TẠP CHÍ KHOA HỌC ĐẠI HỌC ĐÀ LẠT Tập 10, Số 3, 2020 25-38

KHAI THÁC TẬP MỤC LỢI ÍCH CAO CÓ LỢI NHUẬN ÂM TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN DỌC

Cao Tùng Anh[a*], Ngô Quốc Huy[a], Võ Hoàng Khang[a]

[a]Khoa Công nghệ Thông tin, Trường Đại học Công nghệ TP.HCM, TP. Hồ Chí Minh, Việt Nam
[*]Tác giả liên hệ: Email: ct.anh@hutech.edu.vn

Lịch sử bài báo
Nhận ngày 27 tháng 02 năm 2020
Chỉnh sửa ngày 24 tháng 6 năm 2020 | Chấp nhận đăng ngày 24 tháng 9 năm 2020

Tóm tắt

Tập lợi ích cao (TLIC) là một vấn đề quan trọng trong khai phá dữ liệu, xem xét các lợi ích của các mục (chẳng hạn như lợi nhuận và lãi suất) được khám phá từ cơ sở dữ liệu (CSDL) giao dịch hỗ trợ cho việc kinh doanh của các đơn vị. Bài báo trình bày một phương pháp khai thác tập lợi ích cao có lợi nhuận âm trên CSDL phân tán dọc. Việc khai thác tập lợi ích cao đã được nghiên cứu và công bố rộng rãi trong những năm gần đây. Có nhiều thuật toán khai thác các tập lợi ích cao (TLIC) bằng cách cắt tỉa các ứng cử viên dựa trên các giá trị lợi ích và dựa trên các giá trị sử dụng có trọng số giao dịch. Các thuật toán này đều hướng tới mục đích làm giảm không gian tìm kiếm. Trong bài báo này, chúng tôi đề xuất một phương pháp khai thác tập lợi ích cao có lợi nhuận âm (TLIC-TSA) từ CSDL phân tán dọc. Phương pháp này không tích hợp CSDL từ CSDL cục bộ của các bên tham gia để hình thành CSDL tập trung và chỉ thực hiện việc quét các CSDL mỗi bên tham gia một lần. Các thí nghiệm cho thấy thời gian chạy của phương pháp này hiệu quả hơn so với khai thác trên cơ sở dữ liệu tập trung.

Từ khóa: Cơ sở dữ liệu; Cơ sở dữ liệu phân tán dọc; Khai thác dữ liệu; Lợi nhuận âm; Tập lợi ích cao.

DOI: http://dx.doi.org/10.37569/DalatUniversity.10.3.666(2020)

Loại bài báo: Bài báo nghiên cứu gốc có bình duyệt

EXPLOIT MINING HIGH UTILITY ITEMSETS WITH NEGATIVE UNIT PROFITS FROM VERTICALLY DISTRIBUTED DATABASES

Cao Tung Anh[a*], Ngo Quoc Huy[a], Vo Hoang Khang[a]

[a]The Faculty of Information Technology, Ho Chi Minh City University of Technology, Hochiminh City, Vietnam

[*]Corresponding author: Email: ct.anh@hutech.edu.vn

Article history

Received: February 27th, 2020

Received in revised form: June 24th, 2020 | Accepted: September 24th, 2020

Abstract

High Utility Itemset (HUI) mining is an important problem in the data mining literature that considers the utilities for businesses of items (such as profits and margins) that are discovered from transactional databases. There are many algorithms for mining high utility itemsets (HUIs) by pruning candidates based on estimated and transaction-weighted utilization values. These algorithms aim to reduce the search space. In this paper, we propose a method for mining HUIs with negative unit profits from vertically distributed databases. This method does not integrate databases from the relevant local databases to form a centralized database. Experiments show that the run-time of this method is more efficient than that of the centralized database.

Keywords: Data mining; Database; High utility itemset; Negative unit profits; Vertically distributed databases.

DOI: http://dx.doi.org/10.37569/DalatUniversity.10.3.666(2020)

Article type: (peer-reviewed) Full-length research article

1. GIỚI THIỆU

Khai thác các tập lợi ích cao (TLIC) là hình thức chung của việc khai thác các tập thuộc tính thường xuyên (TMTX) (Agrawal & Shafer, 1996). Nó nhằm mục đích tìm các tập lợi ích cao từ cơ sở dữ liệu. Tuy nhiên, nó không giống như khai thác TMTX, TLIC không đáp ứng các tính chất của Apriori, đó là tập hợp con của TLIC không có khả năng là TLIC. Do đó, chúng tôi không thể sử dụng đầy đủ các thuật toán của TMTX cho khai thác TLIC.

Năm 2004, Yao, Hamilton, và Butz (2004) đã đề xuất mô hình khai thác TLIC. Họ đã đề xuất thuật toán UMining và UMining_H (UMining với heuristic) để tìm TLIC (Yao & Hamilton, 2006).

