RESEARCH ARTICLE

CẢI TIẾN THUẬT TOÁN HMINER CHO VIỆC KHAI THÁC TẬP HỮU ÍCH CAO TRÊN DỮ LIỆU GIAO TÁC THƯA

Trần Minh Thái, Trần Anh Duy, Lê Thị Minh Nguyện, Nguyễn Thanh Trung

Khoa Công nghệ Thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM

thaitm@huflit.edu.vn, duyta@huflit.edu.vn, nguyenltm@huflit.edu.vn, trungnt2@huflit.edu.vn

TÓM TẮT— Khai thác tập hữu ích cao đóng vai trò quan trọng trong khai thác dữ liệu. Việc khai thác này giúp khám phá ra các tập mục có nhiều hữu ích, tức là có tầm quan trọng hoặc là lợi nhuận cao, trong cơ sở dữ liệu giao tác. Điều đó giúp cho các công ty, siêu thị có thể định hướng và đưa ra chiến lược kinh doanh cho phù hợp nhằm đem lại lợi nhuận cao nhất. Tùy thuộc vào dạng dữ liệu dày hoặc thưa, những thuật toán khai thác sẽ có chiến lược khai thác phù hợp và có những hiệu quả nhất định. Nội dung bài báo tập trung vào nghiên cứu và đề xuất phương pháp khai thác đối với tập dữ liệu thưa thông qua một số cách thức tổ chức dữ liệu và kỹ thuật cắt tỉa. Kết quả đánh giá thực nghiệm đã chứng tỏ được tính khả thi của giải pháp được đề xuất.

Từ khóa— Dữ liệu giao tác, Luật kết hợp, Khai thác dữ liệu, Tập hữu ích cao.

I. GIỚI THIỆU

Khai thác luật kết hợp [1] là một trong những vấn đề được nghiên cứu và đề cập nhiều nhất trong lĩnh vực khai thác dữ liệu. Thông thường, quá trình khai thác luật kết hợp được chia làm hai giai đoạn: (1) Giai đoạn đầu tiên là khai thác tập phổ biến; (2) sau đó ở giai đoạn thứ hai là sinh luật từ các tập phổ biến. Tuy nhiên, các luật kết hợp chỉ khám phá các tập phổ biến mà không xét các tập ít phổ biến. Trong khi đó, có sự tồn tại của tập ít phổ biến nhưng lại có độ hữu ích cao. Chính vì vậy, khai thác tập phổ biến trong thực tế vẫn còn nhiều hạn chế, không đáp ứng được nhu cầu của người sử dụng vì chúng xem các mục trong giao tác có sự quan trọng ngang nhau và không quan tâm đến số lượng (hữu ích nội) và giá trị hữu ích (hữu ích ngoại) thu được đối với từng mục. Khai thác tập hữu ích cao nhằm giải quyết vấn đề này, nghĩa là có xem xét cả hữu ích nội và hữu ích ngoại của từng mục, để tìm ra các itemset mang lại hữu ích cao trong cơ sở dữ liệu (CSDL) giao tác.

Khai thác tập hữu ích cao (High Utility Itemset - HUI) là sự mở rộng của bài toán khai thác tập phổ biến [2]. Khai thác tập hữu ích cao cung cấp nhiều thông tin hữu ích hơn khai thác tập phổ biến do các mục trong CSDL đều có giá trị hữu ích. Một trong những thuật toán hiệu quả trong khai thác tập hữu ích cao có thể kể đến như HMiner [3], EFIM [4], FHM [5] và Direct Discovery of High Utility Patterns (D^2HUP) [6].

II. ĐỊNH NGHĨA BÀI TOÁN

CSDL giao tác D: Cho $I = \{i_1, i_2, \dots, i_m\}$ là một tập các mục (item) riêng biệt. Một giao tác $T_j = \{x_l | l = 1,2, \dots, N_j, x_l \in I\}$, trong đó N_j là số item trong giao tác T_j. Một CSDL giao tác D chứa các giao tác, $D = \{T_1, T_2, \dots, T_n\}$, trong đó n là tổng số các giao tác trong CSDL. Ví dụ, trong Bảng 1 minh họa một CSDL giao tác mẫu và kèm theo các giá trị hữu ích của các item được thể hiện trong Bảng 2.

Bảng 1. *Một ví dụ về CSDL giao tác*

TID	Giao tác	Số lượng (IU)	Hữu ích (U)	Hữu ích giao tác (TU)
T1	a, c, d	1, 1, 1	5, 1, 2	8
T2	a, c, e, g	2, 6, 2, 5	10, 6, 6, 5	27
T3	a, b, c, d, e, f	1, 2, 1, 6, 1, 5	5, 4, 1, 12, 3, 5	30
T4	b, c, d, e	4, 3, 3, 1	8, 3, 6, 3	20
T5	b, c, e, g	2, 2, 1, 2	4, 2, 3, 2	11
T6	a, c, d	3, 3, 3	15, 3, 6	24
T7	a, b, c, d, f	1, 1, 1, 2, 3	5, 2, 1, 4, 3	15
T8	a, b, c, e, f	1, 2, 2, 1, 1	5, 4, 2, 3, 1	15
Tổng hữu ích giao tác				**150**

Bảng 2. *Giá trị hữu ích của các item*

Item	*a*	*b*	*c*	*d*	*e*	*f*	*g*
Hữu ích ngoại (EU)	5	2	1	2	3	1	1

Chiều dài itemset và item trước: Một itemset $X = \{x_1, x_2, \dots, x_k\} \subseteq I, x_i \in I$ được gọi là một *k-itemset* có chiều dài k. Item trước của một item x đã cho trong một giao tác được ký hiệu là $Prev(x, T_j)$, trong đó $x \in I$. Ví dụ trong Bảng 1, xét giao tác T_1 có item a trước item c nên $Prev\ (c, T_1) = a$ và giao tác T_5 không có item nào trước item b nên $Prev\ (b, T_5) = -1$.

