắp xếp các giao dịch T tăng dần;

7. Duyệt D tính *su*(*X,i*) và *Pex-set(X,i)* cho từng phần tử $i \in$ *Secondary*(*X*);

8. *Primary*(*X*) = {*i*|*i* ∈ *Secondary*(*X*) ∧ *su*(*X, i*) ≥ *minutil*};

9. Search (*X*, *D*, *Primary*(*X*), *Secondary*(*X*), *minutil,Pex-set(X,i)*).

Hình 3. Thuật toán IEFIM.

Do *Pex-set(X,i)* là tập chứa *T.id* của phép chiếu D_X-*ex* nên thực hiện đồng thời với việc tính giá trị hữu ích trên nhánh phụ *su*(*X,i*) không làm tăng độ phức tạp của thuật toán. Tương tự như thế với dòng 5 tại thủ tục Search. Hiệu quả của *Pex-set(X,i)* được thể hiện rõ tại dòng 3 của thủ tục Search, nó chỉ xét các giao dịch có *tid* thuộc *Pex-set(X,i)* thay vì quét toàn bộ D_X.

Thủ tục Search

Input: *X*: Tập phần tử đang xét, cD_x: Các giao dịch được chiếu và trộn bởi *X*, *Primary*(*X*): Các phần tử chính *X*, *Secondary*(*X*): Các phần tử mở rộng của *X*, ngưỡng *minutil,Pex-set(X,i):* Tid các giao dịch dùng mở rộng *X* với *{i}*.

Output: Các tập hữu ích cao mở rộng từ *x*.

1. Foreach *item i* ∈ *P rimary* (*X*) **do**

2. $\beta = X \cup \{i\}$;

3. *Dùng Pex-set(X,i)* để duyệt D_x để tính *u*(*β*) và xây dựng D_e; // dùng phép trộn giao dịch;

4. If *u*(*β*) ≥ *minutil* **then** xuất *β*;

5. Duyệt *β-D* tính *su*(*β, z*), *lu*(*β, z*) và *P-set-ex(β,z)* cho tất cả *z* ∈ *Secondary*(*X*) sau *i*;

6. *Primary*(*β*) = {*z* ∈ *Secondary*(*X*)|*su*(*β, z*) ≥ *minutil*};

7. *Secondary*(*β*) = {*z* ∈ *Secondary*(*X*)|*lu*(*β, z*) ≥ *minutil*};

8. *Search* (*β*, D_e, *Primary*(*β*), *Secondary*(*β*), *minutil, Pex-set(β,z)*);

9. End.

Hình 4. Thủ tục Search của IEFIM.

Kết quả thực nghiệm và đánh giá

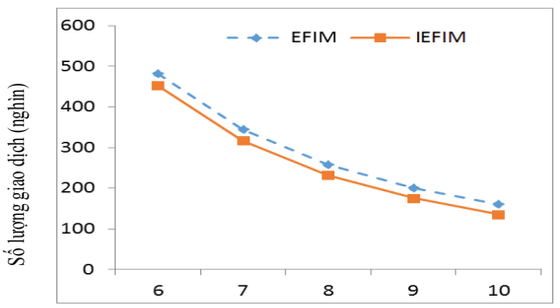
Chúng tôi cài đặt thuật toán IEFIM, tiến hành chạy thực nghiệm so sánh với thuật toán EFIM và CSDL được lấy từ thư viện mở SPMF: An Java Open-Source Data Mining Library tại địa chỉ http://www.philippe-fournier-viger.com/spmf/ [24]. Các thuật toán được thực hiện trên môi trường Java sử dụng hệ điều hành Windows 8.1, 64 bit, RAM 4 GB, CPU Core i3 M350.

Bảng 1. Bảng mô tả dữ liệu thực nghiệm chuẩn.

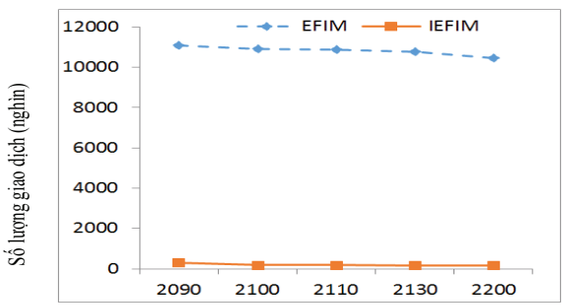
Loại dữ liệu	Số giao dịch	Số phần tử	Độ dài trung bình	Đánh giá
Accident	340183	468	33,8	Đặc
BMS-POS	59601	497	4,8	Thưa
Chess	3196	75	37	Rất đặc
Foodmart	67557	129	43	Thưa
Kosarak	990002	41270	8,1	Thưa
Retail	87943	16465	10,3	Thưa
T10I4D100K	100000	870	10,1	Thưa
T40I10D100K	100000	942	39,6	Thưa

Ngoài ra, các CSDL Retail, T10I4D100K, T40I10D100K được phát sinh ngẫu nhiên từ 1 đến 10 các giá trị: Độ hữu ích của từng phần tử và số lượng trong từng giao dịch, đặc điểm các bộ dữ liệu thực nghiệm chuẩn được mô tả tại bảng 1.

