

# CẢI TIẾN THUẬT TOÁN HMINER CHO VIỆC KHAI THÁC TẬP HỮU ÍCH CAO TRÊN TẬP DỮ LIỆU THỪA

*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 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 đặc 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 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. 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 [2], EFIM [3], FHM [4] và D2HUP [5].

## II ĐỊNH NGHĨA BÀI TOÁN

Cho  $I = \{i_1, i_2, \dots, i_m\}$  là một tập các item riêng biệt. Một giao tác  $T_j = \{x_l \mid l = 1, 2, \dots, N_j, x_l \in I\}$ , 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. Bảng 1 thể hiện một ví dụ về CSDL giao tác và kèm theo chi tiết giá trị hữu ích các item trong Bảng 2. Item trước của một item đã cho trong một giao tác được ký hiệu là  $Prev(x, T_j)$ , trong đó  $x \in I$ . Ví dụ trong Bảng 1,  $Prev(c, T_1) = a$  và  $Prev(b, T_5) = -1$  (không có 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.

**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
<b>Tổng hữu ích giao tác</b>				<b>150</b>

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

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

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 một 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  $T_j$ :  $U(x_i, T_j) = EU(x_i) * IU(x_i, T_j)$ . Ví dụ trong Bảng 1,  $U(b, T_3) = EU(b) * IU(b, T_3) = 2 * 2 = 4$ .

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

Hữu ích của một itemset  $X$  trong CSDL  $D$  được ký hiệu là  $U(X)$ .  $U(X) = \sum_{X \subseteq T_j \in D} U(X, T_j)$ . 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 một giao tác được ký hiệu là  $TU(T_j)$ .  $TU(T_j) = \sum_{X \subseteq T_j, x_i \in X} U(x_i, T_j)$ . Ví dụ,  $TU(T_5) = U(b, T_5) + U(c, T_5) + U(e, T_5) + U(g, T_5) = 11$ .

Ngưỡng hữu ích tối thiểu được người sử dụng chỉ định là  $\delta$ . Giá trị hữu ích tối thiểu (*minutil*). Trong đó,  $minutil = \delta * \sum_{T_j \in D} TU(T_j)$ . Giả sử  $\delta = 28\%$  thì  $minutil = 0,28 * 150 = 42$ . Độ hỗ trợ của một itemset  $X$  trong CSDL  $D$  được ký hiệu là  $Sup(X)$ . Đó là tỷ lệ của tần suất xuất hiện của itemset  $X$  trong  $D$  được chia cho tổng số các giao tác  $n$ .

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

**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

$T4$	$b, d, e, c$	8, 6, 3, 3	20
$T5$	$b, e, c$	4, 3, 2	9
$T6$	$d, a, c$	6, 15, 3	24
$T7$	$f, b, d, a, c$	3, 2, 4, 5, 1	15
$T8$	$f, b, e, a, c$	1, 4, 3, 5, 2	15

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  $T_1/ac = d, T_7/ac = df$ . Kích thước của số item nằm sau  $X$  trong  $T_j$  được ký hiệu là  $S(T_j/X)$ . Ví dụ  $S(T_7/ac) = |df| = 2$ . Hữu ích còn lại của một itemset  $X$  trong giao tác  $T_j (X \subseteq T_j)$  được ký hiệu là  $RU(X, T_j), RU(X, T_j) = \sum_{x_i \in (T_j/X)} U(x_i, T_j)$ . Ví dụ trong

Bảng 1,  $RU(ac, T_1) = 2, RU(ac, T_7) = 4 + 3 = 7$ . Hữu ích còn lại của một itemset  $X$  trong CSDL  $D$  được ký hiệu là  $RU(X), RU(X) = \sum_{X \subseteq T_j \in D} RU(X, T_j)$  Ví dụ trong Bảng 1,  $RU(ac) = 2 + 11 + 20 + 6 + 7 + 4 = 50$ .