Gần đây, một số thuật toán dựa trên việc sử dụng trọng số giao dịch (TWU) đã được phát triển (Erwin, Gopalan & Achuthan, 2007a, 2007b; Le, Nguyen, Cao & Vo, 2009; Liu, Liao, & Choudhary, 2005). Trước tiên, thuật toán hai pha (*Two-Phase*) được đề xuất bởi Liu và ctg. (2005). Sau đó, một số thuật toán hiệu quả đã được đề xuất (Erwin và ctg., 2007b), chúng dựa trên các phương pháp không tạo ra các ứng cử viên để khai thác TLIC. Trong Vo, Nguyen, và Le (2009), các tác giả đã đề xuất WIT-tree, một cấu trúc dữ liệu mới và một thuật toán hiệu quả để khai thác TLIC.

Mặc dù có nhiều thuật toán để khai thác TLIC, nhưng chưa có mô hình khai thác tập lợi ích cao có lợi nhuận âm trên cơ sở dữ liệu (CSDL) phân tán dọc. Ngày nay, do việc cạnh tranh giữa các công ty trở nên ngày càng gay gắt, các chiến dịch khuyến mãi cùng với vô vàn ưu đãi tối đa cho người dùng, với mục tiêu là kích cầu mua hàng, vì thế một số sản phẩm khuyến mãi đính kèm sản phẩm chính không tránh đến việc lỗ và tạo ra khoản đơn vị lợi nhuận âm. Ngoài ra các công ty cũng có thể hạ giá bán thấp hơn giá mua của một số sản phẩm để thu hồi vốn và từ đó phát sinh các đơn vị lợi nhuận âm.

Từ thực tế nghiên cứu, trong bài báo này, chúng tôi đề xuất phương pháp khai thác TLIC-TSA (tập lợi ích cao có lợi nhuận âm) trên CSDL phân tán dọc. Những đóng góp chính của bài viết này như sau:

- Chúng tôi đề xuất một mô hình chung để khai thác TLIC-TSA từ cơ sở dữ liệu phân tán dọc;

- Với phương pháp đề xuất (TLIC-TSA) chỉ thực hiện quét CSDL cục bộ của các bên tham gia một lần và không cần tích hợp các CSDL của nhiều bên thành CSDL tập trung. Điều này nhằm giảm thời gian khai thác theo phương pháp cũ và giảm yêu cầu bộ nhớ tại bên tiến hành khai thác.

Phần còn lại của bài báo này được tổ chức như sau: Phần 2 trình bày nền tảng lý thuyết và một số phương pháp hiện tại để giải quyết vấn đề khai thác TLIC. Mô hình cho TLIC-TSA trong cơ sở dữ liệu phân tán dọc được trình bày trong phần 3, trong phần này, chúng tôi cũng thảo luận về cách hoạt động của MasterSite, SlaverSite và cách chúng trao đổi thông tin với nhau. Phần 4 cung cấp kết quả thử nghiệm và đánh giá hiệu suất của

chiến lược được đề xuất. Cuối cùng, chúng tôi trình bày kết luận và công việc trong tương lai trong phần 5.

2. CÁC NGHIÊN CỨU LIÊN QUAN

Trong những năm gần đây, nhiều thuật toán TLIC đã được đề xuất (Le và ctg., 2009; Liu và ctg., 2005; Yao và ctg., 2004). Tính hữu dụng của một tập thuộc tính được đặc trưng như một ràng buộc lợi ích. Nghĩa là, một tập thuộc tính chỉ thú vị với người dùng nếu lợi ích của nó thỏa mãn một ràng buộc lợi ích nhất định (*minutil*).

Thuật toán FHN: được đề xuất bởi Lin, Fournier-Viger, và Gan (2016). Ý tưởng chính của thuật toán là dựa trên cấu trúc danh sách lợi ích dương và âm để khai thác hiệu quả các nhóm lợi ích cao, đồng thời xem xét cả lợi nhuận đơn vị dương và âm. Tính hữu ích của một tập thuộc tính được tính bằng các giá trị giao dịch và lợi ích của tập thuộc tính. Giá trị giao dịch của một tập thuộc tính, ký hiệu là *xpq*, là giá trị của một thuộc tính được liên kết với một tập thuộc tính *ip* trong một *tq* giao dịch.

Giá trị lợi ích của một mặt hàng, ký hiệu là *yp*, là một số thực được chỉ định bởi người dùng sao cho hai mục i_p và i_q, y_p lớn hơn y_q nếu người dùng thích i_p của tập thuộc tính hơn i_q.

Vấn đề khai thác tập thuộc tính dựa trên lợi ích là khám phá tập *H* của tất cả các TLIC, TLIC = {*S* | *S* ⊆ *I*, *u*(*S*) ⩾ minutil}

$$u(S) = \sum_{i_p \in S} \sum_{t_q \in T_s} f(x_{pq}, y_p) \quad (1)$$

Trong đó f (x_{pq}, y_p) = $x_{pq} \cdot y_p$ và T_s là tập hợp các giao dịch có chứa các mục *S*.