Giá trị hữu ích: Mỗi item $x_i \in I$ có 1 giá trị hữu ích ngoại (ví dụ như lợi nhuận) được ký hiệu là $EU(x_i)$ và mỗi item $x_i \in T_j$ có thông tin thể hiện số lượng trong giao tác gọi là giá trị hữu ích nội ký hiệu là $IU(x_i, T_j)$. Ví dụ trong Bảng 2, $EU(b) = 2$ và $IU(b, T_3) = 2$.

Hữu ích của item: Hữu ích của item $x_i \in T_j$ ký hiệu là $U(x_i, T_j)$ được tính là tích của hữu ích ngoại và hữu ích nội của item trong giao tác T_j , công thức xác định $U(x_i, T_j)$ là $U(x_i, T_j) = EU(x_i) * IU(x_i, T_j)$. Ví dụ trong Bảng 1, $U(b, T_3) = EU(b) * IU(b, T_3) = 2 * 2 = 4$.

Hữu ích của itemset trong giao tác: Hữu ích của itemset X trong giao tác $T_j(X \subseteq T_j)$ được ký hiệu là $U(X, T_j)$. Công thức xác định $U(X, T_j)$ là $U(X, T_j) = \sum_{x_i \in X} U(x_i, T_j)$. Ví dụ như trong Bảng 1,$U(ac, T_1) = 5 + 1 = 6$.

Hữu ích của itemset trong CSDL giao tác: Hữu ích của itemset X trong CSDL D được ký hiệu là $U(X)$. Công thức xác định $U(X)$ là $U(X) = \sum_{X \subseteq T_j \in D} U(X, T_j)$. Ví dụ, $U(ac) = U(ac, T_1) + U(ac, T_2) + U(ac, T_3) + U(ac, T_6) + U(ac, T_7) + U(ac, T_8) = 6 + 16 + 6 + 18 + 6 + 7 = 59$.

Hữu ích của giao tác: Hữu ích của giao tác được ký hiệu là $TU(T_j)$. Công thức xác định $TU(T_j)$ là $TU(T_j) = \sum_{X \subseteq T_j, x_i \in X} U(x_i, T_j)$. Ví dụ, $TU(T_5) = U(b, T_5) + U(c, T_5) + U(e, T_5) + U(g, T_5) = 11$.

Trọng số hữu ích giao tác của itemset: Trọng số hữu ích giao tác của một itemset X là tổng hữu ích của các giao tác có chứa itemset X, ký hiệu $TWU(X)$ được xác định bằng công thức $TWU(X) = \sum_{X \subseteq T_j \in D} TU(T_j)$. Ví dụ trong Bảng 3 thể hiện các TWU của CSDL được cho trong Bảng 1.

Ngưỡng hữu ích tối thiểu được người sử dụng chỉ định trước và được ký hiệu là δ. Giá trị hữu ích tối thiểu ($minutil$) được tính theo công thức $minutil = \delta * \sum_{T_j \in D} TU(T_j)$. Giả sử với CSDL trong Bảng 1 và $\delta = 28\%$ thì $minutil = 0.28 * 150 = 42$. Độ hỗ trợ của một itemset X trong CSDL D được ký hiệu là $Sup(X)$. Đó là tần suất xuất hiện của itemset X trong D được tính bằng số lần xuất hiện của itemset X chia cho tổng số giao tác n.

Bảng 3. *Trọng số hữu ích giao tác trong Bảng 1*

Item	*g*	*f*	*b*	*d*	*e*	*a*	*c*
TWU	38	60	91	97	103	119	120

Bảng 4. *Sắp xếp lại các item trong Bảng 1 theo TWU*

TID	**Giao tác**	**Hữu ích (U)**	**Hữu ích giao tác (TU)**
*T*1	*d, a, c*	2, 5, 1	8
*T*2	*e, a, c*	6, 10, 6	22
*T*3	*f, b, d, e, a, c*	5, 4, 12, 3, 5, 1	30
*T*4	*b, d, e, c*	8, 6, 3, 3	20
*T*5	*b, e, c*	4, 3, 2	9
*T*6	*d, a, c*	6, 15, 3	24
*T*7	*f, b, d, a, c*	3, 2, 4, 5, 1	15
*T*8	*f, b, e, a, c*	1, 4, 3, 5, 2	15

Hữu ích còn lại của itemset: Nếu $TWU(X) < minutil$ thì $\forall X' \supseteq X, TWU(X') \leq TWU(X) < minutil$ [6]. Cho T_j/X là ký hiệu tập tất cả các item sau X trong T_j. Ví dụ trong Bảng 1 $T_1/ac = d, T_7/ac = bdf$. Kích thước tập tất cả các item sau X trong T_j được ký hiệu là $S(T_j/X)$ là số các item có trong tập hợp. Ví dụ $S(T_7/ac) = |bdf| = 3$. Hữu ích còn lại của một itemset X trong giao tác $T_j(X \subseteq T_j)$ được ký hiệu là $RU(X, T_j)$ được xác định bằng công thức $RU(X, T_j) = \sum_{x_i \in \left(\frac{T_j}{X}\right)} U(x_i, T_j)$. Ví dụ trong Bảng 1, $RU(ac, T_1) = 2, RU(ac, T_1) = 4 + 2 + 3 = 9$. Hữu ích còn lại của một itemset X trong CSDL D được ký hiệu là $RU(X)$ được xác định bằng công thức $RU(X) = \sum_{X \subseteq T_j \in D} RU(X, T_j)$ Ví dụ trong Bảng 1, $RU(ac) = 2 + 6 + 24 + 6 + 9 + 8 = 55$.

Tiền tố hữu ích: Cho một itemset X và một item mở rộng $y \in I$, tiền tố hữu ích của một itemset Xy trong giao tác T_j được xác định là $PU(Xy, T_j) = U(X, T_j)$. Nếu $X = \emptyset$, thì $PU(Xy, T_j) = 0$. Các item trong CSDL giao tác được sắp xếp thứ tự các TWU theo thứ tự tăng dần. Đối với CSDL mẫu đang xét (Bảng 1), thứ tự của các item

là: $g > f > b > d > e > a > c$. Đối với CSDL mẫu trong Bảng 1, tập thứ tự của các item được cung cấp trong Bảng 4, giả sử $minutil = 42$.