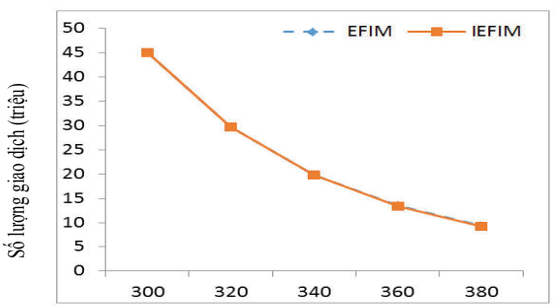
Chúng tôi chạy thực nghiệm trên các CSDL nêu trên và ghi lại thời gian thực hiện, số giao dịch được quét để thực hiện phép chiếu nhằm xây dựng vùng dữ liệu mới dùng mở rộng ứng viên và tính giá trị hữu ích.



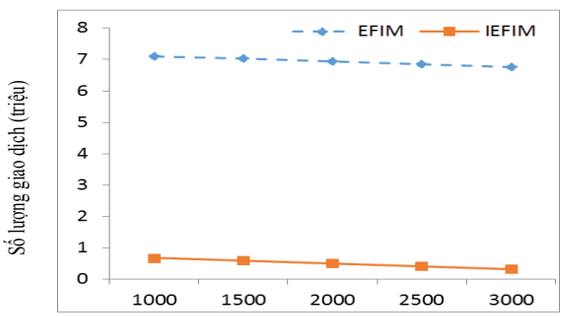
(A) Đồ thị so sánh số lượng giao dịch trên CSDL Accident.



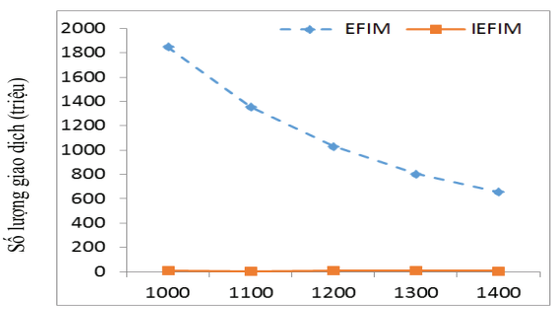
(B) Đồ thị so sánh số lượng giao dịch trên CSDL BMS-POS.



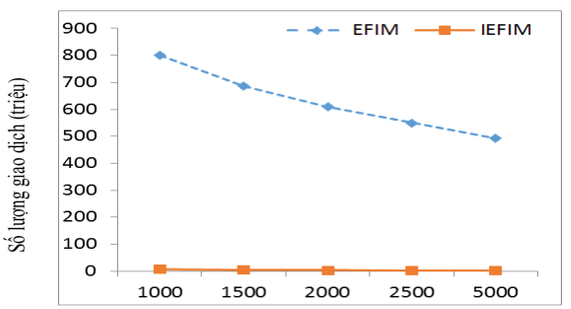
(C) Đồ thị so sánh số lượng giao dịch trên CSDL Chess.



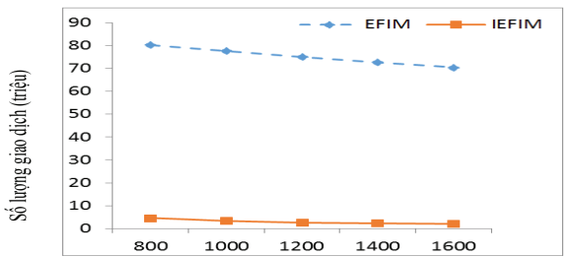
(D) Đồ thị so sánh số lượng giao dịch trên CSDL Foodmart.



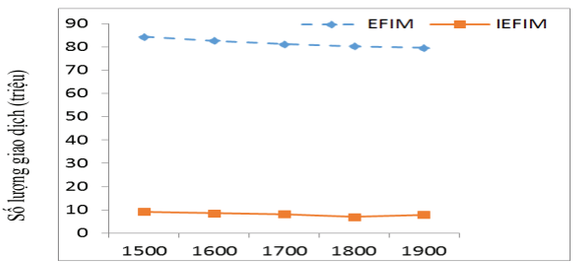
(E) Đồ thị so sánh số lượng giao dịch trên CSDL Kosarak.