2.1. Phương pháp giá trị lợi ích ước tính

Yao và ctg. (2004) đã làm giảm không gian tìm kiếm bằng cách cắt tỉa các ứng cử viên dựa trên giá trị lợi ích ước tính. Lợi ích của một tập thuộc tính S^k luôn nhỏ hơn hoặc bằng giới hạn trên của lợi ích S^k và dựa trên lợi ích giới hạn trên của S^k, Yao và Hamilton đã đề xuất thuật toán UMining (Yao & Hamilton, 2006) để khai thác tất cả các tập lợi ích cao.

2.2. Các công thức

Liu và ctg. (2005) đã giảm không gian tìm kiếm bằng cách cắt tỉa các ứng cử viên dựa trên giá trị sử dụng trọng số giao dịch (TWU). Lợi ích của một vật phẩm *S* luôn nhỏ hơn hoặc bằng giá trị *TWU* của *S*.

$$TWU(S) = tu(T_s) = \sum_{t_q \in T_s} tu(t_q) = \sum_{t_q \in T_s} \sum_{i_p \in t_q} f(x_{pq}, y_p) \quad (2)$$

$$u(S) = \sum_{t_q \in T_s} \sum_{i_p \in S} f(x_{pq}, y_p) \leq \sum_{t_q \in T_s} \sum_{i_p \in t_q} f(x_{pq}, y_p) = TWU(S) \quad (3)$$

$$TUW(S^{k-1}) = \sum_{t_q \in T_{S^{k-1}}} tu(t_q) \geq \sum_{t_q \in T_{S^k}} tu(t_q) = TWU(S^k) \quad (4)$$

Cách tính này sẽ được áp dụng trong tính toán của chúng tôi khi dữ liệu của các bên được gửi về một máy để khai thác.

Erwin và ctg. (2007a) đã đề xuất các thuật toán hiệu quả bằng cách sử dụng phương pháp tăng trưởng mẫu. Họ đã phát triển một biểu diễn dữ liệu nhỏ gọn mới có tên là cây nén lợi ích mở rộng cây CFP (Gopalan & Sucahyo, 2004) để khai thác TLIC và thuật toán mới có tên CTU-PRO.

Zida, Fournier-Viger, Lin, Wu, và Tseng (2017) đã đề xuất thuật toán giải quyết tốc độ chạy, một nghiên cứu thử nghiệm trên nhiều bộ dữ liệu khác nhau cho thấy rằng EFIM nói chung nhanh hơn hai đến ba bậc so với các thuật toán hiện đại d2HUP, HUI-Miner, HUP-Miner, FHM, và UP-Growth+ trên các bộ dữ liệu dày đặc và hoạt động khá tốt trên các tập dữ liệu thưa thớt tuy nhiên nó chưa so sánh với TWU và TWU cũng nhanh hơn những thuật toán trên, EFIM so với HUI-miner nhanh ít hơn so với TWU, TWU nhanh hơn 3 lần, HUI trong khi EFIM chỉ nhanh hơn 2,5 lần, nên chúng tôi chọn TWU.

Khái niệm TWU được sử dụng để cắt xén không gian tìm kiếm trong CTU-PRO, nhưng nó phải quét lại cơ sở dữ liệu để xác định lợi ích thực tế của các mục TWU cao. Thuật toán tạo cây CUP có tên GlobalCUP-Tree từ CSDL giao dịch sau lần đầu tiên xác định các mục TWU cao riêng lẻ. Đối với mỗi mục TWU cao, một cây chiếu nhỏ hơn có tên LocalCUP-Tree được trích xuất từ cây GlobalCUP để khai thác tất cả các TLIC bắt đầu với mục đó làm tiền tố.

Le và ctg. (2009) đã đề xuất cấu trúc dữ liệu cây WIT (WIT-TREE) và thuật toán khai thác TLIC (thuật toán khai thác TWU), thuật toán này đã cải tiến thời gian khai thác, nhằm tính nhanh TWU và độ có ích của itemset. Chúng tôi nhận thấy thuật toán này phù hợp để khai thác TLIC-TSA trong cơ sở dữ liệu phân tán dọc.

Cấu trúc dữ liệu cây WIT:

Đỉnh: ký hiệu: $Xx_{twu(X)}^{Tidset}$, bao gồm ba trường: Mục dữ liệu *X*, Tidset: bộ giao dịch chứa *X*, và *twu*: Tổng trọng số giao dịch của *X*.

Giá trị của *TWU(X)* được tính bằng cách tổng hợp tất cả các giá trị TWU của các giao dịch mà giá trị của chúng được chứa trong Tidset. Do đó, việc tính toán *TWU(X)* và *u(X)* sẽ được thực hiện nhanh chóng bằng cách sử dụng Tidset.

Cung: Kết nối đỉnh ở cấp thứ *k* (gọi là *X*) với đỉnh tại cấp thứ *k + 1* (gọi là *Y*) trong đó $X \equiv \theta_k Y$.