Các phần mở rộng của một itemset X được xác định như là tập của tất cả các item sau X trong tập thứ tự của các item. Ví dụ các phần mở rộng của một itemset b là $\{d, e, a, c\}$. Kích thước của phần mở rộng hoàn chỉnh của một itemset X được ký hiệu là $C(X)$. Ví dụ, $C(be) = |\{a, c\}| = 2$, và $C(fb) = |\{d, e, a, c\}| = 4$

Hữu ích đóng và hữu ích đóng còn lại: Hữu ích đóng của một itemset $X = \{x_1, x_2, \ldots, x_k\}$ trong giao tác T_j ký hiệu là $CU(X, T_j)$, được xác định là:

$$CU(X, T_j) = \begin{cases} U(X, T_j), if |X| > 1, C(X - x_k) = S(T_j/X - x_k) \\ 0 \end{cases}$$

Ví dụ $CU(f, T_3) = 0$ vì kích thước của itemset là 1. Tương tự, $CU(fb, T_3) = U(fb, T_3) = 9$ vì $C(fb, T_3) = |\{b, d, e, a, c\}| = 5$ và $S(T_3/fb - b) = S(\{b, d, e, a, c\}) = 5$. $CU(fb, T_7) = 0$ vì $C(fb - b) <> S(T_7/fb - b)$ và $CU(fe, T_8) = 0$ vì $C(fe - e) = |\{e, a, c\}| <> S(T_8/fe - e) = S(T_8/f) = |\{b, e, a, c\}|$. Hữu ích đóng của một itemset X có kích thước bằng hai được xác định là: $CU(X) = \sum_{X \subseteq T_j \in D} CU(X, T_j)$. Ví dụ, $CU(fb) = CU(fb, T_3) + CU(fb, T_7) + CU(fb, T_8) = U(fb, T_3) + 0 + 0 = 9$. Hữu ích đóng còn lại của một itemset $X = \{x_1, x_2, \ldots, x_k\}$ trong giao tác T_j ký hiệu là $CRU(X, T_j)$ được xác định là:

$$CRU(X, T_j) = \begin{cases} RU(X, T_j), if |X| > 1, C(X - x_k) = S(T_j/X - x_k) \\ 0 \end{cases}$$

Đối với ví dụ đang xét, $CRU(f, T_3) = 0$ vì kích thước của itemset là 1. Tương tự, $CRU(fb, T_3) = RU(fb, T_3) = 21$. Hữu ích đóng còn lại của một itemset X có kích thước bằng 2 được xác định là: $CRU(\mathrm{X}) = \sum_{X \subseteq T_j \in D} CRU(X, T_j)$. Ví dụ $CRU(fb) = CRU(fb, T_3) + CRU(fb, T_7) + CRU(fb, T_8) = 21 + 0 + 0 = 21$.

Tiền tố hữu ích đóng: Tiền tố hữu ích đóng của một itemset $X = \{x_1, x_2, \ldots, x_k\}$ trong giao tác T_j ký hiệu là $CPU(X, T_j)$ được xác định là:

$$CPU(X, T_j) = \begin{cases} PU(X, T_j), if |X| > 1, C(X - x_k) = S(T_j/X - x_k) \\ 0 \end{cases}$$

Tiền tố hữu ích đóng của một itemset X có kích thước bằng hai được xác định là: $CPU(\mathrm{X}) = \sum_{X \subseteq T_j \in D} CPU(X, T_j)$. Đối với ví dụ đang xét, $CPU(fb) = CPU(fb, T_3) + CPU(fb, T_7) + CPU(fb, T_8) = 5 + 0 + 0 = 5$. Hữu ích không đóng ký hiệu là NU, hữu ích không đóng còn lại ký hiệu là NRU và tiền tố hữu ích không đóng ký hiệu là NPU của một itemset X được xác định tương ứng là: $NU(X) = U(X) - CU(X)$; $NRU(X) = RU(X) - CRU(X)$; $NPU(X) = PU(X) - CPU(X)$. Ví dụ trong Bảng 1, $NU(fb) = U(fb) - CU(fb) = 19 - 9 = 10$, $NRU(fb) = RU(fb) - CRU(fb) = 41 - 21 = 20$ và $NPU(fb) = PU(fb) - CPU(fb) = 9 - 5 = 4$.

Bài toán khai thác tập hữu ích cao bao gồm xác định tất cả các itemsets trong D mà có các giá trị hữu ích lớn hơn hoặc bằng với giá trị hữu ích tối thiểu $minutil$ do người dùng chỉ định. Đó là: $HUI = \{X: U(X) \mid X \subseteq I, U(X) \geq minutil\}$. Đối với CSDL giao tác trong Bảng 1 khi $minutil = 42$, thì $HUI = \{f\ bdac: 42, da: 54, dac: 60, a: 45, ac: 59\}$.

III. CÁC NGHIÊN CỨU LIÊN QUAN

Thuật toán khai thác tập hữu ích cao Two-Phase [7] (được đưa ra dựa trên Apriori [1]) đề xuất một phương pháp ước lượng mẫu dùng TWU. Tuy nhiên, thuật toán này có hạn chế là tạo ra quá nhiều ứng viên. Do đó, thuật toán IIDS [8] được đề xuất nhằm giải quyết vấn đề này. Tiếp theo, thuật toán IHUP [12] với cách tiếp cận phương pháp FP-Growth [10] trên mô hình TWU. Điểm chung của các thuật toán Two-Phase, IIDS, và IHUP đều tìm tất cả các mẫu có trọng số lợi nhuận giao tác cao hơn hay bằng giá trị ngưỡng cho trước và sau đó xác định các mẫu có hữu ích cao thực sự.