(F) Đồ thị so sánh số lượng giao dịch trên CSDL Retail.



(G) Đồ thị so sánh số lượng giao dịch trên CSDL 10I4D100K.

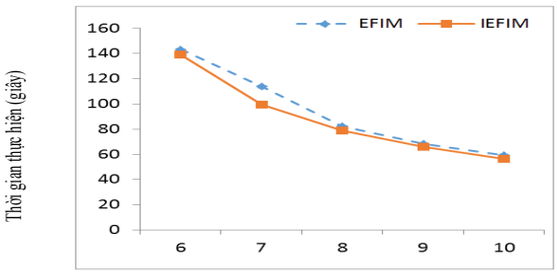


(H) Đồ thị so sánh số lượng giao dịch trên CSDL T40I10D100K.

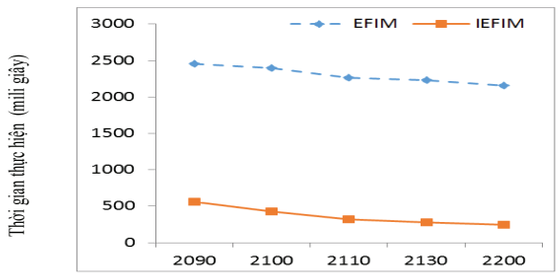
Hình 5. Đồ thị so sánh số lượng giao dịch.

Từ kết quả thực nghiệm được thể hiện qua các đồ thị so sánh số lượng giao dịch tham gia phép chiếu tạo vùng dữ liệu để mở rộng ứng viên và tính giá trị hữu ích của tập ứng viên (hình 5) ta có nhận xét, khi áp dụng phương pháp chiếu ngược, thuật toán IEFIM giảm hẳn số giao dịch, giảm từ 9 (như T40I10D100K, hình 5H) đến 400 lần (như Kosarak, hình 5E) đối với loại CSDL được đánh giá thưa, và tỷ lệ này giảm dần đối với các loại dữ liệu được đánh giá dày và rất dày, cụ thể với Accident và Chess (hình 5A và 5C), số lượng giao dịch được quét giảm không đáng kể.

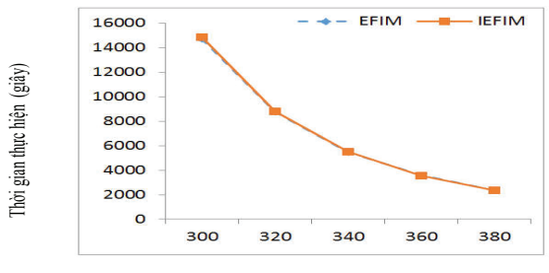
Về thời gian thực hiện, thuật toán IEFIM nhanh hơn hẳn EFIM trên CSDL thưa, giảm thời gian thực hiện từ 2 (Foodmart, hình 6D) đến 60 lần (Retail, hình 6F). Đối với CSDL đặc/rất đặc như Accident, Chess thì thời gian cải thiện không đáng kể (hình 6A và 6C).



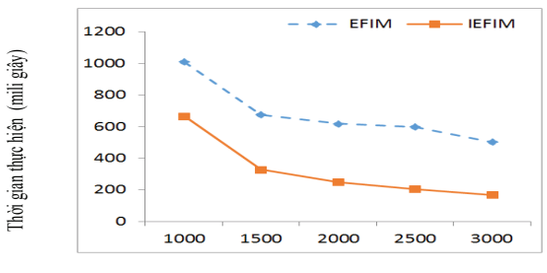
(A) Đồ thị so sánh thời gian trên CSDL Accident.



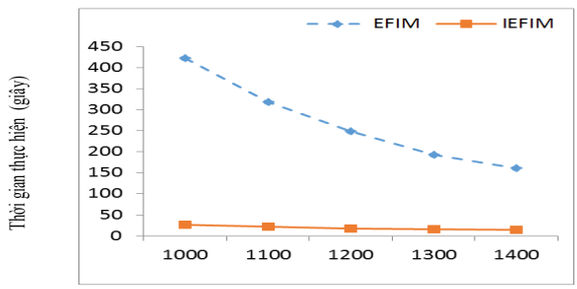
(B) Đồ thị so sánh thời gian trên CSDL BMS-POS.