3. DỮ LIỆU (HOẶC VẬT LIỆU) VÀ PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Đặt vấn đề

Một siêu thị đã bán *n* mặt hàng $I = \{i_1, i_2, \dots i_n\}$ vì cần chuyên môn hóa, siêu thị cần lưu trữ thông tin giao dịch trong *k* máy tính (*k* chi nhánh), tức là mỗi chi nhánh lưu trữ thông tin của các mặt hàng (bộ sản phẩm). Chúng ta có thể hình thành như sau:

Cơ sở dữ liệu *D* được chia thành *n* (chi nhánh tham gia) chi nhánh {D_1, D_2 …, D_n}, trong đó D_j chứa tập hợp các mục $J = \{i_{j_1}, i_{j_2}, \dots i_{j_v}\}$ (*v* là số mục dữ liệu của chi nhánh D_j), các giao dịch trong D_j chỉ chứa mục chứa trong *ij*. Giả sử rằng $Ii \cap Ij = \emptyset$, $\forall i \neq j$ và $\bigcup_{j=1}^{k} I_j = I$. Khi mỗi giao dịch được tạo, có ID giao dịch mới, các mặt hàng được mua và số lượng mặt hàng được cập nhật trong các chi nhánh tương ứng. Do đó, nó không phải là CSDL tập trung, làm cho siêu thị dễ quản lý và không bị quá tải trong trường hợp lượng dữ liệu khổng lồ.

Vấn đề là làm thế nào để khai thác TLIC từ CSDL của nhiều chi nhánh mà không tích hợp (thực hiện phép kết) chúng lại với nhau thành một CSDL tập trung (cơ sở dữ liệu rất lớn trong trường hợp tích hợp tất cả các chi nhánh lại với nhau)?

Ví dụ: Giả sử ta có các CSDL giao dịch của một siêu thị như trong Bảng 1.

Bảng 1. Dữ liệu giao dịch

Tập thuộc tính TID	A	B	C	D	E	F	G	H
T1	1	0	1	1	0	0	0	3
T2	2	0	6	0	2	0	5	0
T3	1	2	1	5	1	5	0	0
T4	2	4	3	3	1	0	0	0
T5	0	1	2	0	1	0	2	0

Nhưng trong thực tế, dữ liệu giao dịch, các mục và lợi nhuận có trọng số âm của các mục lại được được chia và lưu trữ tại ba chi nhánh (ở ba địa điểm khác nhau) như trong các Bảng 2, 3, và 4.

Bảng 2. Dữ liệu giao dịch của chi nhánh 1

	A	B	C	D
T1	1	0	1	1
T2	2	0	6	0
T3	1	2	1	5
T4	2	4	3	3
T5	0	1	2	0

Tập thuộc tính	Lợi ích
A	-1
B	2
C	1
D	2

Bảng 3. Dữ liệu giao dịch của chi nhánh 2

	E	F
T1	0	0
T2	2	0
T3	1	5
T4	1	0
T5	1	0

Tập thuộc tính	Lợi ích
E	3
F	-1

Bảng 4. Dữ liệu giao dịch của chi nhánh 3

	G	H
T_1	0	3
T_2	5	0
T_3	0	0
T_4	0	0
T_5	2	0

Tập thuộc tính	Lợi ích
G	-1
H	5

Yêu cầu khai thác là từ dữ liệu của ba chi nhánh này khai thác ra tập lợi ích cao của CSDL tập trung toàn cục.

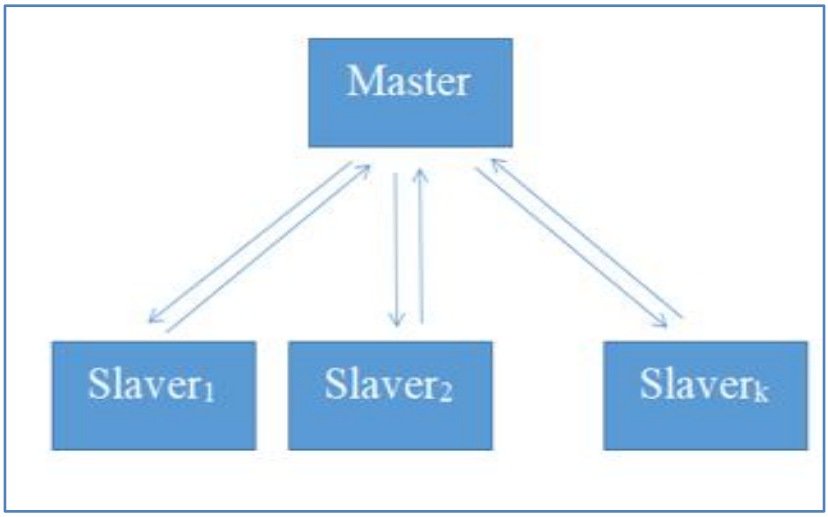
3.2. Mô hình khai thác

Bước 1: MasterSite (MS) gửi yêu cầu khai thác tới tất cả chi nhánh (tên của CSDL sẽ khai thác, *minutil*) và chờ thông tin từ các chi nhánh.

