Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015
DOI: 10.15625/vap.2015.000170

MÔ HÌNH MỚI TRÊN CÂY NÉN CHO KHAI PHÁ TẬP MỤC LỢI ÍCH CAO

Đậu Hải Phong[1], Đoàn Văn Ban[2], Đỗ Thị Mai Hường[3]

[1] Khoa Toán và Tin học, Trường Đại học Thăng Long
[2] Phòng các hệ thống phần mềm tích hợp, Viện Công nghệ thông tin
[3] Khoa Công nghệ thông tin, Học viện Kỹ thuật Quân sự

phong4u@gmail.com, dvban@ioit.ac.vn, dohuong@gmail.com

TÓM TẮT: *Hiện nay, một trong những vấn đề được quan tâm trong khai phá dữ liệu là tìm kiếm tập lợi ích cao từ cơ sở dữ liệu lớn. Trong kỹ thuật tìm kiếm tập lợi ích cao thì cả giá trị lợi ích và số lượng khác nhau của từng phần tử trong giao dịch đều được xem xét. Một vấn đề khó khăn trong kỹ thuật này là số lượng các tập các ứng viên được sinh ra là rất lớn vì tập lợi ích cao không có tính chất đóng. Hầu hết các thuật toán khai phá tập lợi ích cao như: UP-Growth, Udepth, Two-Phase, PB, CTU-PRO,… đều sử dụng mô hình TWU (Transactions Weight Utility) để tỉa tập ứng viên. Trong bài báo này chúng tôi đề xuất mô hình CWU (Candidate Weight Utility) trên cây tiền tố nén mẫu lợi ích. Xây dựng thuật toán CTU-PRO$^+$ dựa trên thuật toán CTU-PRO và sử dụng mô hình chúng tôi đề xuất CWU. Kết quả thử nghiệm thuật toán CTU-PRO$^+$ cho thấy thời gian thực hiện với các thuật toán Two-Phase, CTU-PRO cho kết quả tốt hơn..*

Từ khóa: *Khai phá dữ liệu, tập lợi ích, tập phổ biến, CWU, TWU*

I. GIỚI THIỆU

Khai phá tập phổ biến là nhiệm vụ quan trọng trong khai phá tri thức và có ứng dụng rộng rãi trong kinh doanh, khoa học và các lĩnh vực khác. Mục tiêu của khai phá tập phổ biến là tìm ra các phần tử cùng xuất hiện với một tần suất lớn hơn một ngưỡng tối thiểu cho trước trong cơ sở dữ liệu giao dịch. Tuy nhiên, khai phá tập phổ biến vẫn tồn tại một số hạn chế như: các phần tử trong giao dịch có sự quan trọng ngang nhau và không xem xét số lượng của nó hay trọng lượng liên quan như giá hoặc lợi nhuận. Vì vậy, mô hình khai phá tập phổ biến chỉ phản ánh mối tương quan giữa các phần tử, và nó không phản ánh ý nghĩa của các mặt hàng. Nhưng trong thực tế số lượng và trọng lượng của các phần tử là rất quan trọng như trong bài toán tối đa hóa lợi ích. Để khắc phục những hạn chế trong khai phá tập phổ biến thì một số mô hình khai thác tập lợi ích cao [1][2][3][4][5],… Trong mô hình này cho phép người sử dụng đánh giá được tầm quan trọng của một tập phổ biến bằng giá trị lợi ích. Một tập phần tử được gọi là tập lợi ích cao khi giá trị lợi ích của nó lớn hơn một ngưỡng do người dùng định nghĩa trước.

Trong khai phá tập lợi ích cao thì các tập lợi ích không có tính chất đóng [6]. Tính chất này đảm bảo một tập là tập lợi ích thấp thì các tập chứa nó cũng là tập lợi ích thấp. Ví dụ, tập {AB} là tập lợi ích thấp thì tập {ABC} vẫn có thể là tập lợi ích cao. Điều này sẽ làm số lượng ứng cử viên được sinh ra rất lớn và tốn nhiều thời gian để kiểm tra các ứng viên. Trong một số thuật toán [1][3][4],... đều dựa trên hai bước là sinh ứng viên và kiểm tra ứng viên đó có khả năng sinh ra các tập lợi ích cao không với một số lần duyệt dữ liệu nhất định. Để tránh duyệt dữ liệu nhiều lần thì một số thuật toán dùng cấu trúc cây để tìm tập lợi ích cao như [9][10],... nhưng các thuật toán này tiêu tốn nhiều thời gian và không gian bộ nhớ để sinh ra các cây điều kiện của từng tiền tố. Năm 2005, Liu cùng nhóm tác giả đã đưa ra mô hình TWU[12] trong khai phá tập lợi ích cao để loại bớt tập ứng viên. Giá trị lợi ích của tất cả các phần tử trong giao dịch được tổng hợp như là lợi ích của giao dịch và lấy nó làm cận trên lợi ích của các tập phần tử trong giao dịch đó. Khi đó, lợi ích giao dịch có trọng số (viết tắt: TWU – Transactions Weight Utility) của một tập phần tử được định nghĩa là tổng các lợi ích của giao dịch có chứa tập đó. Trong thuật toán [12], Liu đã sử dụng mô hình TWU để loại đi các tập có TWU nhỏ hơn ngưỡng lợi ích tối thiểu cho trước để giảm số lượng tập ứng cử viên, sau đó duyệt lại cơ sở dữ liệu để xác định lợi ích thực tế của các tập và tiếp tục xác định khả năng các tập lớn hơn của nó có là tập lợi ích cao hay không để sinh tiếp các ứng cử viên.

Trong các thuật toán sử dụng mô hình TWU, thuật toán [4][5][13],... thực hiện tìm kiếm tập ứng viên theo chiều sâu. Ví dụ, xét tập phần tử I={A, B, C, D, E}, từ phần tử A có thể sinh ra các tập ứng viên {AB}, {ABC}, {ABCD}, {ABCDE},...; từ phần tử B sinh ra các tập ứng viên {BC}, {BCD}, {BCDE},... Ta thấy, các tập ứng viên sinh ra từ phần tử B không còn xuất hiện phần tử A, nhưng khi lấy TWU làm cận trên thì vẫn còn giá trị lợi ích của phần tử A. Điều này làm tăng số lượng tập ứng viên và chi phí kiểm tra các tập ứng viên.

Trong bài báo này, chúng tôi đề xuất mô hình CWU (Candidate Weight Utility) và thuật toán CTU-PRO$^+$ khai phá tập lợi ích cao dựa trên mô hình CWU để giảm số lượng tập ứng viên và thời gian thực hiện.

Nội dung tiếp theo của bài báo được tổ chức như sau. Phần II vấn đề liên quan đến khai phá tập lợi ích cao. Phần III đề xuất mô hình CWU. Phần IV thuật toán CTU-PRO$^+$. Phần V, trình bày kết quả đạt được và so sánh với các thuật toán khác. Phần VI kết luận.