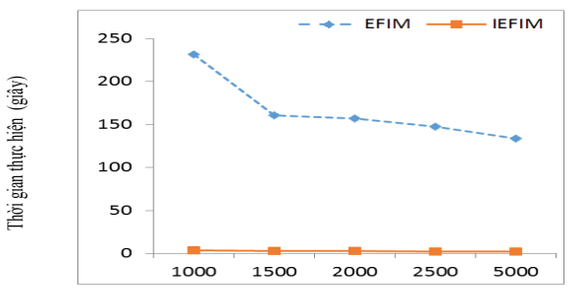
(C) Đồ thị so sánh thời gian trên CSDL Chess.



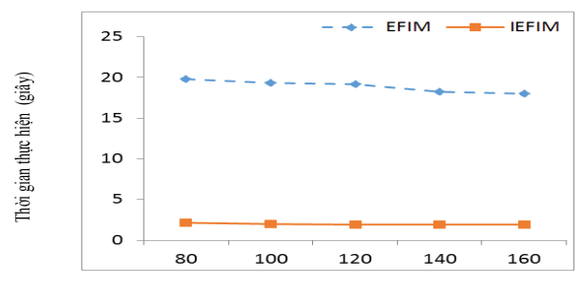
(D) Đồ thị so sánh thời gian trên CSDL Foodmart.



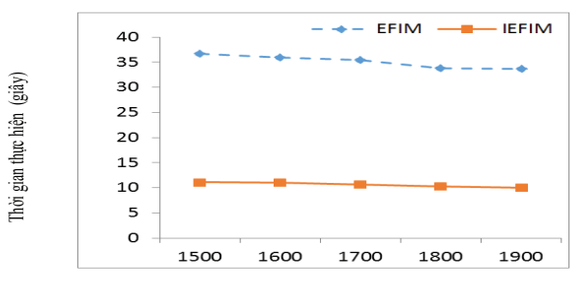
(E) Đồ thị so sánh thời gian trên CSDL Kosarak.



(F) Đồ thị so sánh thời gian trên CSDL Retail.



(G) Đồ thị so sánh thời gian trên CSDL T10I4D100K.



(H) Đồ thị so sánh thời gian trên CSDL T40I10D100K.

Hình 6. Đồ thị so sánh thời gian thực hiện.

Nguyên nhân: Hiệu quả của thuật toán IEFIM tập trung vào việc giảm tổng số lần các giao dịch được quét qua phép chiếu để tạo vùng dữ liệu mới phục vụ mở rộng ứng viên, nên khi tỷ lệ chênh lệch này không đáng kể thì hiệu quả thuật toán cải tiến không nhiều. Tốc độ thuật toán không được cải thiện nhiều do việc giảm số lượng giao dịch thừa đối với CSDL dày và rất dày không đáng kể nhưng chi phí tạo phép chiếu ngược lại tăng so với các loại dữ liệu khác. Kết quả so sánh về số lượng giao dịch cần xét và thời gian chạy thuật toán thể hiện ở đồ thị minh họa ở hình 5 và hình 6.

Kết luận và hướng phát triển

Trong bài báo này, chúng tôi đã giới thiệu giải pháp chiếu ngược P-set để tăng tốc độ khai phá tập hữu ích cao bằng cách hạn chế quét các số giao dịch thừa. Bằng thực nghiệm đã chứng minh được hiệu quả của P-set với dữ liệu thưa và cũng phù hợp với các môi trường dữ liệu kinh doanh trong thực tế được thể hiện như Foodmart. Với hiệu quả này, chúng tôi sẽ tiếp tục nghiên cứu để áp dụng vào các hướng khai phá khác tập hữu ích cao như khai phá HUI đóng, khai phá Top-k HUI... Ngoài ra, việc lai ghép nhiều kỹ thuật khác nhau để tăng tốc độ, giảm không gian tìm kiếm và không gian bộ nhớ cũng được chúng tôi quan tâm.

LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi Quỹ Phát triển Khoa học và Công nghệ Quốc gia (NAFOSTED) trong khuôn khổ đề tài mã số 102.05-2015.10. Chúng tôi xin trân trọng cảm ơn.

TÀI LIỆU THAM KHẢO

[1] R. Agrawal, T. Imielinski, A.N. Swami (1993), "Mining association rules between sets of items in large databases", *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington D.C., pp.207-216.

[2] R. Agrawal, R. Srikant (1994), "Fast algorithms for mining association rules in large databases", *Proc. Int'l Conf. Very Large Data Bases*, pp.487-499.

[3] M. Liu, J. Qu (2012), "High utility itemsets without candidate generation", *21st ACM International Conference on Information and Knowledge Management*, pp.55-64.