Thuật toán UP-Growth [9] có cải tiến đáng kể với bốn chiến lược khai thác bằng cách dùng cấu trúc cây UP-Tree bao gồm: loại bỏ những item không tiềm năng toàn cục, giảm độ hữu ích của các nút toàn cục, loại bỏ những item không tiềm năng cục bộ (DLU) và giảm độ hữu ích các nút cục bộ (DLN). Cấu trúc dữ liệu này quét CSDL hai lần và phát sinh mẫu hữu ích cao tiềm năng (PHUPs) từ cây. Thêm vào đó, thuật toán MU-Growth [11] đề xuất thêm hai kỹ thuật tỉa trên cấu trúc cây MIQ-Tree. Ngoài ra, một cấu trúc dữ liệu dạng danh sách cũng được trình bày trong [6]. Gần đây, một trong những thuật toán khai thác hiệu quả tập có độ hữu ích cao là thuật toán EFIM [4]. Thuật toán sử dụng kỹ thuật trộn các giao tác trùng lặp và chiếu trên CSDL. Thuật toán HMiner [3] đề xuất một cấu trúc danh sách hữu ích để lưu trữ hiệu quả thông tin. Khái niệm hữu ích đóng và không đóng của một itemset trong giao tác được trình bày với giá trị lợi hữu ích cô đọng hơn. HMiner đưa ra phương pháp xác định

các giao tác trùng lặp và áp dụng một số chiến lược cắt tỉa để khai thác hiệu quả tập hữu ích cao. Đánh giá so sánh với EFIM, FHM và D2HUP đã cho thấy HMiner cải thiện đáng kể về bộ nhớ sử dụng và thời gian thực hiện trên hầu hết các dữ liệu.

IV. THUẬT TOÁN HMINER VÀ ĐỀ XUẤT CẢI TIẾN

Trong phần này, bài báo trình bày cấu trúc dữ liệu được sử dụng và thuật toán HMiner [3] cho việc khai thác tập hữu ích cao. Sau đó, bài báo trình bày một cấu trúc cải tiến trong việc tỉa mẫu nhằm giúp giảm bớt thời gian xử lý và xét tập ứng viên trong quá trình khai thác tập hữu ích cao.

A. CẤU TRÚC DANH SÁCH HỮU ÍCH

Một cấu trúc danh sách hữu ích (Compact Utility List - *CUL*) của một itemset $X = \{x_1, x_2 \ldots x_k\}$ là một cấu trúc lưu trữ thông tin về itemset. *CUL*(*X*) lưu trữ: (1) thông tin tổng hợp như *NU(X), NRU(X), CU(X), CRU(X), CPU(X)* và (2) thông tin xác định các giao tác (tidlist). Mỗi tidlist là một bộ 5 thành phần <*tid, NU(X,Tj), NRU(X,Tj*), *NPU(X,Tj), PPOS(X,Tj*)> (Bảng 5).

Bảng 5. *Cấu trúc danh sách hữu ích của 1 itemset X*

	Itemset X			
	NU(X)	*NRU(X)*	*CU(X) / CRU(X) / CPU(X)*	
T_j	*NU(X, T_j)*	*NRU(X, T_j)*	*NPU(X, T_j)*	*PPOS(X, T_j)*

Trong đó, PPOS đề cập đến vị trí tidlist của item trước trong cùng một giao tác và được định nghĩa như sau:

$$PPOS(X, T_j) = \begin{cases} \left|CUL[Prev(x_k, T_j)].tidlist\right| & nếu\ Prev(x_k)! - 1 \\ -1 & ngược\ lại \end{cases}$$

Ví dụ, xét dữ liệu mẫu ở Bảng 1, danh sách hữu ích cho 1-itemset và k-itemset được thể hiện trong Hình 1 và 2 tương ứng.