II. CÁC VẤN ĐỀ LIÊN QUAN

Cho một cơ sở dữ liệu gồm các giao dịch Ti là D = {T1,T2,T3,…Tn}, các giao dịch được xác định duy nhất bởi Tid, I={i1,i2,i3,…in} là các phần tử (item) xuất hiện trong các giao dịch, $X \subseteq I$ là tập các phần tử (itemsets). Một tập X được gọi là tập k-phần tử khi số lượng phần tử của X là k.

Để thuận lợi trong giải thích các khái niệm, chúng tôi đưa ra một cơ sở dữ liệu giao dịch được biểu diễn dưới dạng bảng như Bảng 1. Bảng lợi ích ngoài của các phần tử được cho trong Bảng 2.

Bảng 1. Cơ sở dữ liệu giao dịch minh họa

Tid	Giao dịch						TU
	A	B	C	D	E	F	
1	2	0	1	1	0	0	80
2	2	1	1	0	0	0	195
3	0	0	1	1	10	0	110
4	0	1	0	0	15	0	225
5	1	0	1	0	0	1	37
6	2	0	0	1	10	0	105
7	2	0	0	0	8	1	62
8	1	1	0	1	2	0	205
9	1	0	0	1	10	0	95
10	1	1	0	0	5	0	185
Tổng	12	4	4	5	60	2	1299

Bảng 2. Bảng lợi ích của các phần tử

Item	A	B	C	D	E	F
Lợi ích	10	150	25	35	5	2

Định nghĩa 1 [5] - Lợi ích trong (internal utility) của mỗi phần tử là giá trị của mỗi phần tử trong từng giao dịch. Ký hiệu: $O(i_k,T_j)$ – là lợi ích trong của phần tử i_k trong giao dịch T_j.

Ví dụ, $O(A,T_1) = 2$; $O(C,T_1) = 1$ trong Bảng 1.

Định nghĩa 2 [5] - Lợi ích ngoài (external utility) của mỗi phần tử là giá trị lợi ích của mỗi phần tử trong bảng lợi ích. Ký hiệu: $S(\{i_k\})$ là lợi ích ngoài của phần tử i_k.

Ví dụ, $S(\{A\}) = 10$; $S(\{B\}) = 150$ trong Bảng 2.

Định nghĩa 3 [5] - Lợi ích của một phần tử trong giao dịch là tích của lợi ích trong và lợi ích ngoài của phần tử đó. Ký hiệu: $U(i_k,T_j) = S(\{i_k\}) * O(i_k,T_j)$ là lợi ích của phần tử i_k trong giao dịch T_j.

Ví dụ, $U(\{A\},T_1) = 2*10 = 20$; $U(\{C\},T_1) = 1*25 = 25$,…

Định nghĩa 4 [5] - Lợi ích của một tập phần tử X trong một giao dịch T_j là tổng giá trị lợi ích tất cả phần tử của tập X trong giao dịch T_j. Ký hsiệu: $U(X,T_j) = \sum_{i_k \in X \wedge X \subseteq T_j} U(i_k, T_j)$ – là lợi ích của tập phần tử X trong một giao dịch T_j.

Ví dụ, $U(\{AC\},T_1) = 2*10 + 1*25 = 45$.

Định nghĩa 5 [5] - Lợi ích của một tập phần tử X trong cơ sở dữ liệu là tổng lợi ích của tập phần tử X trong tất cả giao dịch chứa X. Ký hiệu: $AU(X) = \sum_{T_j \in D \wedge X \subseteq T_j} U(X, T_j)$.

Ví dụ, xét tập{AC}, ta thấy {AC}, xuất hiện trong các giao dịch: T_1, T_5 nên ta có: $AU(\{AC\}) = U(\{AC\},T_1) + U(\{AC\},T_2) + U(\{AC\},T_5) = (2*10 + 1*25) + (2*10 + 1*25) + (1*10 + 1*25) = 160$.

Định nghĩa 6 [5] – Tập phần tử lợi ích cao: Tập X được gọi là tập phần tử lợi ích cao (HU – High Utility) nếu $AU(X) \geq \alpha$, ngược lại gọi X là tập phần tử lợi ích thấp. Trong đó α là ngưỡng lợi ích tối thiểu cho trước.

Ví dụ, lợi ích tối thiểu α = 150 thì {AC} là tập phần tử lợi ích cao.

Định nghĩa 7 [5] - Lợi ích của một giao dịch là tổng lợi ích của các phần tử trong giao dịch đó. Ký hiệu: $TU(T_j) = \sum_{i_k \in T_j} U(i_k, T_j)$ – là lợi ích của giao dịch T_j.

Ví dụ, $TU(T_1) = 2*10 + 1*25 + 1*35 = 80$, $TU(T_2) = 2*10 + 1*150 + 1*25 = 195$.

Định nghĩa 8 [5] - Lợi ích giao dịch có trọng số của một tập phần tử X là tổng lợi ích của các giao dịch có chứa tập phần tử X. Ký hiệu: $TWU(X) = \sum_{X \subseteq T_j \wedge T_j \in D} TU(T_j)$ là lợi ích giao dịch có trọng số của tập phần tử X.

Ví dụ: $TWU(\{AC\}) = TU(T_1) + TU(T_2) + TU(T_5) = 80 + 195 + 37 = 312$.

Hiện nay, đã có nhiều thuật toán khai phá tập lợi ích cao như CTU-Mine [11], HUC-Prune [9], UP-Growth [10], UDepth [4], PB [5], Two – Phase [3],... Thuật toán Two – Phase sử dụng mô hình TWU để loại bớt tập ứng viên, sau đó duyệt lại cơ sở dữ liệu để xác định lợi ích thực tế của các tập. Với phương pháp sinh ứng viên và kiểm tra ứng viên thì phải duyệt dữ liệu nhiều lần để tìm tập lợi ích cao.

Thuật toán UDepth [4] được Wei đưa ra thực hiện khai phá cơ sở dữ liệu theo chiều dọc. Thuật toán này gồm các bước: duyệt dữ liệu để xác định TWU của từng phần tử; loại bỏ các phần tử có TWU nhỏ hơn ngưỡng tối thiểu; sắp xếp lại các phần tử có TWU cao theo thứ tự giảm dần; từ mỗi phần tử i_k có TWU cao, tìm tất cả các tập có phần tử i_k là tiền tố và duyệt lại cơ sở dữ liệu một lần nữa để xác định các tập lợi ích cao. Năm 2013, Gou và cộng sự đưa ra thuật toán PB [5] dựa trên các bảng chỉ số để tăng tốc độ thực hiện và giảm yêu cầu bộ nhớ. Thuật toán này sử dụng bảng chỉ số của các tập để sinh các ứng viên, tìm tập lợi ích cao và tạo nhanh bảng chỉ số từ tập tiền tố của nó.