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 ví dụ đang xét, 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 của một itemset  $X = \{x_1, x_2, \dots, x_k\}$  trong giao tác  $T_j$  ký hiệu là  $CU(X, T_j)$ , được xác định là:

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

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  $C(fb - b) = |\{b, d, e, a, c\}| = 5$  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  $X = \{x_1, x_2, \dots, x_k\}$  trong giao tác  $T_j$  ký hiệu là  $CRU(X, T_j)$  được

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

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:  $CRU(X) = \sum_{X \subseteq T_j \in D} CRU(X, T_j)$ . 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 của một itemset  $X = \{x_1, x_2, \dots, x_k\}$  trong giao tác  $T_j$  ký hiệu là  $CPU(X, T_j)$  được xác định là:

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

Tiền tố hữu ích đóng của một itemset  $X$  có kích thước bằng hai được xác định là:  $CPU(X) = \sum_{X \subseteq T_j \in D} CPU(X, T_j)$ . 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 (NU), hữu ích không đóng còn lại (NRU) và tiền tố hữu ích không đóng (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 có độ 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 sử dụng đã chỉ định. Đó là:  $HUI = \{X: U(X) \mid X \subseteq I,$

$U(X) \geq \text{minutil}\}$ . Đối với CSDL giao tác trong Bảng 1 khi  $\text{minutil} = 42$ , thì  $HUI = \{fbdac: 42, da: 54, dac: 60, a: 45, ac: 59\}$ .

### III CÁC NGHIÊN CỨU LIÊN QUAN

Thuật toán khai thác tập hữu ích cao Two-Phase [7] được đưa ra dựa trên Apriori [5] 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 [9] 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 [4] 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 hai kỹ thuật tĩa trên cấu trúc cây MIQ-Tree. Ngoài ra, một cấu trúc dữ liệu dạng danh sách cũng được trình bày trong [12]. Gần đây, một trong những thuật toán hiệu quả trong khai thác tập có độ hữu ích cao là thuật toán EFIM [3]. Thuật toán sử dụng kỹ thuật trộn các giao tác trùng lặp và chiếu trên CSDL. Thuật toán HMiner [2] 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 tĩa 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 cho việc 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  $X = \{x_1, x_2, \dots, x_k\}$  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 giao tác (hoặc tidlist). Mỗi bộ hoặc bản ghi trong tidlist là một bộ 5 thành phần  $\langle tid, NU(X, T_j), NRU(X, T_j), NPU(X, T_j), PPOS(X, T_j) \rangle$  (Hình 1).

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

**Hình 1.** Cấu trúc danh sách hữu ích.

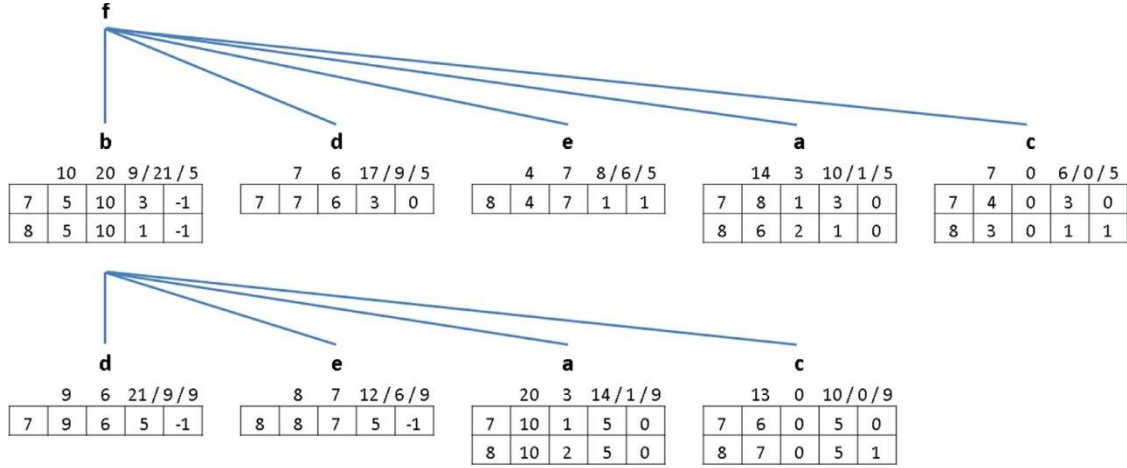
$PPOS$  đề cập đến vị trí tidlist của item trước trong cùng một giao tác và được định nghĩa như sau:

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

Hình 2 và 3 thể hiện danh sách hữu ích trong dữ liệu mẫu ở Bảng 1.

f	b	d	e	a	c
9 51 0/0/0	22 58 0/0/0	30 45 0/0/0	18 34 0/0/0	45 14 0/0/0	19 0 0/0/0
3 5 25 0 -1	3 4 21 0 0	1 8 24 0 -1	2 6 16 0 -1	1 20 4 0 0	1 4 0 0 0
7 3 12 0 -1	4 8 12 0 -1	3 12 9 0 0	3 3 6 0 1	2 10 6 0 0	2 6 0 0 1
8 1 14 0 -1	5 4 5 0 -1	4 6 6 0 1	4 3 3 0 2	3 5 1 0 1	3 1 0 0 2
	7 2 10 0 1	7 4 6 0 3	5 3 2 0 2	7 5 1 0 3	4 3 0 0 2
	8 4 10 0 2		8 3 7 0 4	8 5 2 0 4	5 2 0 0 3
					7 1 0 0 3
					8 2 0 0 4

**Hình 2.** Danh sách hữu ích cho 1-itemset.



**Hình 3.** Danh sách hữu ích cho k- itemset.

## B Thuật toán HMiner

Thuật toán HMiner gồm ba bước: (1) tính toán ban đầu của TWU và tạo ra *CUL* 1-itemset, (2) khảo sát cây tìm kiếm và (3) xây dựng *CUL* k-itemset. Mã giả đầy đủ cho HMiner được cung cấp trong các thuật toán được trình bày trong Thuật toán 1.

**Thuật toán 1** HMiner: Main.

---

**Input:**  $D$ ,  $minutil$       **Output:**  $HUI$

```

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:  $newT = \{ x \mid TWU(x) \geq minutil \ \forall x \in T_j \} \{\}$  //TWU-Prune
5: Sort  $newT$  as per the ordering heuristic (definition 14)
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  $CULs[x].tidlist$  ,  $\langle T_j, U(x, T_j), ru, 0, PPOS(x, T_j) \rangle$ 
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  $\langle \_, U(x, T_j), ru, 0, \_ \rangle$ 

```



```

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:  $HUI = Explore\text{-}Search\text{-}Tree(\emptyset, CULs, minutil)$  {//Thuật toán 1B}

```

---

**Thuật toán 1B**                      HMiner: Explore-Search-Tree.

---

**Input:**  $R$  the itemset prefix,  $CULs$ ,  $minutil$   
**Output:** all  $HUIs$  with prefix  $R$

```

1: for each utility list position  $i$  in  $CULs$  do
2:    $X = CULs[i]$ 
3:   if  $U(X) \geq minutil$  then  $HUI = \{HUI \cup X\}$ 
4:   if  $U(X) + RU(X) \geq minutil$  then {//U-Prune}
5:      $exCULs = ConstructCUL(X, CULs, i + 1, minutil)$  {//Thuật toán 1C}
6:      $R = \{R \cup X\}$  {//update prefix with extension}
7:      $Explore\text{-}Search\text{-}Tree(R, exCULs, minutil)$ 
8:   end if
9: end for

```

---

**Thuật toán 1C**                      HMiner: ConstructCUL.

---

**Input:**  $X$ ,  $CULs$ ,  $st$  starting position of  $Xs$  extension  $CULs$ ,  $minutil$   
**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:    $newT = \emptyset$ 
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) incre-
ment ey[j]
18:   if ey[j] < |eylist| and eylist[ey[j]].tid = ex.tid then
19:       newT = { newT  $\cup$  j}
20:   else { //apply LA-Prune}
21:       LAU = LAU - NU(X, ex.tid) - NRU(X, ex.tid)
22:       if ( LAU < minutil)          exCULs[j] = NULL, decrement extSz by
1
23:   end if
24: end for