f — 9 51 0/0/0

3	5	25	0	-1
7	3	12	0	-1
8	1	14	0	-1

b — 22 58 0/0/0

3	4	21	0	0
4	8	12	0	-1
5	4	5	0	-1
7	2	10	0	1
8	4	10	0	2

d — 30 45 0/0/0

1	8	24	0	-1
3	12	9	0	0
4	6	6	0	1
7	4	6	0	3

e — 18 34 0/0/0

2	6	16	0	-1
3	3	6	0	1
4	3	3	0	2
5	3	2	0	2
8	3	7	0	4

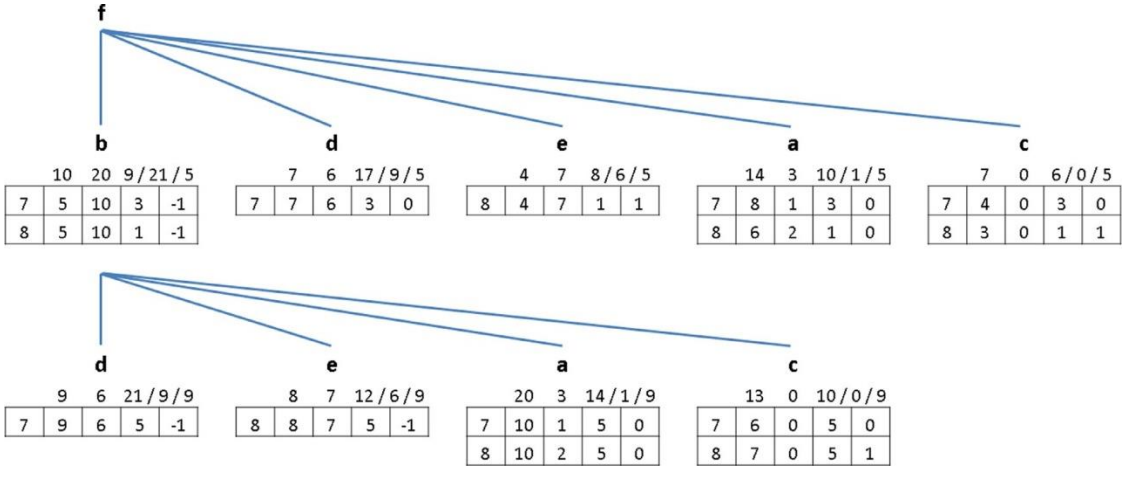
a — 45 14 0/0/0

1	20	4	0	0
2	10	6	0	0
3	5	1	0	1
7	5	1	0	3
8	5	2	0	4

c — 19 0 0/0/0

1	4	0	0	0
2	6	0	0	1
3	1	0	0	2
4	3	0	0	2
5	2	0	0	3
7	1	0	0	3
8	2	0	0	4

Hình 1. *Danh sách hữu ích cho 1-itemset.*



Hình 2. *Danh sách hữu ích cho k-itemset (k>1).*

B. THUẬT TOÁN HMINER

Thuật toán HMiner gồm ba bước: (1) tính toán ban đầu các TWU và tạo ra *CUL* 1-itemset, (2) duyệt cây tìm kiếm và (3) xây dựng *CUL* k-itemset. Mã giả của thuật toán HMiner được thể hiện trong Thuật toán 1.

Thuật toán 1: HMiner

Input: CSDL giao tác (D) và ngưỡng minutil

Output: Tập hữu ích cao (HUI)

1: Scan *D* and Compute *TWU* for all 1-itemsets

2: Initialize 1-itemset *CULs*, and a hash table HT = {}

3: **for** each $T_j \in D$ **do**

4: $newT = \{ x \mid TWU(x) >= minutil \ \forall x \in T_j \}$ //TWU-Prune

5: Sort *newT* as per the ordering heuristic

6: ru = 0 //remaining utility

7: *dupPos* = *HT.get*(*newT*) //check if duplicate transaction exists

8: **if** *dupPos* == *NULL* **then** //new transaction

9: HT[newT] = |CULs[x_k].tidlist| //x_k is the last item in newT

10: **for** each item $x \in reverseOrder(newT)$ **do**

11: add to $CULs[x].tidlist$, $<T_j, U(x, T_j), ru, 0, PPOS(x, T_j)>$

12: $ru = ru + U(x, T_j)$

13: end for

14: **else** //duplicate transaction, update utilities

15: pos = dupPos //position of last item in CULs

16: **for** each item $x \in reverseOrder(newT)$ **do**

17: update $CULs[x].tidlist$ at *pos* with $<\_, U(x, T_j), ru, 0, \_>$

18: $ru = ru + U(x, T_j)$

19: pos = CULs[x].tidlist[pos].PPOS //previous item position

20: end for

21: **end if**

22: **if** (EUCS_PRUNE) build EUCS

23: end for

24: HUI = Explore-Search-Tree(∅, *CULs, minutil)*

Thuật toán 1B: Explore-Search-Tree

Input: Tiền tố itemset (R), CULs, minutil

Output: Tất cả HUIs với tiền tố R

1: **for** each utility list position *i* in *CULs* **do**

2: $X = CULs[i]$

3: **if** U(X) ≥ minutil **then** HUI = {HUI ∪ X}

4: **if** U(X) + RU(X) ≥ minutil **then** //U-Prune

5: exCULs = ConstructCUL(X, CULs, i + 1, minutil)

6: R = {R ∪ X} //update prefix with extension

7: Explore-Search-Tree(R, exCULs, minutil)

8: end if

9: end for

Thuật toán 1C: ConstructCUL

Input: *X*, *CULs*, vị trí bắt đầu mở rộng CULs của itemset *X*s, *minutil*

Output: *exCULs* list of extensions of *X*

1: sz = |CULs| - st, extSz = sz //if a transaction has all extensions}
2: **for** each position *j* in sz **do**
3: **if** (EUCS_PRUNE and EUCS[X, CULs[st + j]] < minutil) **then**
4: *exCULs*[*j*] = *NULL* , decrement *extSz* by 1
5: else
6: exCULs[j] = {}, ey[j] = 0 //track tid position in CULs
7: LAU[j] = CU(X) + CRU(X) + NU(X) + NRU(X) //LA-Prune
8: CUTIL[j] = CU(X) + CRU(X) //C-Prune
9: end if
10: end for
11: initialize a hash table *HT* = {}
12: **for** each tidlist element *ex* in *X* **do**
13: newT = ∅
14: **for** each j in *sz* **do**
15: **if** (*exCULs*[j] == *NULL*) go to step 14
16: eylist = CULs[st + j].tidlist
17: **while** (*ey*[j]<|*eylist*| and *eylist*[*ey*[j]].*tid*<*ex.tid*) **do** increment *ey*[j]
18: **if** *ey*[j] < |*eylist*| and *eylist*[*ey*[j]].*tid* = *ex.tid* **then**
19: *newT* = { *newT* ∪ j}
20: **else** {//apply LA-Prune}
21: LAU = LAU - NU(X, ex.tid) - NRU(X, ex.tid)
22: **if** (*LAU* < *minutil*) **then** *exCULs*[j] = *NULL*, decrement *extSz* by 1
23: **end if**
24: end for
25: **if** |*newT*| == *extSz* **then**
26: Update-Closed(*X*, *CULs*, *st*, *exCULs*, *newT* , *ex.tid*)
27: **else**
28: *dupPos* = *HT.get(newT)* //check if a duplicate transaction exists
29: **if** *dupPos* == *NULL* **then** //new transaction
30: HT[newT] = |CULs[x_k].tidlist| //x_k is the last item in newT
31: Insert new entries in *exCULs* for each *newT*
32: **else** //duplicate transaction, update utilities
33: Update-Element(*X*, *CULs*, *st*, *exCULs*, *newT* , *ex.tid*, *dupPos*)
34: **end if**
35: end if
36: **for** each (j in *sz)* increment *CUTIL*[j] by *NU*(*X*, *tid)* + *nru(X,tid)*

37: end for

38: filter *exCULs* where *CUTIL*[j] < *minutil* or *exCULs*[j] = *NULL*

39: **return** *exCULs*

Thuật toán 1D: Update-Closed

Input: X, CULs, st, exCULs, newT, tid

Output: *exCULs* updated

1: *nru* = 0 //remaining utility

2: **for** each element j in reverseOrder(*newT*) **do**

3: *ey* = *CULs*[*st* + j]