Để tiết kiệm chi phí trong việc sinh các tập ứng cử viên và giảm số lần duyệt cơ sở dữ liệu, một số thuật toán sử dụng cấu trúc cây đã được đề xuất như: CTU-Tree [11], HUC-Tree [14], UP-Growth [10] các thuật toán này gồm một số bước chính sau: xây dựng cây, sinh các ứng viên từ cây bằng thuật toán tăng trưởng mẫu, xác định các tập lợi ích cao từ các tập ứng viên. Với cách tiếp cận này thì thuật toán vẫn tốn nhiều bộ nhớ và thời gian duyệt cây. Trong các thuật toán sử dụng cấu trúc cây thì thuật toán CTU-PRO[13] đã sử dụng cây mẫu nén lợi ích (Compressed Utility Pattern tree -CUP-tree) là sự mở rộng của cây CFP cho kết quả tốt hơn trên cả bộ dữ liệu thưa và đậm đặc.

Hầu hết các thuật toán khai phá tập lợi ích cao đều sử dụng mô hình TWU làm cơ sở để cắt tỉa tập ứng viên. Trong mô hình TWU, đã sử dụng tổng lợi ích của tất cả các giao dịch chứa tập X làm cận trên để xem xét khả năng cao nhất của tập X có thể tạo ra tập lợi ích cao không. Ta thấy với một phần tử a và một tập phần tử {X}, ta có $TWU(\{aX\}) = \sum_{aX \subseteq T_j \wedge T_j \in D} TU(T_j)$ là cận trên của AU({aX}). Tương tự, có TWU({X}) là cận trên của AU({X}). Ta thấy {X} $\subseteq$ {aX} nên số giao dịch chứa {X} sẽ lớn hơn hoặc bằng số giao dịch chứa {aX}. Vậy, TWU({X}) là tổng lợi ích của các giao dịch chứa {X} sẽ lớn hơn hoặc bằng TWU({aX}) là tổng lợi ích của các giao dịch chứa {aX}.

Trong các thuật toán khai phá tập lợi ích cao UDepth [4], PB [5], CTU-PRO[13],... thì tập lợi ích cao được khai phá theo chiều sâu. Giả sử, {aX} là tất cả các tập có tiền tố là phần tử a, {bX} là tất cả các tập có tiền tố là phần tử b. Khi khai phá các tập trong {bX} sẽ không còn chứa phần tử a. Nhưng khi tính TWU({bX}) có thể vẫn gồm giá trị lợi ích của phần tử a. Điều này làm TWU({bX}) là cận trên của AU({bX}) lớn hơn mức cần thiết và khi dùng TWU({bX}) để tỉa các tập ứng viên sẽ không hiệu quả.

III. ĐỀ XUẤT MÔ HÌNH CWU

Từ những nhận xét trong phần 2, chúng tôi đề xuất mô hình CWU (Candidate Weight Utility) để khắc phục nhược điểm của mô hình TWU.

Định nghĩa 9 - Tập tiền tố của một phần tử It_x là tập các phần tử trong I mà đứng trước phần tử It_x: $ListItemPrefix(It_x) = \{j \in I \mid j \prec It_x\}$.

Định nghĩa 10 - Tiền tố của một tập Y là tập các phần tử trong I mà đứng trước phần tử đầu tiên của tập Y: $ListItemPrefix(Y) = \{ j \in I \mid j \prec y_1\}$, trong đó: y_1 là phần tử đầu tiên trong tập Y.

Định nghĩa 11 - Lợi ích ứng viên có trọng số (CWU – Candidate Weight Utility) của tập phần tử Y, ký hiệu là CWU(Y) được xác định như sau:

Đặt X = ListItemPrefix(Y), thì

$$CWU(Y) = \sum_{Y \subseteq T_j} TU(T_j) - \sum_{Y \subseteq T_j} \sum_{i_k \in X \wedge i_k \in T_j} U(i_k, T_j) \quad (1)$$

Nếu ListItemPrefix(Y) = {$\emptyset$} thì $\sum_{Y \subseteq T_j} \sum_{i_k \in X \wedge i_k \in T_j} U(i_k, T_j) = 0$.

Định nghĩa 12 - Nếu $CWU(Y) \geq \alpha'$ với α' là ngưỡng tối thiểu lợi ích ứng viên cho trước thì Y gọi là tập ứng viên lợi ích trọng số cao (HCWU- High Candidate Weight Utility). Ngược lại, Y được gọi là tập ứng viên lợi ích trọng số thấp (LCWU – Low Candidate Weight Utility).

Định lý 1 - Cho 2 tập Y_k – là tập k-phần tử, Y_{k-1} – (k-1)-phần tử và là tập tiền tố của tập Y_k. Nếu Y_k là HCWU thì Y_{k-1} cũng là HCWU.

Chứng minh:

Ta có, {aY_k} là tập các giao dịch chứa Y_k, {aY_{k-1}} là tập các giao dịch chứa Y_{k-1}. Khi Y_{k-1} là tập tiền tố của Y_k thì {aY_k} $\subseteq$ {aY_{k-1}}. Gọi X = ListItemPrefix(Y_{k-1}) = ListItemPrefix(Y_k), theo Định nghĩa 11 ta có:

$$CWU(Y_{k-1}) = \sum_{Y_{k-1} \subseteq T_q} TU(T_q) - \sum_{Y_{k-1} \subseteq T_q} \sum_{i_t \in X \wedge i_t \in T_q} U(i_t, T_q)$$

$$\geq \sum_{Y_k \subseteq T_p} TU(T_p) - \sum_{Y_k \subseteq T_p} \sum_{i_t \in X \wedge i_t \in T_p} U(i_t, T_p) = CWU(Y_k) \geq \alpha'$$

Như vậy, nếu tập Y_{k-1} là tập ứng viên lợi ích trọng số thấp thì tất cả các tập có tiền tố là Y_{k-1} cũng là tập ứng viên lợi ích trọng số thấp và không thể là tập lợi ích cao. Điều này cho phép loại các tập ứng viên.

Định lý 2 - Cho HCWUs – gồm các tập Y' có CWU(Y') ≥ α', HUs – gồm các tập Y có AU(Y) ≥ α với α là các ngưỡng lợi ích tối thiểu cho trước. Nếu α = α' thì HUs ⊆ HCWUs.

Chứng minh:

Gọi X = ListItemPrefix(Y), $Z_j = T_j \backslash XY$. Khi đó,

$TU(T_j) = \sum_{i_t \in X \wedge i_t \in T_j} U(i_t, T_j) + \sum_{i_k \in Y \wedge i_k \in T_j} U(i_k, T_j) + U(Z_j, T_j)$.

$$\alpha' = \alpha \le AU(Y) = \sum_{Y \subseteq T_q} U(Y, T_q)$$

$$= \sum_{Y \subseteq T_q} TU(T_q) - \sum_{Y \subseteq T_q} \sum_{i_t \in X \wedge i_t \in T_q} U(i_t, T_q) - \sum_{Y \subseteq T_q} \sum_{i_k \in Z_q \wedge i_k \in T_q} U(i_k, T_q) \le \sum_{Y \subseteq T_q} TU(T_q) - \sum_{Y \subseteq T_q} \sum_{i_t \in X \wedge i_t \in T_q} U(i_t, T_q) = CWU(Y)$$

Như vậy, nếu dùng mô hình CWU để loại bỏ các tập ứng viên thì không bỏ sót các tập lợi ích cao.