Bước 2: SlaverSite (SS) nhận được thông tin yêu cầu từ MasterSite. SlaverSite sẽ tính toán các thông tin cần thiết và gửi đến MasterSite. Trình tự các bước như sau:

- Nhận yêu cầu khai thác, *minutil*.
- Tính tổng lợi ích của tất cả các giao dịch theo dữ liệu tại chi nhánh (*TWU*(*Ti*,*j*)) với *i* là giao dịch thứ *i* và *j* là chi nhánh. Tính tập giao dịch của từng mục dữ liệu (*tidset*) và độ lợi ích của từng mục dữ liệu trong tất cả các giao dịch trên CSDL cục bộ.
- Gửi thông tin đến MasterSite.

Bước 3: Khi nhận được đủ thông tin từ tất cả các chi nhánh, MS sẽ khai thác TLIC-TSA bằng cách gọi thuật toán TWU-Mining (Le và ctg., 2009). Sau khi có kết quả khai thác, MS sẽ gửi tập lợi ích cao toàn cục cho tất cả các chi nhánh.



Hình 1. Mô hình khai thác TLIC-TSA

Áp dụng mô hình khai thác (Hình 1) cho dữ liệu minh họa:

Bước 1: MS gửi yêu cầu khai thác là CSDL của các chi nhánh như trong các Bảng 2, 3, và 4 với *minutil* = 30 đến ba chi nhánh.

Bước 2: Tại các chi nhánh tiến hành khai thác bằng cách: tính tidset và giá trị lợi ích cao của từng mục đơn. Tiếp đó tính tổng lợi nhuận của từng giao dịch tại CSDL cục bộ của từng chi nhánh. Ví dụ tại chi nhánh 1 tính tidset và giá trị lợi ích cao của mặt hàng B ta có: B x 345/14, trong đó mặt hàng B xuất hiện tại các giao dịch 3, 4, 5 và tổng lợi ích của B trong các giao dịch là 14. Tổng lợi nhuận của giao dịch T_1, T_2 tại chi nhánh 1 là:

$TWU(T_1,1) = 1 \text{ x } (-1) + 1 \text{ x } (1) + 1 \text{ x } (2) = 2.$

$TWU(T_2,1) = 2 \text{ x } (-1) + 6 \text{ x } (1) = 4.$

Tương tự, tính tidset và tổng lợi ích của tất các mặt hàng (mục dữ liệu đơn) và tổng lợi ích của các giao dịch tại chi các chi nhánh ta có Bảng 5, 6, và 7:

Bảng 5. Kết quả tính toán tại chi nhánh 1

A	Tdset	1	2	3	4	
	Lợi ích	-1	-2	-1	-2	
B	Tdset	3	4	5		
	Lợi ích	4	8	2		
C	Tdset	1	2	3	4	5
	Lợi ích	1	6	1	3	2
D	Tdset	1	3	4		
	Lợi ích	2	10	6		

TID	TWU
T1	2
T2	4
T3	14
T4	15
T5	4

Bảng 6. Kết quả tính toán tại chi nhánh 2

E	Tdset	2	3	4	5
	Lợi ích	6	3	3	3
F	Tdset	3			
	Lợi ích	-5			

TID	TWU
T_1	0
T_2	6
T_3	-2
T_4	3
T_5	3

Bảng 7. Kết quả tính toán tại chi nhánh 3

G	Tdset	2	5
	Lợi ích	-5	-2
H	Tdset	1	
	Lợi ích	15	

TID	TWU
T_1	15
T_2	-5
T_3	0
T_4	0
T_5	-2

Sau đó các chi nhánh gửi các kết quả đã tính toán được cho bên Master.

Bước 3: Tại MS, sau khi nhận được thông tin từ các chi nhánh (*n* chi nhánh), MS tính tổng lợi nhuận của các giao dịch (T_i) dựa trên kết quả từ CSDL cục bộ của các chi nhánh (*j*). Ta có Bảng 8 chứa tổng lợi ích của các giao dịch.

$$TWU(T_i) = \sum_{j\in[1,n]} TWU(T_{ij}) \quad (5)$$

Bảng 8. Tổng lợi ích của các giao dịch từ các CSDL cục bộ

TID	TWU
T_1	17
T_2	5
T_3	12
T_4	18
T_5	5

Để tính giá trị *TWU*(*X*) toàn cục của mục dữ liệu *X*, MS sẽ tính bằng tổng các giá trị *TWU* của các giao tác mà *tid* của chúng chứa trong Tidset với giá trị lợi nhuận dương.

Ví dụ: *TWU*(*B*) = *TWU*(*T3*) + *TWU*(*T4*) + *TWU*(*T5*) = 35

TWU(*D*) = *TWU*(*T1*) + *TWU*(*T3*) + *TWU*(*T4*) = 47

Từ các tính toán *TWU* và so sánh với *minutil* = 30, MS sẽ xây dựng ra mức thứ nhất của WIT-Tree (như Hình 2).