4: increment exCULs[j].CU by NU(X,tid) + NU(ey,tid) - NPU(X,tid)

5: increment *exCULs*[j].*CRU* by *nru*, and *exCULs*[j].*CPU* by N*U(X,tid*)

6: nru = nru + NU(ey, tid) - NPU(X, tid)

7: end for

Thuật toán 1E: Update-Element

Input: X, CULs, st, exCULs, newT, tid, pos

Output: *exCULs* updated

1: *nru* = 0 //remaining utility

2: **for** each element *j* ∈ *reverseOrder(newT)* **do**

3: *ey* = *CULs*[*st* + j]

4: update *exCULs*[j].*tidlist* at *pos* with <_,NU(X,tid) + NU(ey,tid) - NPU(X,tid), nru, NU(X,tid),_>

5: nru = nru + NU(ey, tid) - NPU(X, tid)

6: pos = ey.tidlist[pos].PPOS //previous item position

7: end for

C. ĐỀ XUẤT CẢI TIẾN

HMiner sử dụng cấu trúc ước lượng giá trị hữu ích đồng thời (Estimated Utility Co-occurrence Structure - EUCS) để tỉa một lượng lớn các k-itemset không thỏa ngưỡng (dòng 22 trong Thuật toán 1). Cấu trúc này được sử dụng trong FHM [5]. Đây là một ma trận tam giác (Bảng 6), trong đó chứa thông tin *TWU* cho một cặp item và được xác định là: EUCS[i, j] = TWU(X = {i, j}).

Bảng 6. *Ma trận EUCS với minutil = 52*

Item	***f***	***b***	***d***	***e***	***a***
b	60				
d	**45**	65			
e	**45**	74	**50**		
a	60	60	77	67	
c	60	89	97	96	114

Tuy nhiên, trong ma trận EUCS vẫn còn tồn tại những thông tin của những cặp không thỏa ngưỡng *minutil* gây tốn kém không gian lưu trữ và xử lý. Do vậy, bài báo này đề xuất thay thế thông qua cấu trúc gồm 3 danh sách để lưu TWU của các cặp item nhằm giúp giảm không gian và thời gian tỉa mẫu trong quá trình tìm tập hữu ích cao (Bảng 7).

Bảng 7. *Danh sách kết nối các item và giá trị hữu ích thay cho EUCS với minutil = 52*

Item 1	***a***	***a***	***a***	***a***	***a***	***b***	***b***	***b***	***b***	***c***	***c***	***c***
Item 2	***b***	***c***	***d***	***f***	***e***	***c***	***d***	***e***	***e***	***d***	***e***	***f***
Utility	60	114	77	60	67	89	65	74	74	97	96	60

Việc dùng cấu trúc thay thế này có một số đặc điểm như: (i) Không tồn tại những thao tác so sánh và lưu trữ dư thừa do cấu trúc danh sách không tồn tại những cặp item không thoả ngưỡng TWU. Điều này sẽ giúp giảm chi

phí so sánh và không gian lưu trữ so với ma trận tam giác; (ii) Việc định vị và xác định TWU của cặp item nhanh hơn bằng cách sử dụng chiến lược tìm kiếm trên danh sách. Cụ thể, dựa vào danh sách Item 1 để xác định vị trí bắt đầu i, vị trí i sẽ được dùng để tìm item còn lại trên danh sách Item 2 mà không cần phải xét từ đầu như xử lý trên ma trận tam giác. Giả sử cần xác định TWU của cặp item *{b, d}*. Đầu tiên, xác định vị trí bắt đầu của item *b* trên danh sách Item 1 (vị trí 6). Tiếp theo, chúng ta bắt đầu tìm item *d* trên danh sách Item 2 bắt đầu tại vị trí 6 (Bảng 8).

Bảng 8. *Giới hạn không gian tìm kiếm cho cặp item {b, d} trên Bảng 7*

Item 1	***a***	***a***	***a***	***a***	***a***	***b***	***b***	***b***	***b***	***c***	***c***	***c***
Item 2	***b***	***c***	***d***	***f***	***e***	***c***	***d***	***e***	***e***	***d***	***e***	***f***
Utility	60	114	77	60	67	89	65	74	74	97	96	60