Để minh chứng mô hình CWU có số ứng viên ít hơn mô hình TWU, chúng tôi đưa ra Định lý sau.

Định lý 3 - Cho tập mục bất kỳ Y, ta luôn có CWU(Y)≤ TWU(Y).

Chứng minh:

Gọi X = ListItemPrefix(Y). Khi đó,

$$CWU(Y) = \sum_{Y \subseteq T_q} TU(T_q) - \sum_{Y \subseteq T_q} \sum_{i_t \in X \wedge i_t \in T_q} U(i_t, T_q) \le \sum_{Y \subseteq T_q} TU(T_q) = TWU(Y)$$

Định lý 4 - Cho HCWUs – gồm các tập Y' có CWU(Y') ≥ α và HTWUs – gồm các tập Y có TWU(Y) ≥ α, với α là các ngưỡng lợi ích tối thiểu cho trước, thì HCWUs ⊆ HTWUs.

Chứng minh:

Nếu X là tập mục bất kỳ thuộc HCWUs thì CWU (X) ≥ α.

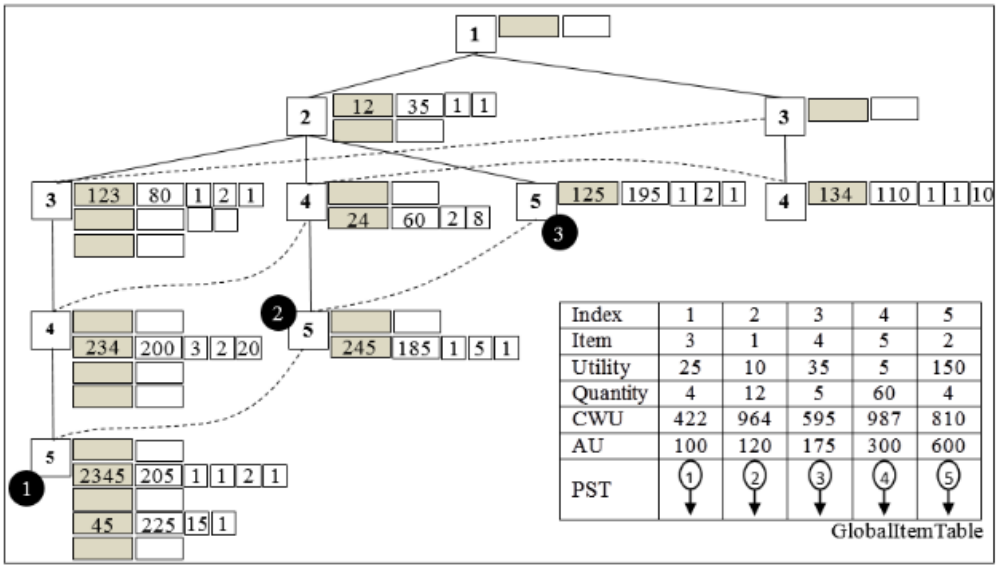
Theo Định lý 3, ta có TWU(X) ≥ CWU(X) ≥ α, do đó X thuộc HTWUs.

Vậy, HCWUs ⊆ HTWUs.

Theo Định lý 3, ta có mô hình CWU sinh ra số lượng tập ứng viên ít hơn so với mô hình TWU.

IV. THUẬT TOÁN CTU-PRO$^+$

Trong phần này chúng tôi giới thiệu thuật toán CTU-PRO$^+$ dựa trên ý tưởng thuật toán CTU-PRO[13] nhưng sử dụng mô hình CWU được chúng tôi giới thiệu ở trên. Giá trị CWU sẽ được tính lại sau khi khai phá tất cả các tập lợi ích cao với tiền tố a bằng cách trừ đi lợi ích của a trên các giao dịch chứa a. Điều này sẽ làm giảm nhanh số phần tử HCWU trong lần khai phá các tập lợi ích cao với tiền tố tiếp theo. Thuật toán này sử dụng một cấu trúc cây có tên gọi là Cây nén mẫu lợi ích – Compressed Utility Patttern Tree (CUP-Tree). Nó là sự mở rộng của cây CFP[14] và biến thể của cây CTU-Tree[11]. Với cây CUP, chúng ta có thể duyệt từ dưới lên để khai phá. Các phần tử có lợi ích thực tế (AU) cao sẽ lấy làm tiền tố để khai phá trước để đảm bảo rằng giá trị CWU sẽ giảm nhanh hơn sau khi trừ đi lợi ích của một phần tử được làm tiền tố. Dưới đây, chúng tôi giới thiệu một số cấu trúc và các bước xây dựng các cấu này.



Hình 1. GlobalCUP-Tree và GlobalItemTable

Một số cấu trúc

- Các phần tử A, B, C, D, E, F trong bảng 1 và bảng 2 được mã tương ứng thành 1, 2, 3, 4, 5, 6.

- Bảng phần tử chung – GlobalItemTable gồm các phần tử ứng viên lợi ích có trọng số cao được sắp xếp tăng dần theo AU. Chú ý, lần đầu tiên xây dựng bảng phần tử chung (GlobalItemTable) thì CWU của các phần tử chính là TWU. Trong bảng này gồm: chỉ số (index), phần tử (item-đã được mã hóa), lợi ích trên một đơn vị phần tử (utility), tổng số lượng của phần tử (quantity), lợi ích ứng viên có trọng số (CWU), lợi ích thực tế của phần tử (AU) và con trỏ trỏ đến gốc của nhánh cây trong CUP-Tree chung.

- Mỗi nút của cây CUP chung – GlobalCUP-Tree bao gồm: chỉ số (index), mảng CWU tương ứng với giá giá trị lợi ích ứng viên có trọng số của 1 tập phần tử, mảng con trỏ chứa số lượng tương ứng của từng phần tử trong giao dịch, con trỏ trỏ đến nút anh em cùng mức, con trỏ trỏ đến nút cha.

- Mảng CWU = $\{T_0, T_1,\ldots T_n\}$, trong đó: T_i là giá trị CWU của tập phần tử từ mức i đến nút chứa T_i. Ví dụ, trong hình 1 tại nút ❷ có CWU[2] = 185, đây là giá trị CWU của tập phần tử xuất phát từ nút hiện tại lên đến nút thứ 2 tương ứng với các chỉ số 5 4 2 (và phần tử thực tế là 2 5 1).

- Tập I = $\{i_1, i_2,\ldots i_n\}$ là tập hợp các phần tử HCWU trong giao dịch được ánh xạ tương ứng với các chỉ số trong GlobalItemTable sau đó chèn các chỉ số index vào cây CUP, bắt đầu từ nút gốc của nhánh cây được trỏ bởi con trỏ PST của phần tử i_1 trong GlobalItemTable.