[4] H. Yao, H.J. Hamilton, C.J. Butz (2004), "A foundational approach to mining itemset utilities from databases", *In Proc. SIAM Int'l Conf. Data Mining*, pp.482-486.

[5] H. Yao, H.J. Hamilton (2006), "Mining Itemset Utilitied from Transaction Databases", *Data and Knowledge Engeneering,* **59(3)**, pp.603-626.

[6] Y. Liu, W.K. Liao, A.N. Choudhary (2005), "A two-phase algorithm for fast discovery of high utility itemsets", *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp.689-695.

[7] C. Ahmed, S.K. Tanbeer, B.S. Jeong, Y.K. Lee (2009), "Efficient tree structures for high utility pattern mining in incremental databases", *IEEE Transactions on Knowledge and Data Engineering,* **21(12)**, pp.1708-1721.

[8] B. Le, H. Nguyen, T.A. Cao, B. Vo (2009), "A Novel Algorithm for Mining High Utility Itemsets", *Proceedings of 1st Asian Conference on Intelligent Information and Database Systems*, Quang Binh, Vietnam (IEEE press), pp.13-17.

[9] V.S. Tseng, C.W. Wu, B.E. Shie, P.S. Yu (2010), "Upgrowth: Anefficientalgorithm for high utility itemset mining", *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp.253-262.

[10] B. Le, H. Nguyen, B. Vo (2011), "An efficient strategy for mining high utility itemsets", *International Journal of Intelligent Information and Database Systems,* **5(2)**, pp.164-176.

[11] S. Zida, P. Fournier-Viger, J.C.W. Lin, C.W. Wu, V.S. Tseng (2015), "EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining", *Advances in Artificial Intelligence and Soft Computing,* Springer., pp.530-546.

[12] C.W. Wu, P. Fournier-Viger, P.S. Yu, V.S. Tseng (2011), "Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets", *IEEE 11th International Conference on Data Mining*, pp.824-833.

[13] V.T. Tseng, C.W. Wu, P. Fournier-Viger, P.S. Yu (2015), "Efficient Algorithms for Mining the Concise and Lossless Representation of High Utility Itemsets", *IEEE Transactions on Knowledge and Data Engineering,* **27(3)**, pp.726-739.

[14] C.W. Wu, B.E. Shie, V.T. Tseng, P.S. Yu (2012), "Mining top-K high utility itemsets", *KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* , pp.78-86.

[15] V.T. Tseng, C.W. Wu, P. Fournier-Viger, P.S. Yu (2016), "Efficient Algorithms for Mining Top-K High Utility Itemsets", *IEEE Transactions on Knowledge and Data Engineering,* **28(1)**, pp.54-67.

[16] C.J. Chu, V.S. Tseng, T. Liang (2008), "An efficient algorithm for mining temporal high utility itemsets from data streams", *Journal of Systems and Software,* **81(7)**, pp.1105-1117.

[17] Bai-En Shie, S. Yu Philip, V.S. Tseng (2012), "Efficient algorithms for mining maximal high utility itemsets from data streams with different models", *Expert Systems with Applications,* **39(17)**, pp.12947-12960.

[18] J.C.W. Lin, W. Gan, P. Fournier-Viger, T.P. Hong, V.T. Tseng (2016), "Efficient algorithms for mining high-utility itemsets in uncertain databases", *Knowledge-Based Systems,* **96**, pp.171-187.

[19] V.S. Tseng, B.E. Shie, C.W. Wu, P.S. Yu (2013), "Efficient algorithms for mining high utility itemsets from transactional databases", *IEEE Transactions on Knowledge and Data Engineering,* **25(8)**, pp.1772-1786.

[20] K. Krishnamoorthy (2015), "Pruning strategies for mining high utility itemsets", *Expert Systems with Applications,* **42(5)**, pp. 2371-2381.

[21] M. Zaki (2000), "Scalable algorithms for association mining", *IEEE Transactions on Knowledge and Data Engineering,* **12(3)**, pp.372-390.

[22] J. Han, J. Pei, Y. Yin, R. Mao (2004), "Mining frequent patterns without candidate generation: A frequent pattern tree approach", *Data Mining and Knowledge Discovery,* **8(1)**, pp.53-87.

[23] P. Fournier-Viger, C.W. Wu, S. Zida, V.T. Tseng (2014), "FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning", *Proc. 21st International Symposium on Methodologies for Intelligent Systems (ISMIS 2014),* Springer, pp.83-92.

[24] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.W. Wu, V.S. Tseng (2014), "SPMF: A java open-source pattern mining library", *The Journal of Machine Learning Research,* **15(1)**, pp.3389-3393.