V. KẾT QUẢ THỰC NGHIỆM

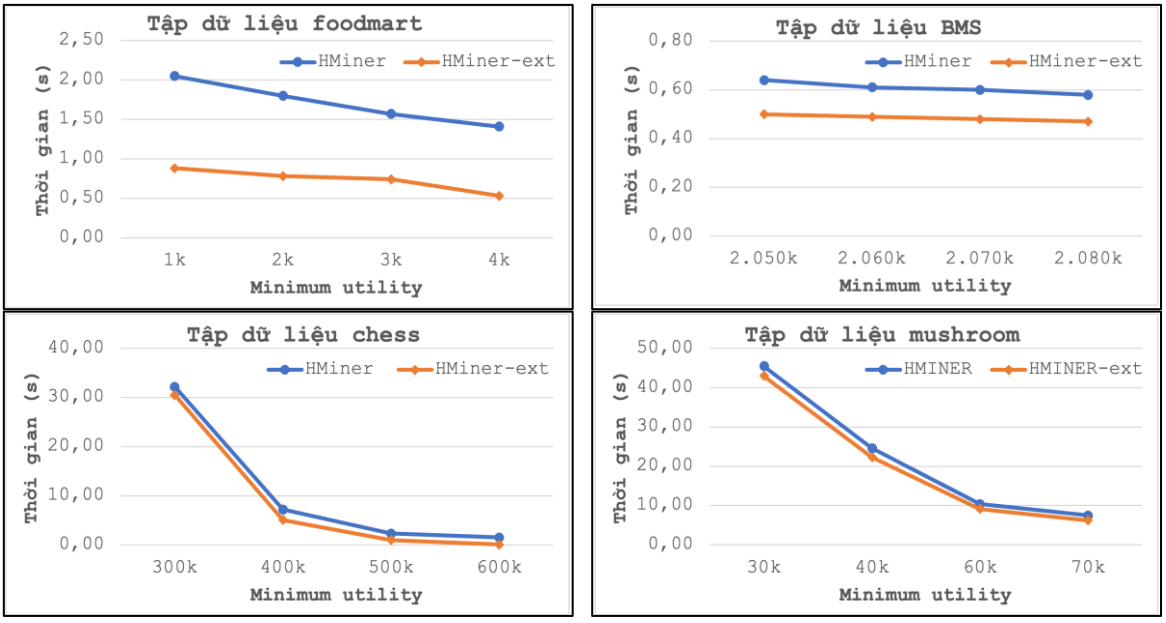
Thực nghiệm được tiến hành đánh giá thuật toán HMiner với thuật toán đề xuất cải tiến cấu trúc dữ liệu phục vụ cho việc tỉa mẫu với tên HMiner-ext. Chương trình được viết bằng ngôn ngữ Java chạy trên máy tính Intel Core i5 3.3GHz, bộ nhớ 8GB và hệ điều hành Windows 11. Dữ liệu thực nghiệm và các thông số được thể hiện trong Bảng 9. Các giá trị hữu ích ngoại cho các item được phát sinh trong phạm vi từ từ 1.0 đến 10.0 và các giá trị hữu ích nội cho các item được phát sinh trong phạm vi từ 1 đến 10.

Bảng 9. *Dữ liệu dùng cho thực nghiệm*

Tập dữ liệu	Số giao tác	Số item	Độ dài trung bình	Loại dữ liệu
foodmart	4,141	1,559	4.4	Thưa
BMS	59,601	497	4.8	Thưa
mushroom	8,124	119	23	Dày
chess	3,196	75	37	Rất dày

A. THỜI GIAN THỰC HIỆN

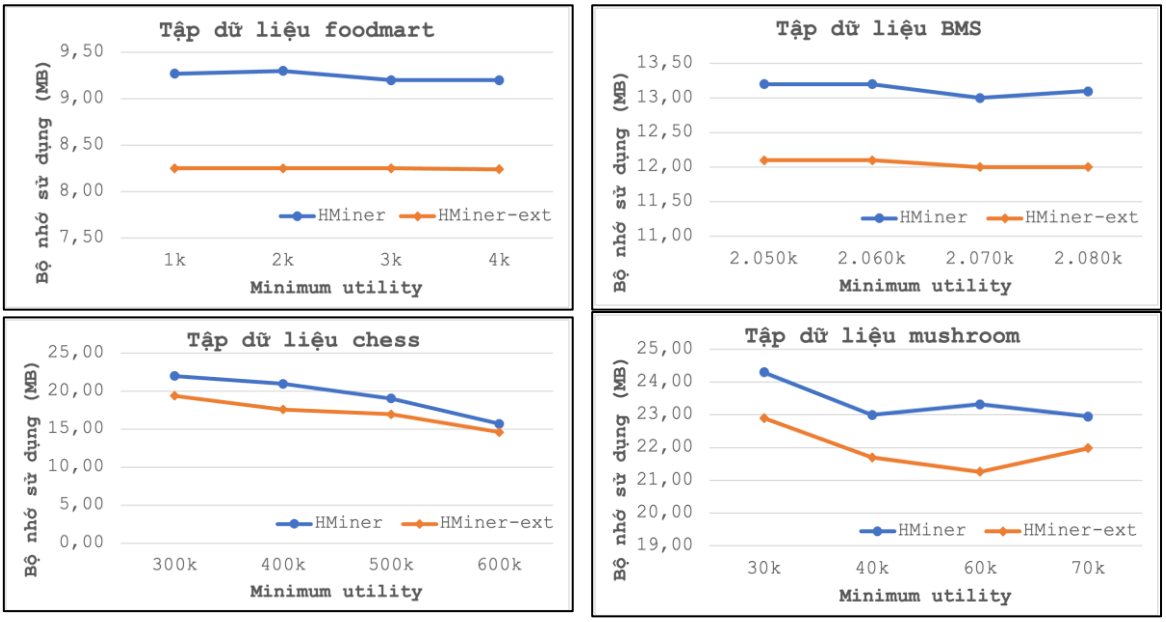
HMiner-ext loại bỏ những cặp item không thỏa ngưỡng thông qua cấu trúc danh sách TWU đã giúp chương trình giảm đáng kể thời gian xét và tỉa mẫu. Hình 3 cho thấy thời gian thực hiện của thuật toán Hminer và HMiner-ext trên các tập dữ liệu và các ngưỡng hữu ích khác nhau. Kết quả thực nghiệm chứng tỏ HMiner-ext thực hiện nhanh hơn so với HMiner trên hầu hết các tập dữ liệu, đặc biệt là đối với những tập dữ liệu thưa. Do đặc điểm của tập dữ liệu thưa, CSDL có nhiều item nhưng tần suất xuất hiện của các item này trên mỗi giao dịch rất thấp. Chính vì vậy, khi sử dụng ma trận tam giác sẽ có nhiều cặp item khi kết hợp sẽ có giá trị TWU không thoả ngưỡng nhiều. Điều này, dẫn đến không gian lưu trữ dư thừa lớn so với việc sử dụng cấu trúc danh sách TWU.



Hình 3. *So sánh thời gian thực hiện với các thay đổi giá trị minutil.*

B. BỘ NHỚ SỬ DỤNG

Tương tự như đối với thực nghiệm về thời gian thực thi, bộ nhớ sử dụng của thuật toán cải tiến HMiner-ext cũng cho kết quả tốt hơn đối với tập dữ liệu thưa. Hình 4 thể hiện kết quả so sánh về bộ nhớ sử dụng của các thuật toán. Trong thực nghiệm này, HMiner-ext cũng sử dụng bộ nhớ ít hơn so với HMiner trong các tập dữ liệu và nhất là trên tập dữ liệu thưa.