25: if | newT | == extSz then { //all item extensions present in
transaction}
26:   Update-Closed( X, CULs, st, exCULs, newT , ex.tid) { //Algo 1D
27: else

28:   dupPos = HT.get(newT ) {} //check if a duplicate transaction
exists
29:   if dupPos == NULL then { //new transaction}
30:       HT[newT] = |CULs[xk].tidlist| { //x/ is the last item in newT
}
31:       Insert new entries in exCULs for each newT
32:   else { //duplicate transaction, update utilities Thuật toán 1E}
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

```

---

<b>Thuật toán 1D</b>	HMiner: 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 NU(X, tid)
6:   nru = nru + NU(ey, tid) - NPU(X, tid)
7: end for

```

---

<b>Thuật toán 1E</b>	HMiner: Update-Element.
----------------------	-------------------------

---

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

**Output:**  $exCULs$  updated

```

1:  $nru = 0$  { } //remaining utility
2: for each element  $j \in reverseOrder(newT)$  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

Tại dòng 22 của Thuật toán 1 có sử dụng cấu trúc ước lượng giá trị hữu ích đồng thời (Estimated Utility Co-occurrence Structure - EUCS) là một ma trận tam giác được sử dụng trong thuật toán FHM [4], trong đó chứa thông tin  $TWU$  cho một cặp item và được xác định là:  $EUCS[i, j] = TWU(X = \{i, j\})$ . Ma trận này dùng để tìm kiếm một lượng lớn k-itemset ( $k \geq 2$ ).

Dựa vào đặc điểm này, bài báo đề xuất cách tiếp cận tạo ra ma trận EUCS' (Hình 4b với  $minutil = 52$ ) được cải tiến từ ma trận EUCS (Hình 4a). Trong đó, EUCS' chỉ lưu những bộ dữ liệu có  $TWU \geq minutil$  nhằm giảm bớt không gian lưu trữ và thời gian tìm kiếm mẫu trong quá trình tìm tập hữu ích.

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

(a) Ma trận EUCS