MS tiếp tục quá trình khai thác TLIC-TSA dựa trên thuật toán TWU-Mining với *minutil* = 30.

Xét nút {A} kết hợp với {B}, ta có itemset mới AB x 34 với *TWU*(AB) = 18 + 20 = 38.

Kết hợp với C, ta có itemset mới AC x 1234 với *TWU*(AC) = 18 + 12 + 18 + 20 = 68.

Kết hợp với D, ta có itemset mới AD x 134 với *TWU*(AD) = 18 + 18 + 20 = 56.

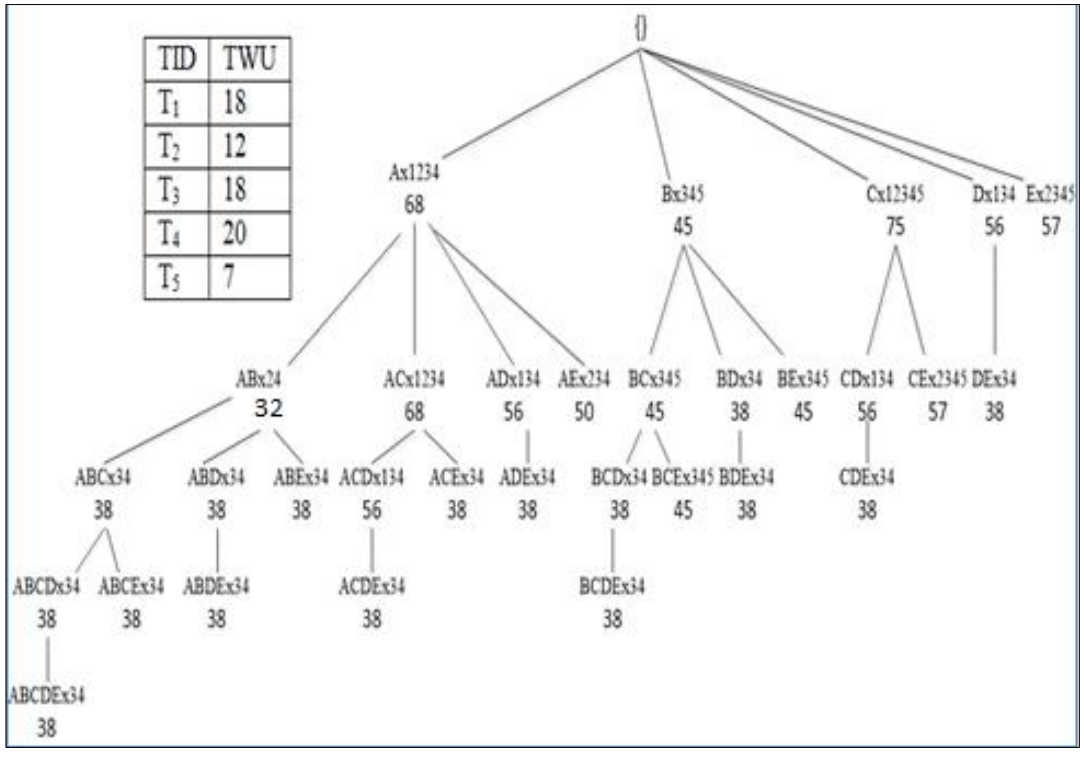
Kết hợp với E, ta có itemset mới AE x 234 với *TWU*(AE) = 12 + 18 + 20 = 50.

Kết hợp với BD, ta có itemset mới ABD x 34 với *TWU*(ABD) = 18 + 20 = 38.

Kết hợp với {ABE}, ta có itemset mới ABDE x 34 với *TWU*(ABDE) = 18 + 20 = 38

Tiếp đó, tính *u*(ABDE) = 31, thỏa *minutil*, vì vậy thêm vào TLIC - TSA, TLIC - TSA = {ABDE}. Làm tương tự cho các bước tiếp theo (chi tiết trong Hình 2).

Sau khi MS tính toán xong, ta có tập lợi ích cao có lợi nhuận âm TLIC - TSA = {BCD, BDE, ABDE, BCDE, ABCDE}. MS sẽ gửi kết quả này cho tất cả các bên.