Các bước xây dựng cây CUP được mô tả dưới đây:

Bước 1: Xây dựng bảng phần tử chung - GlobalItemTable

Đầu vào: cơ sở dữ liệu - D, ngưỡng lợi ích tối thiểu - minUtility

Đầu ra: GlobalItemTable

Procedure ConstructGlobalItemTable

Begin

1. Với mỗi giao dịch t ∈ D

1.1. Với mỗi phần tử i ∈ t

1.1.1. Nếu i ∈ GlobalItemTable

Tăng số lượng và CWU của i

1.1.2. Ngược lại

Chèn i, số lượng và CWU của i vào GlobalItemTable

2. Loại bỏ các item i có CWU(i) < minUtility

3. Sắp xếp các item trong GlobalItemTable theo AU tăng dần

4. Gán chỉ số index cho các item trong GlobalItemTable

End

Bước 2: Xây dựng cây nén mẫu lợi ích chung – GlobalCUP-Tree

Đầu vào: cơ sở dữ liệu, GlobalItemTable

Đầu ra: CUP-Tree

Procedure ConstructGlobalCUPTree

Begin

1. Với mỗi giao dịch t ∈ D

1.1. mappedTrans = {}

1.2. For each item i ∈ t

1.2.1. mappedTrans = mappedTrans ∪ getIndex(i); // Lấy chỉ số index của i trong GlobalItemTable

1.3. Sắp xếp mappedTrans theo thứ tự tăng dần index

1.4. Gọi ***InsertToCUPTree(mappedTrans)***

End

Procedure InsertToCUPTree(mappedTrans)

Begin

1. firstItem = mappedTrans[1]

2. currNode = nút gốc của nhánh cây được trỏ bởi GlobalItemTable[firstItem].PST

3. Với mỗi phần tử tiếp theo i ∈ mappedTrans

3.1. Nếu currNode có con là i thì

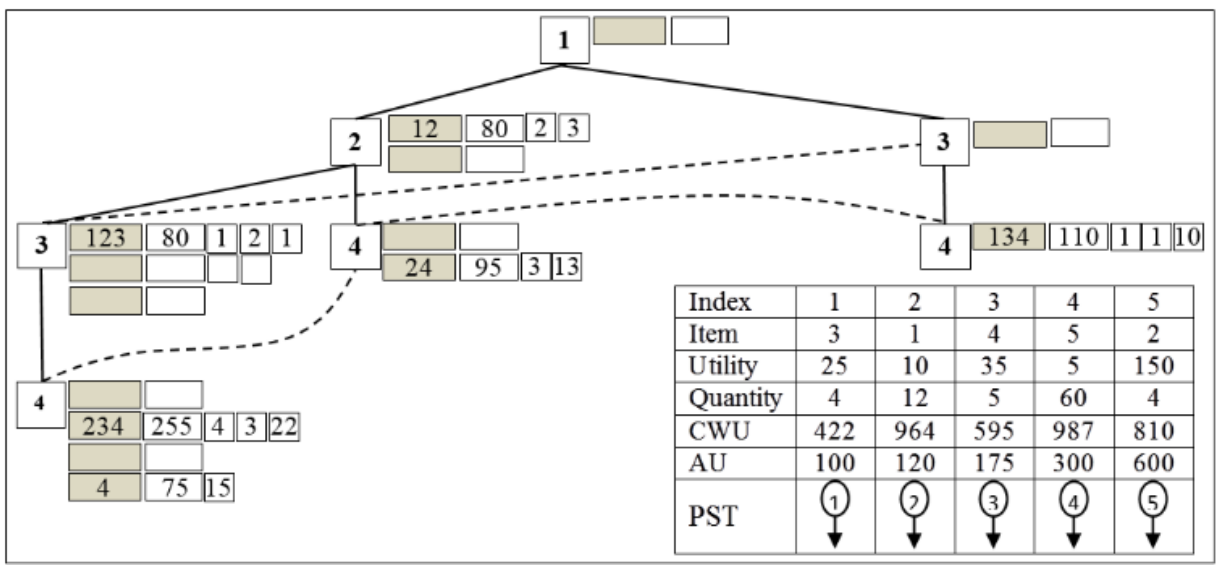
3.1.1. Tăng số lượng, CWU[firstItem-1] của nút con

3.2. Ngược lại

3.2.1. Tạo nút con và gán giá trị cho số lượng và CWU[firstItem-1] của nó

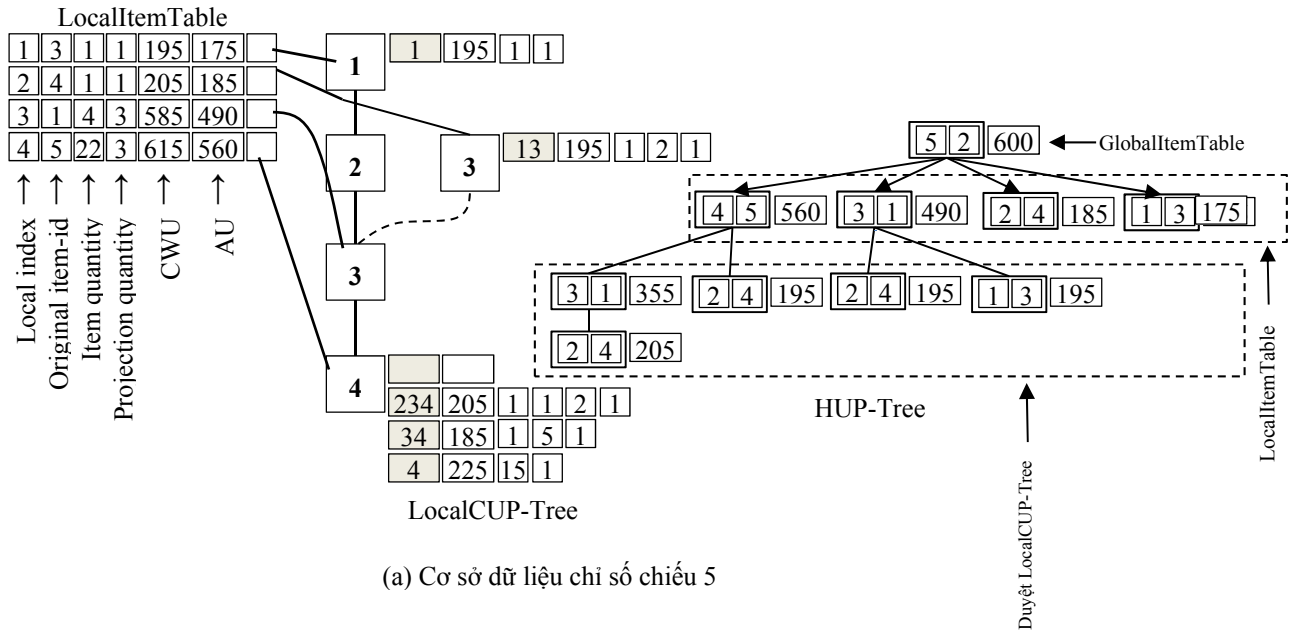
3.2.2. Liên kết với nút anh em cùng mức với nó