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

(b) Ma trận EUCS'

**Hình 4.** Cải tiến ma trận EUCS

## V KẾT QUẢ THỰC NGHIỆM

Thực nghiệm được tiến hành đánh giá thuật toán HMiner với cải tiến cấu trúc dữ liệu phục vụ cho việc tìm kiếm mẫu (HMiner2). Chương trình được viết bằng ngôn ngữ Java chạy trên máy tính Intel Core i5 1.6GHz, bộ nhớ 4GB và hệ điều hành Windows 10. Dữ liệu thực nghiệm được thể hiện trong Bảng 5. Các giá trị hữu

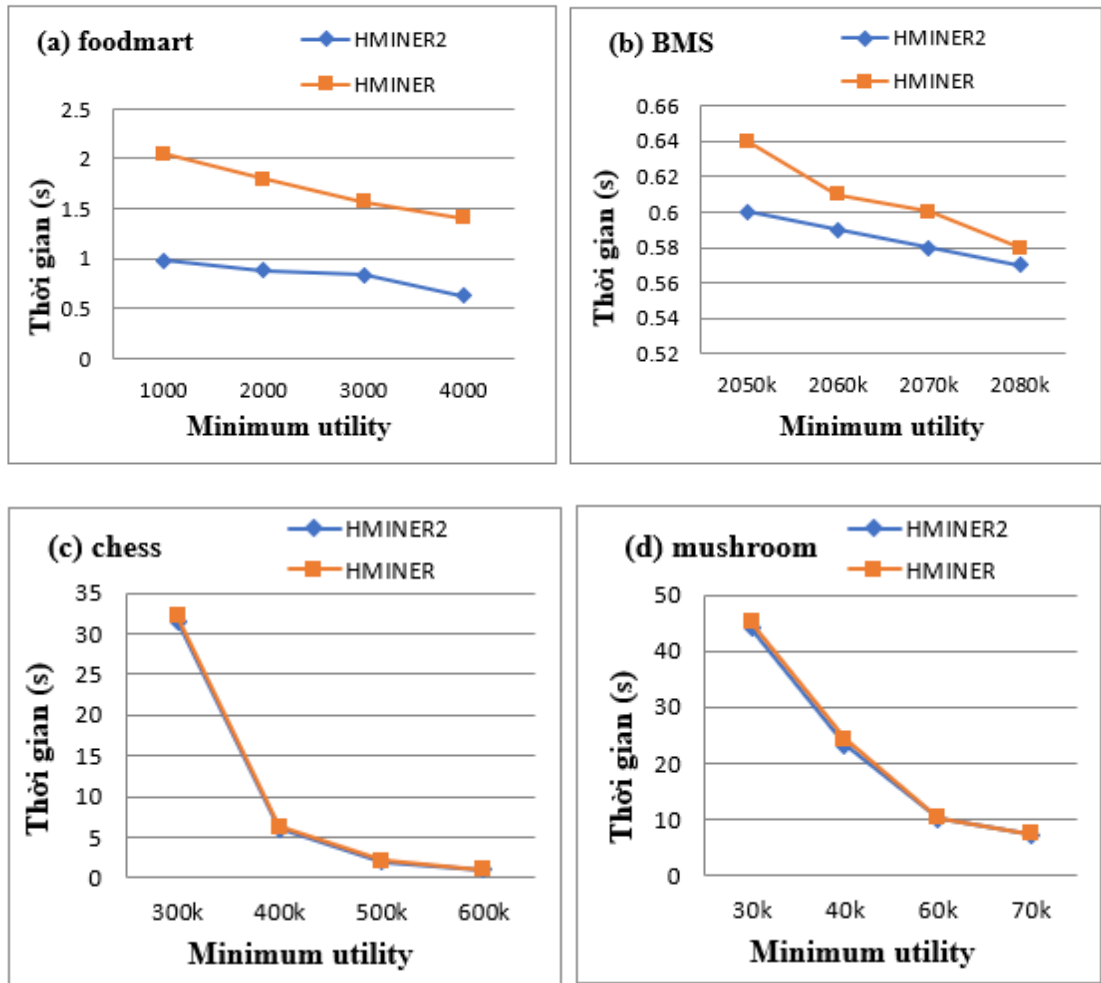
ích ngoại cho các item được phát sinh trong phạm vi từ 1.0 đến 10.0 và các giá trị hữu ích nội cho các item được phát sinh trong phạm vi từ 1 đến 10.

**Bảng 5.** Dữ liệu dùng cho thực nghiệm

<b>Tập dữ liệu</b>	<b>Số giao tác</b>	<b>Số item</b>	<b>Độ dài trung bình</b>	<b>Loại dữ liệu</b>
foodmart	4141	1559	4.4	Thưa
BMS	59601	497	4.8	Thưa
mushroom	8124	119	23	Đặc
chess	3196	75	37	Rất đặc

#### **A Thời gian thực hiện**

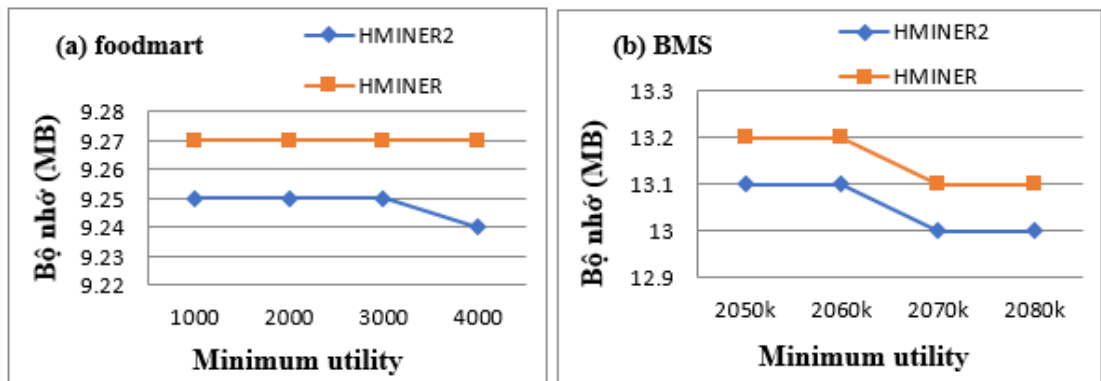
Hình 5 cho thấy HMiner2 thực hiện nhanh hơn so với HMiner trên hầu hết các tập dữ liệu, đặc biệt là đối với những tập dữ liệu thưa.

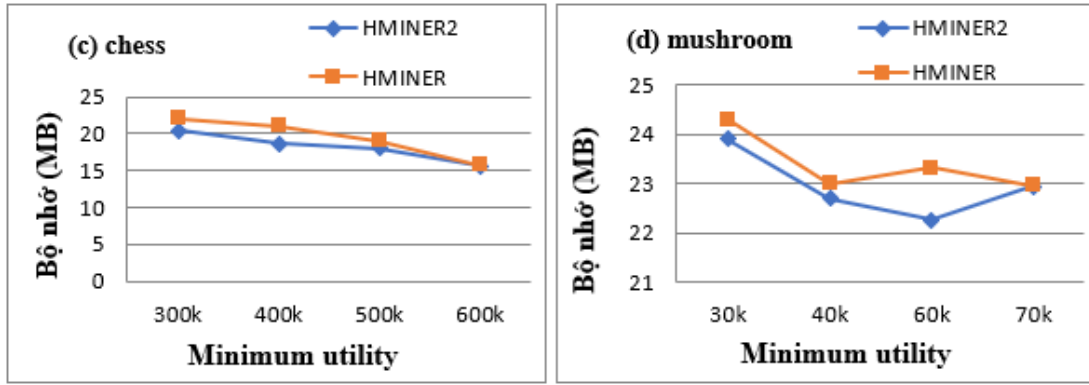


**Hình 5.** So sánh thời gian thực hiện với các giá trị *minutil*.

## 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 HMiner2 cũng cho kết quả tốt hơn đối với tập dữ liệu thưa. Hình 6 thể hiện so sánh về bộ nhớ sử dụng của các thuật toán trên các tập dữ liệu với các giá trị *minutil* khác nhau.





**Hình 6.** So sánh bộ nhớ sử dụng với các giá trị *minutil*.

## VI KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

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 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 tìm kiếm 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 khác nhằm hiệu quả hơn trong khai thác tập hữu ích cao.