Hình 4. *So sánh bộ nhớ sử dụng với các thay đổi giá trị minutil.*

Bài báo đã khảo sát và trình bày vấn đề khai thác tập hữu ích cao trong CSDL giao tác. Trong đó, nội dung bài báo tập trung vào nghiên cứu một thuật toán khai thác tiêu biểu là thuật toán HMiner. Thông qua đó, bài báo cũng đề xuất một cách tiếp cận nhằm cải tiến cấu trúc EUCS giúp cho việc tỉa mẫu nhanh chóng và ít tốn kém bộ nhớ trong quá trình khai thác. Thực nghiệm đã chứng tỏ được sự cải tiến này cho kết quả thực hiện tốt hơn trên các tập dữ liệu, đặc biệt là với tập dữ liệu thưa. Hướng phát triển có thể đánh giá hơn nữa trên tập dữ liệu lớn và phức tạp hơn. Đồng thời nghiên cứu, thực nghiệm và kết hợp các phương pháp tiếp cận với nhau nhằm hiệu quả hơn trong khai thác tập hữu ích cao.

VI. TÀI LIỆU THAM KHẢO

[1] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules," *Proc. Of the 20th Int'l Conf. on Very Large Data Bases (VLDB 1994),* p. 487–499, 1994.

[2] P. Fournier-Viger, J. C.-W. Lin, T. Truong-Chi and R. Nkambou, "A Survey of High Utility Itemset Mining," *Springer,* pp. 1-45, 2019.

[3] S. Krishnamoorthy, "HMiner: Efficiently mining high utility itemsets," *Expert Systems with Applications,* v. 90, p. 168–183, 2017.

[4] S. Zida, P. Fournier-Viger, J. C. W. Lin, C. W. Wu and V. S. Tseng, "EFIM: a fast and memory efficient algorithm for high-utility itemset mining," *Knowlegde and Information Systems, V*. 51, No. 2, pp. 595-625, 2017.

[5] P. Fournier-Viger, C. W. Wu, S. Zida and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," 2014.

[6] J. Liu, K. Wang, B.C.M. Fung, "Direct discovery of high utility itemsets without candidate generation," 2012.

[7] Y. Liu, W.-K. Liao, A.N. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," *Advances in Knowledge Discovery and Data Mining (PAKDD 2005),* p. 689–695, 2005.

[8] Y.-C. Li, J.-S. Yeh, C.-C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," *Data Knowl. Eng. 61 (1),* p. 198–217, 2008.

[9] V.S. Tseng, C.-W. Wu, B.-E. Shie, P.S. Yu, "UP-growth: an efficient algorithm for high utility itemset mining," *Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2010),* p. 253–262, 2010.

[10] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation," *Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data,* p. 1–12, 2000.

[11] U. Yun, H. Ryang, K. Ryu, "High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates," *Expert Syst. Appl. 41 (8),* p. 3861–3878, 2014.
[12] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Transactions on Knowledge and Data Engineering, pp. 1708--1721, 2009.

AN IMPROVED OF HMINER FOR MINING HIGH UTILITY ITEMSETS ON SPARSE TRANSACTION DATASETS

Tran Minh Thai, Tran Anh Duy, Le Thi Minh Nguyen, Nguyen Thanh Trung

ABSTRACT – High-utility itemset mining plays an important role in data mining. This mining helps to discover highly useful itemsets, i.e., itemsets of high importance or profit, in transactional databases. That helps companies and supermarkets to give appropriate business orientation and strategies to bring the highest profit. Depending on the form of dense or sparse data, mining algorithms will have suitable mining strategies and achieve certain efficiency. This paper focuses on researching and proposing mining methods on sparse datasets through data presentations and pruning techniques. The experimental evaluation results have proved the feasibility of the proposed solution.

TS. Trần Minh Thái tốt nghiệp cử nhân CNTT năm 2001 và thạc sỹ Tin học năm 2006 ĐH Khoa học Tự nhiên – ĐH Quốc gia TPHCM, nhận bằng tiến sỹ CNTT năm 2017 do ĐH Quốc gia TPHCM cấp. Anh ta từng là giảng viên và quản lý khoa CNTT trường CĐ CNTT TPHCM từ 2002 - 2015. Từ 2015 - hiện tại, anh là giảng viên và là trưởng bộ môn HTTT thuộc khoa CNTT trường ĐH Ngoại ngữ - Tin học TPHCM. Lĩnh vực nghiên cứu chính của anh liên quan đến vấn đề khai thác dữ liệu, ẩn dữ liệu, xử lý dữ liệu lớn và nhận dạng.

ThS. Trần Anh Duy nhận học vị thạc sĩ Khoa học máy tính trường Đại học Khoa Học Tự Nhiên năm 2017. Hiện là giảng viên khoa Công nghệ Thông tin trường Đại Học Ngoại ngữ - tin học TPHCM. Lĩnh vực nghiên cứu đang quan tâm là: Khai thác dữ liệu.

ThS. Lê Thị Minh Nguyện tốt nghiệp thạc sĩ Khoa học máy tính năm 2007 tại trường Đại học Công nghệ thông tin Tp.HCM; từng là giảng viên tại trường Cao đẳng Công nghệ thông tin từ 2003-2015. Từ năm 2015 đến nay là giảng viên thuộc khoa Công nghệ Thông tin trường Đại học Ngoại ngữ Tin học TpHCM. Lĩnh vực nghiên cứu, quan tâm là Khai thác dữ liệu.

ThS. Nguyễn Thanh Trung tốt nghiệp thạc sĩ Khoa học Máy tính năm 2006 tại trường ĐH Khoa học Tự nhiên - ĐH Quốc gia TPHCM. Hiện là giảng viên khoa Công nghệ Thông tin trường ĐH Ngoại ngữ - Tin học TPHCM. Lĩnh vực nghiên cứu chính là Khai thác dữ liệu.