End

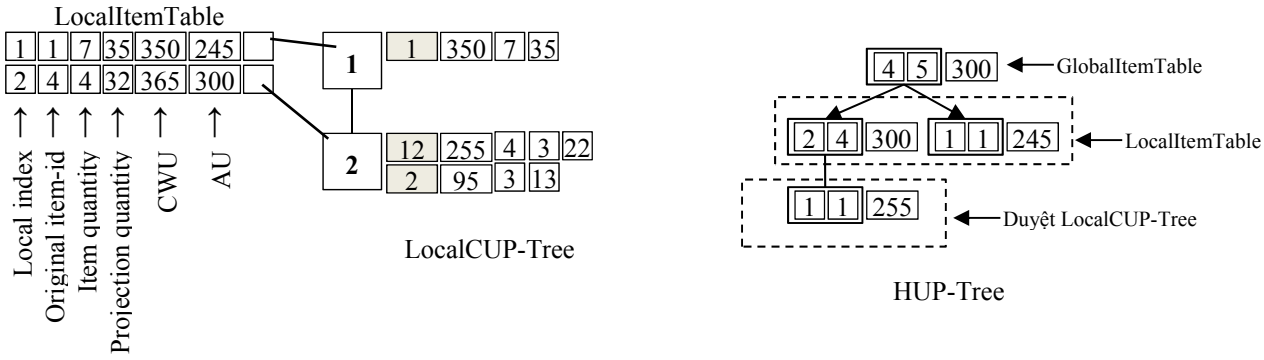


Index	1	2	3	4	5
Item	3	1	4	5	2
Utility	25	10	35	5	150
Quantity	4	12	5	60	4
CWU	422	964	595	987	810
AU	100	120	175	300	600
PST	①	②	③	④	⑤

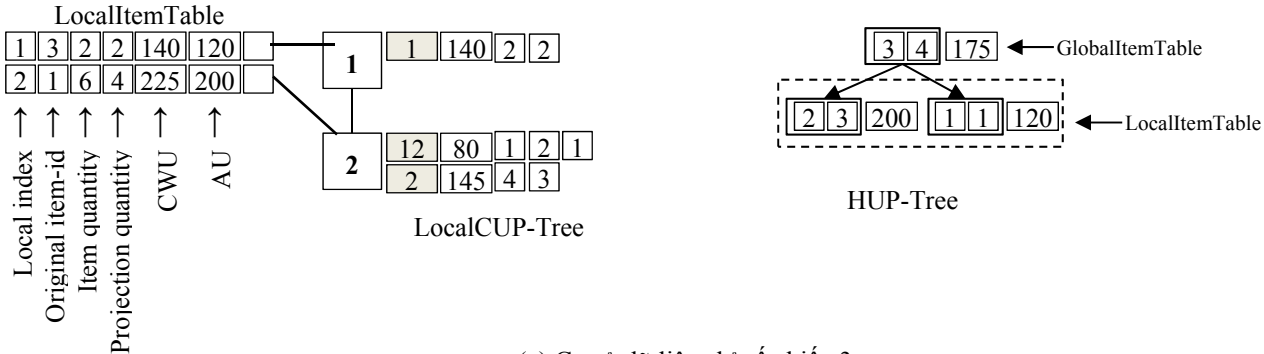
Hình 2. GlobalCUP-Tree và GlobalItemTable sau khi khai phá các tập phần tử có chỉ số 5



(a) Cơ sở dữ liệu chỉ số chiếu 5



(b) Cơ sở dữ liệu chỉ số chiếu 4



(c) Cơ sở dữ liệu chỉ số chiếu 3

Hình 3. Khai phá trên Cơ sở dữ liệu chiếu

Thuật toán CTU-PRO$^+$

Ý tưởng thuật toán: Sau khi xây dựng được GlobalItemTable, GlobalCUP-Tree như hình 1, thuật toán CTU-PRO$^+$ sẽ duyệt cây từ dưới lên trên dựa vào GlobalItemTable để tiến hành khai phá. Các bước cơ bản của thuật toán như sau: (1) duyệt cây từ dưới lên trên với chỉ số j bắt đầu từ phần tử có AU lớn nhất. (2) dựa vào phần tử có chỉ số j, thuật toán sẽ chiếu đến tất cả giao dịch bắt đầu với phần tử chỉ số j. (3) duyệt GlobalCUP-Tree từ nút với phần tử chỉ số j làm tiền tố lên nút gốc để xây dựng LocalItemTable và LocalCUP-Tree cho phần tử chỉ số j như hình 3. (4) xây dựng

HUP-Tree và tính lợi ích thực tế của các tập phần tử. (5) duyệt HUP-Tree để khai phá tập lợi ích cao. (6) cập nhật số lượng, giá trị CWU lên nút cha của nút chỉ số j như hình 2.

Thuật toán CTU-PRO$^+$ gồm các bước chính sau:

1. Với mỗi phần tử HCWU i ∈ GlobalItemTable

1.1. Khởi tạo HUP-Tree //Tạo nút gốc với phần tử i và AU của i trong GlobalItemTable

1.2. Gọi ***ConstructLocalItemTable(x)***

1.3. Với mỗi phần tử HCWU j ∈ LocalItemTable

1.3.1. Gán j là nút con của i trên HUP-Tree

1.4. Gọi ***ConstructLocalCUPTree(x)***

1.5. Gọi ***RecMine(x)***

1.6. Gọi ***Display(x)***

Dưới đây là mô tả chi tiết các thủ tục được gọi trong thuật toán CTU-PRO$^+$:

Xây dựng bảng phần tử riêng – LocalItemTable

Cấu trúc LocalItemTable: chỉ số cục bộ - Local Index, mã phần tử gốc – Item id, số lượng phần tử gốc – Item Quantity, số lượng của phần tử chiếu – Projection Index, CWU

Đầu vào: GlobalCUP-Tree, phần tử chỉ số i ∈ GlobalItemTable

Đầu ra: Bảng các phần tử HCWU xuất hiện cùng phần tử chỉ số tại node i

Procedure ConstructLocalItemTable

Begin

1. Với mỗi nút i trong GlobalCUP-Tree

1.1. Với mỗi phần tử j lên đến gốc

1.1.1. Nếu j ∈ LocalItemTable

1.1.1.1. Tăng số lượng, CWU của j trong LocalItemTable lên

1.1.2. Ngược lại

1.1.2.1. Đưa j và số lượng, CWU tương ứng vào LocalItemTable

End

Xây dựng cây nén mẫu lợi ích riêng – LocalCUP-Tree

Cấu trúc nút trong LocalCUP-Tree: mã nút (node-id), mảng CWU tương ứng với giá trị lợi ích ứng viên có trọng số của 1 tập, mảng con trỏ chứa số lượng tương ứng của từng phần tử trong từng giao dịch (giá trị số lượng cuối cùng là số lượng của phần tử hình chiếu), con trỏ trỏ đến nút anh em cùng mức, con trỏ trỏ đến nút cha.