Hình 2. WIT-tree áp dụng cho TLIC-TSA

4. THỰC NGHIỆM

Ngôn ngữ thực nghiệm chúng tôi sử dụng là ngôn ngữ *C#* phiên bản 2014. Cấu hình máy tính tại các bên là Intel 3.2GHz, bộ xử lý Core i5, Ram 8GB, hệ điều hành Window 10-64 bit. Chúng tôi cũng thực nghiệm với năm bên khác nhau để đo thời gian thực hiện. Thời gian đo được tính từ khi MS gửi yêu cầu khai thác cho các bên và được tính là tổng thời gian thực hiện ở tất cả các bên và ở MS. Thời gian truyền dữ liệu giữa các bên trong trường hợp này được coi là không đáng kể. Cơ sở dữ liệu thử nghiệm có các tính năng như (Bảng 9):

Bảng 9. Dữ liệu thực nghiệm

CSDL	#Giao dịch	#Tập thuộc tính	Ghi chú
BMS-POS	515597	1656	Chỉnh sửa
Retails	88162	16469	Chỉnh sửa
Accidents	340183	468	Chỉnh sửa

Chúng tôi đã sửa đổi dữ liệu thực nghiệm bằng cách thêm một cột giá trị (ngẫu nhiên trong phạm vi từ 1 đến 10) cho mỗi mục tương ứng với mỗi giao dịch.

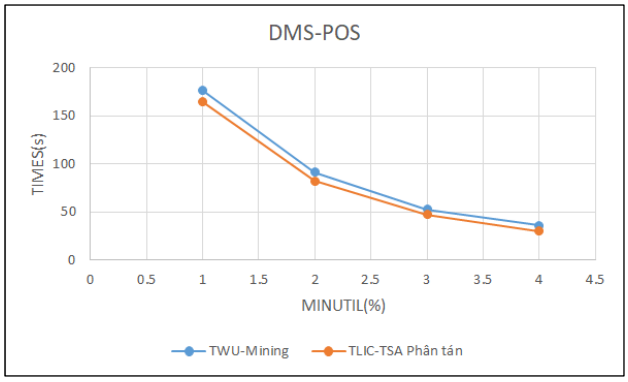
Chúng tôi tạo thêm một bảng để lưu trữ giá trị lợi ích của các tập thuộc tính, trong cột giá trị lợi ích có cả giá trị âm và dương (giá trị trong phạm vi từ 1 đến 10). Mỗi CSDL thực nghiệm được chia thành năm phần ngẫu nhiên xấp xỉ nhau và lưu trữ ở năm máy tích khác nhau trên cùng mạng cục bộ (Bảng 10).

Bảng 10. Kết quả thực nghiệm

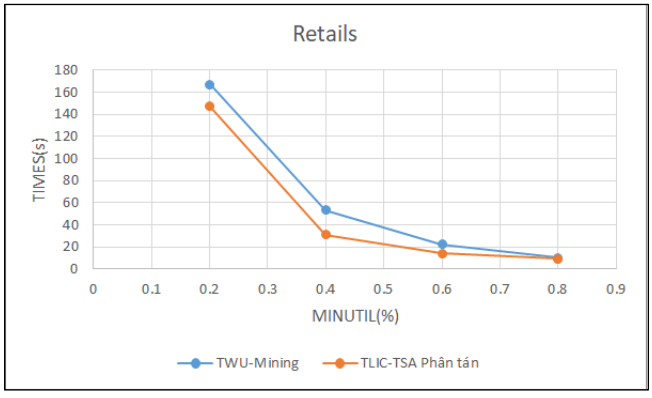
CSDL	Minutil **(%)**	TWU-Mining	TLIC-TSA Phân tán	#TLIC
BMS-POS	4.0	36.09	30.19	5
	3.0	52.75	47.64	6
	2.0	91.35	82.23	18
	1.0	176.46	164.33	142
Retails	0.8	10.43	9.25	24
	0.6	22.23	13.94	41
	0.4	53.44	31.26	59
	0.2	167.19	146.97	215
Accidents	0.8	12.34	10.26	3
	0.6	28.63	14.31	4
	0.4	62.25	49.69	11
	0.2	183.20	156.75	123

Trong Le và ctg. (2009), các tác giả đã thực nghiệm và cho kết quả TWU-Mining nhanh hơn các thuật toán dựa trên giới hạn trên lợi ích (Yao & Hamilton, 2006) và Two-

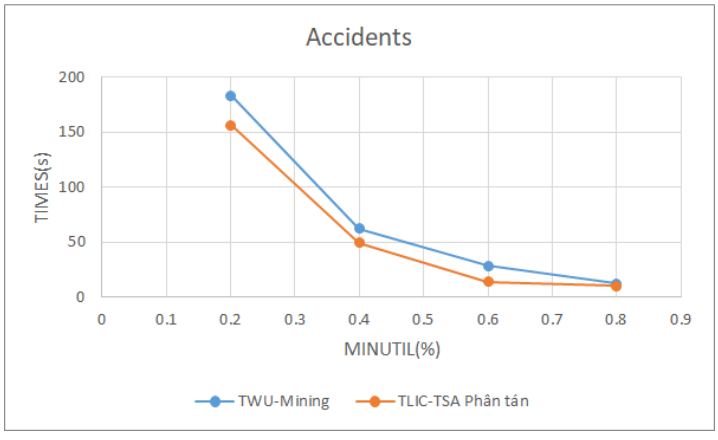
Phase (Liu và ctg., 2005), vì vậy chúng tôi sẽ so sánh mô hình đề xuất với TWU-Mining để đánh giá thời gian thực hiện.