## VII 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] S. Krishnamoorthy, "HMiner: Efficiently mining high utility itemsets," *Expert Systems with Applications*, vol. 90, p. 168–183, 2017.
- [3] 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," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 595–625, 2017.
- [4] 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.

- [5] J. Liu, K. Wang, B.C.M. Fung, "Direct discovery of high utility itemsets without candidate generation," *Proc. of the 2012 IEEE Int'l Conf. on Data Mining (ICDM 2012)*, p. 984–989, 2012.
- [6] 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.
- [7] 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.
- [8] 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.
- [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] 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.
- [11] V.S. Tseng, B.-E. Shie, C.-W. Wu, P.S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Trans. Knowl. Data Eng. 25 (8)*, p. 1772–1786, 2013.
- [12] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Trans. Knowl. Data Eng. 21 (12)*, p. 1708–1721, 2009.
- [13] U. Yun, J. Kim, "A fast perturbation algorithm using tree structure for privacy preserving utility mining," *Expert Syst. Appl. 42 (3)*, p. 1149–1165, 2015.
- [14] C.-W. Wu, P. Fournier-Viger, P.S. Yu, V.S. Tseng, "Efficient mining of a concise and loss-less representation of high utility itemsets," *The 11th IEEE Int'l Conf. on Data Mining (ICDM 2011)*, p. 824–833, 2011.
- [15] M.-Y. Lin, T.-F. Tu, S.-C. Hsueh, "High utility pattern mining using the maximal itemset property and lexicographic tree structures," *Inf. Sci. 215*, p. 1–14, 2012.
- [16] C.-W. Lin, G.-C. Lan, T.-P. Hong, "An incremental mining algorithm for high utility itemsets," *Expert Syst. Appl. 39 (8)*, p. 7173–7180, 2012.