Đầu vào: GlobalCUP-Tree, phần tử chỉ số i ∈ GlobalItemTable

Đầu ra: LocalCUP-Tree

Procedure ConstructLocalCUPTree

Begin

1. Với mỗi nút i trong GlobalCUP-Tree

1.1. Khởi tạo mappedTrans

1.2. Với mỗi phần tử HCWU j ∈ LocalItemTable từ nút hiện tại lên nút gốc

mappedTrans = mappedTrans ∪ getIndex(j)

1.3. Sắp xếp mappedTrans tăng dần theo mã phần tử

1.4. Gọi ***InsertToCUPTree(mappedTrans)***

End

Khai phá từ cơ sở dữ liệu hình chiếu

Cấu trúc HUP-Tree: mỗi nút gồm: chỉ số chiếu, phần tử gốc, giá trị lợi ích thực tế của tập phần tử. Nút gốc là nút chỉ số chiếu i của một phần tử trong GlobalItemTable và số lượng của nó.

Ý tưởng: sau khi xây dựng xong LocalItemTable cho một phần tử có chỉ số chiếu i sẽ gán các phần tử j ∈ LocalItemTable có HCWU là con của nút gốc chỉ số chiếu i, đồng thời tính luôn giá trị lợi ích thực tế của của tập hai phần tử (phần tử gốc và con của nó). Với các tập lớn hơn hai phần tử thì sẽ duyệt LocalCUP-Tree từ các nút con của chỉ số chiếu i lên đến gốc để tìm các tập mới và tính giá trị CWU cho các tập đó. Sau đó những phần tử nào có CWU lớn hơn minUtility thì sẽ đính kèm với con của nút chỉ số chiếu i.

Đầu vào: LocalItemTable, LocalCUP-Tree, nút x với chỉ số chiếu i.

Đầu ra: HUP-Tree

Procedure RecMine

Begin

1.Gán tất cả CWU trong LocalItemTable bằng 0

2. Với mỗi nút i xuất hiện trong LocalCUP-Tree

2.1. Với mỗi phần tử j từ nút hiện tại lên tới gốc

2.1.1. Tăng giá giá trị CWU của j trong LocalCUP-Tree

2.2. Với mỗi phần tử k ∈ LocalItemTable có HCWU

2.2.1. Tính giá trị lợi ích thực tế và gán kèm k là con của i

End

Cập nhật lại giá trị CWU

Dựa vào khái niệm CWU đã được chứng minh ở trên ta có thể tính toán lại giá trị CWU sau khi khai phá các tập với tiền tố là một phần tử. Ví dụ, như trong hình 1 sau khi tìm tất cả các tập với tiền tố là phần tử chỉ số chiếu 5 (phần tử gốc là 3) thì ta chuyển CWU và số lượng tương ứng của các phần tử còn lại lên trên để tìm tiếp các tập với các tiền tố khác. Kết quả GlobalCUP-Tree sau khi tìm xong các tập lợi ích cao với phần tử chỉ số 5 như hình 2. Ví dụ, tại nút ❷ nếu dùng mô hình TWU thì khi chuyển nút chỉ số 5 lên nút chỉ số 4 thì TWU của tập phần tử có chỉ số 2 4 là 60 + 185 = 245 và số lượng tương ứng là 3 13. Nhưng nếu sử dụng mô hình CWU thì giá trị CWU của tập phần tử có chỉ số 2 4 sẽ là 60 + (185 – 1*150) = 95 và số lượng tương ứng là 3 13. Tương tự, từ nút ❶ chuyển giá trị lên nút cha của nó được 2 tập: 4 và 2 3 4 có CWU là 75 và 255; từ nút ❸ chuyển giá trị lên nút cha của nó được một tập 1 2 với CWU là 80.

Hiển thị các tập HU - DispalyHU

Sau khi xây dựng cây HUP-Tree chứa các giá lợi ích thực tế của các tập. Khi đó sẽ tiến hành duyệt cây và kiểm tra tập nào có giá trị lợi ích lớn hơn giá trị lợi ích tối thiểu minUtility thì hiển thị. Ví dụ, cơ sở dữ liệu như ở bảng 1 với ngưỡng min_Utility = 129.9 ta được các tập lợi ích cao gồm: B: 600, BE: 560, BEA: 355, BEAD: 205, BA: 490, BAC: 195, BAD: 195, BD: 185, BDE: 195, BC: 175, E: 300, ED: 300, EDA: 255, EA: 245, D:175, DA: 200

V. KẾT QUẢ ĐÁNH GIÁ

Trong phần này, chúng tôi so sánh kết quả thực hiện thuật toán CTU-PRO$^+$ sử dụng mô hình CWU của chúng tôi với thuật toán TwoPhase[3], CTU-PRO[13]. Đầu tiên chúng tôi giới thiệu về dữ liệu dùng để thử nghiệm. Tiếp theo là kết quả thực hiện và số lượng các tập ứng viên.

a. Môi trường và dữ liệu

Thuật toán được thực hiện trên máy tính IBM core 2 due 2.4GHz với 2 GB bộ nhớ, chạy trên Windows 7. Chương trình chúng tôi viết bằng Visual C++ 2010. Dữ liệu thử nghiệm gồm: T5N5D100K và T10N5D100K được sinh từ bộ sinh dữ liệu của IBM [15]. Đặc điểm của bộ dữ liệu được mô tả như sau:

Database	T	D	N
T5N5D100K	5	100.000	5
T10N5D100K	10	100.000	5

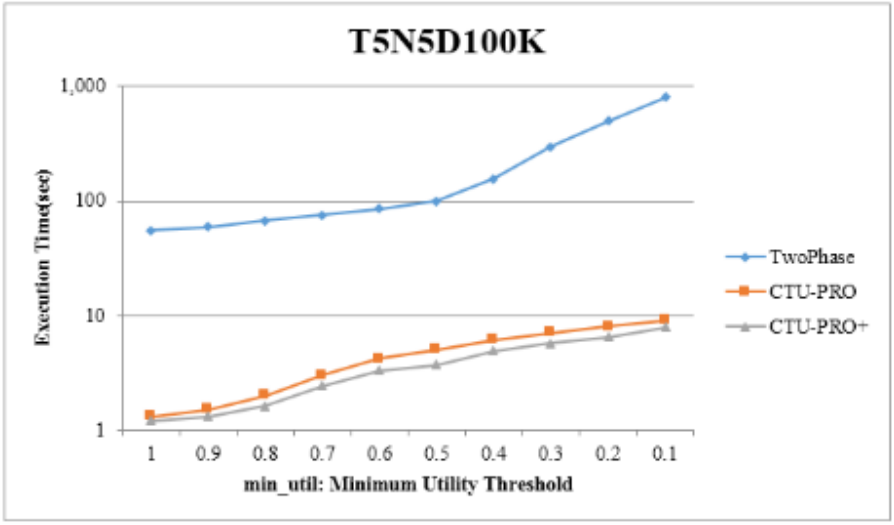
Trong đó: T – là số phần tử trung bình trong một giao dịch; N – là số phần tử khác nhau; D – số giao dịch.