Hình 3. Thời gian thực nghiệm trên CSDL BMS-POS



Hình 4. Thời gian thực nghiệm trên CSDL Retails



Hình 5. Thời gian thực nghiệm CSDL Accidents

Kết quả thực nghiệm trong Bảng 10, Hình 3, 4, và 5 cho thấy thời gian thực hiện phương pháp khai thác TLIC-TSA được đề xuất trên cơ sở dữ liệu phân tán dọc ít hơn thời gian thực hiện trên cơ sở dữ liệu tập trung. Do tính toán phân tán tại chi nhánh được thực hiện trước khi dữ liệu được gom tập trung về một nơi nên việc khai thác tập lợi ích

cao có lợi nhuận âm tại MasterSite tốn ít thời gian và bộ nhớ hơn so với thực hiện trên một CSDL tập trung lớn.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài báo này, chúng tôi đã trình bày mô hình khai thác TLIC-TSA có lợi nhuận âm từ cơ sở dữ liệu phân tán dọc và đã thực nghiệm để thấy được hiệu quả của mô hình đề xuất. Theo kỹ thuật cây WIT, thuật toán chỉ quét một lần cơ sở dữ liệu cục bộ của các bên tham gia sau đó tính toán và gửi kết quả cho MS. Do đó, việc khai thác tập thuộc tính lợi ích cao tại MasterSite tốn rất ít thời gian. Nếu như cộng cả thời gian tính toán của các bên lại thì kết quả thực nghiệm cũng cho thấy tổng thời gian là ít hơn so với TWU-Mining tại CSDL tập trung.

Tuy nhiên, chúng tôi mới chỉ đề cập đến khai thác tập lợi ích cao có lợi nhuận âm từ cơ sở dữ liệu phân tán dọc, một thuật toán hiệu quả để khai thác TLIC có lợi nhuận âm trong cơ sở dữ liệu phân tán ngang sẽ được thảo luận. Bên cạnh đó việc nghiên cứu bảo toàn tính riêng tư cho dữ liệu của các bên tham gia cũng sẽ được nghiên cứu trong thời gian tới.

TÀI LIỆU THAM KHẢO

Agrawal, R., & Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Transactions on knowledge and Data Engineering*, *8*(6), 962-969. http://doi.org/ 10.1109/69.553164.

Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2007a). *CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach*. Paper presented at The 7th IEEE International Conference on Computer and Information Technology (CIT 2007), Fukushima, Japan. http://doi.org/10.1109/CIT.2007.120.

Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2007b). A bottom-up projection based algorithm for mining high utility itemsets. In K. L. Ong, W. Li, & J. Gao (Eds.), *Proceedings of the 2nd international workshop on Integrating artificial intelligence and data mining - Volume 84* (pp. 3-11). Australian Computer Society Inc, Australia.

Gopalan, R. P., & Sucahyo, Y. G. (2004). *High performance frequent patterns extraction using compressed FP-tree*. Paper presented at The SIAM International Workshop on High Performance and Distributed Mining (HPDM), Orlando, USA.

Le, B., Nguyen, H., Cao, T. A., & Vo, B. (2009). *A novel algorithm for mining high utility itemsets*. Paper presented at The 2009 First Asian Conference on Intelligent Information and Database Systems, Donghoi, Quangbinh, Vietnam. http://doi.org/ 10.1109/ACIIDS.2009.55

Lin, J. C. W., Fournier-Viger, P., & Gan, W. (2016). FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits. *Knowledge-Based Systems*, *111*, 283-298. https://doi.org/10.1016/j.knosys.2016.08.022

Liu, Y., Liao, W. K., & Choudhary, A. (2005). A fast high utility itemsets mining algorithm. In G. Weiss, M. Saar-Tsechansky, B. Zadrozny (Eds), *Proceedings of*

the 1st international workshop on Utility-based data mining (pp. 90-99). Association for Computing Machinery, USA.

Vo, B., Nguyen, H., & Le, B. (2009). *Mining high utility itemsets from vertical distributed databases*. Paper presented at The 2009 IEEE-RIVF International Conference on Computing and Communication Technologies, Danang, Vietnam. http://doi.org/10.1109/RIVF.2009.5174650.

Yao, H., & Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering, 59*(3), 603-626. http://doi.org/10.1016/j.datak.2005.10.004

Yao, H., Hamilton, H. J., & Butz, C. J. (2004). A foundational approach to mining itemset utilities from databases. In M. W. Berry, U. Dayal, C. Kamath, & D. Skillicorn (Eds), *Proceedings of the 2004 SIAM International Conference on Data Mining* (pp. 482-486). Society for Industrial and Applied Mathematics, USA.

Zida, S., Fournier-Viger, P., Lin, J. C. W., Wu, C. W., & Tseng, V. S. (2017). EFIM: a fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems, 51*(2), 595-625. http://doi.org/10.1007/s10115-016-0986-0.