Các bộ dữ liệu này đều chưa có giá trị lợi ích ngoài cho từng phần tử và trong các giao dịch chỉ cho biết phần tử xuất hiện. Do vậy, chúng tôi sinh ngẫu nhiên số lượng cho mỗi phần tử trong mỗi giao dịch với giá trị thuộc từ 1 đến 10 và lợi ích ngoài của mỗi phần tử từ 0.1 đến 10. Hình 4 cho biết việc phân bố lợi ích ngoài của các phần tử.

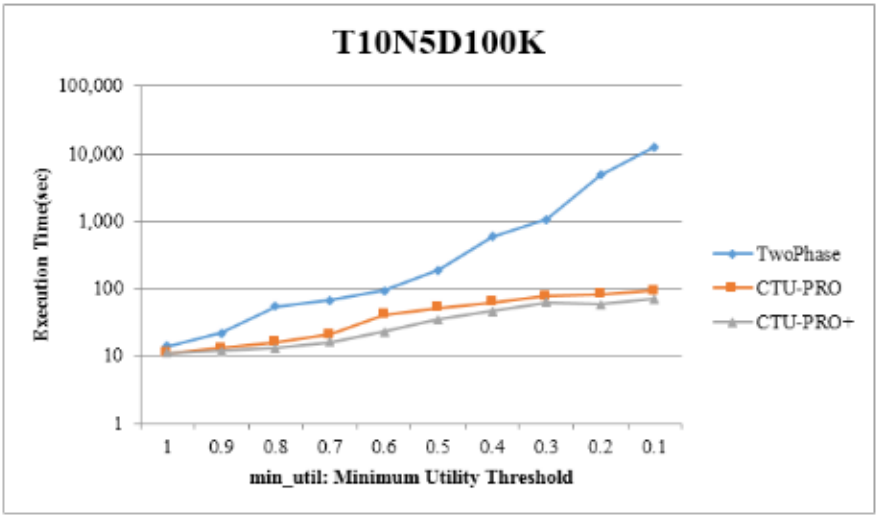
Hình 4. Phân tán giá trị lợi ích

b. Thời gian thực hiện và số ứng viên

Kết quả thử nghiệm, so sánh giữa thuật toán CTU-PRO$^+$ với các thuật toán TwoPhase, CTU-PRO trên bộ dữ liệu T5N5D100K thể hiện trong Hình 5 về so sánh thời gian thực hiện khai phá khi thay đổi ngưỡng lợi ích tối thiểu. Hình 6 so sánh về thời gian thực hiện khai phá khi thay đổi ngưỡng lợi ích tối thiểu trên cơ sở dữ liệu T10N5D100K.



Hình 5. Thời gian thực hiện với các ngưỡng lợi ích tối thiểu khác nhau trên CSDL T5N5D100K



Hình 6. Thời gian thực hiện với các ngưỡng lợi ích tối thiểu khác nhau trên CSDL T10N5D100K

VI. KẾT LUẬN

Trong bài báo này chúng tôi đã tìm hiểu các ưu, nhược điểm của một số thuật toán khai phá tập lợi ích cao được đề xuất trong những năm gần đây, phân tích ưu, nhược điểm của mô hình TWU nhằm loại bớt tập ứng viên. Trên cơ sở đó, chúng tôi đã đề xuất mô hình lợi ích ứng viên có trọng số – CWU (Candidate Weight Utility). Kết quả thử nghiệm cho thấy mô hình CWU sinh ra số lượng ứng viên ít hơn mô hình TWU.

Chúng tôi đã đề xuất thuật toán CTU-PRO$^+$ được cải tiến từ thuật toán CTU-PRO [13] ở một số điểm: thực hiện khai phá theo thứ tự giảm dần lợi ích của phần tử, sử dụng mô hình CWU để loại tập ứng viên, thay đổi cấu trúc của cây để tính toán nhanh các giá trị CWU và AU. Thử nghiệm, so sánh thuật toán đề xuất với hai thuật toán TwoPhase và CTU-PRO trên các bộ dữ liệu đã cho kết quả tốt hơn.

Thời gian tiếp theo, chúng tôi sẽ thử nghiệm mô hình CWU trên một số thuật toán khác nhằm tiếp tục khẳng định ưu điểm của mô hình này.

VII. TÀI LIỆU THAM KHẢO

[1] Yao, H., Hamilton, H. J., Butz, C. J.: A Foundational Approach to Mining Itemset Utilities from Databases. In: Third SIAM Int. Conf. on Data Mining, pp. 482–486 (2004)

[2] Yao, H., Hamilton, H. J.: Mining itemset utilities from transaction databases. Data & Knowledge Engineering 59, 603–626 (2006).

[3] Liu, Y., Liao, W. K., Choudhary, A.: A Two Phase algorithm for fast discovery of High Utility of Itemsets. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS(LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005)

[4] Wei Song, Yu Liu, Jinhong Li, Vertical Mining for High Utility Itemsets, IEEE International Conference on Granular Computing, 2012.

[5] Guo-Cheng Lan · Tzung-Pei Hong · Vincent S. Tseng, An efficient projection-based indexing approach for mining high utility itemsets, Knowl Inf Syst (2014) 38:85–107, DOI 10.1007/s10115-012-0492-y

[6] R. Agrawal, and R. Sri kant, "Fast algorithms for mining association rules in large databases", Proc. 20th IntI. Conf. Very Large Data Bases (VLDB'94), pp. 487-499, September 1994.

[9] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "HUC-Prune: an efficient candidate pruning technique to mine high utility patterns," Appl. Intell., vol. 34, pp.181-198, April 2011.

[10] Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie, Philip S.Yu, "UP-Growth: An Efficient Algorithm for High Utility Itemset Mining", KDD'10, July 25–28, Washington, DC, USA, 2010.

[11] A. Erwin, R. P. Gopalan, and N. R. Achuthan, "CTU-Mine: an efficient high utility item set mining algorithm using the pattern growth approach," Proc. 7th IEEE IntI. Conf. Computer and Information Technology (CIT'07), pp. 71-76, October 2007.

[12] Liu Y, Liao W, Choudhary A "A fast high utility itemsets mining algorithm". In: Proceeding of the utility-based data mining workshop, pp 90–99, 2005.

[13] Alva Erwin, Raj P. Gopalan, N.R. Achuthan, A Bottom-Up Projection Based Algorithm for Mining High Utility Itemsets, Workshop on Integrating AI and Data Mining (AIDM 2007), Gold Coast, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol.84

[14] Sucahyo, Y. G., & Gopalan, R. P. (2004, Nov 1-4). CT-PRO: A Bottom-Up Non Recursive Frequent Itemset Mining Algorithm Using Compressed FP-Tree Data Structure. Paper presented at the IEEE ICDM Workshop on Frequent Itemset Mining Implementation (FIMI), Brighton UK.

[15] IBM Quest Data Mining Project, Quest Synthetic Data Generation Code. Available at (http:// www.almaden.ibm.com/cs/quest/syndata.html)