A FAST ALGORITHM FOR MINING K-ITEM HIGH UTILITY ITEMSETS

Nong Thi Hoa[1*], Nguyen Van Tao[2], Nguyen Thi Tan Tien[3]
[1]Duy Tan University, [2]TNU - University of Information Technology and Communication
[3]TNU - University of Medicine and Pharmacy

ARTICLE INFO

Received: 23/6/2021
Revised: 31/7/2021
Published: 02/8/2021

KEYWORDS

High utility set
High utility set mining
k-item high utility set
Vertical segmentation
Data mining

ABSTRACT

High utility itemset mining performs to find sold item sets whose profit overcomes a given threshold. Finding high utility itemsets helps to suggest related items on e-commerce sites and makes effective policies of sales. Recommender systems usually introduct from 5 to 7 related products to help users choose products. In previous studies, finding high utility itemsets often consume the time because many combinations of items were considered. In this paper, we present a fast algorithm for mining *k*-item high utility itemsets. A compact list structure is used to store information about the itemsets appearing from the curent database. First, the proposed algorithm perform a vertical segmentation of the transaction database to obtain sub-partitions. Next, mine high utility itemsets on each subpartiton by listing k-itemsets and its utility. Experiments were performed on brenchmark databases. Experimental results show that the proposed approach significantly reduces both memory and computation time. Moreover, it is better than the compared algorithm.

MỘT THUẬT TOÁN NHANH CHO KHAI THÁC CÁC TẬP HỮU ÍCH CAO CHỨA K MỤC

Nông Thị Hoa[1*], Nguyễn Văn Tảo[2], Nguyễn Thị Tân Tiến[3]
[1]Trường Đại học Duy Tân, [2]Trường Đại học Công nghệ thông tin và truyền thông - ĐH Thái Nguyên
[3]Trường Đại học Y Dược - ĐH Thái Nguyên

THÔNG TIN BÀI BÁO

Ngày nhận bài: 23/6/2021
Ngày hoàn thiện: 31/7/2021
Ngày đăng: 02/8/2021

TỪ KHÓA

Tập hữu ích cao
Khai thác tập hữu ích cao
Tập hữu ích cao chứa *k* mục
Phân đoạn dọc
Khai phá dữ liệu

TÓM TẮT

Khai thác tập hữu ích cao thực hiện tìm kiếm các tập mục được bán ra mang lại mức lãi cao hơn một ngưỡng cho trước. Việc tìm ra các tập hữu ích cao giúp gợi ý các mặt hàng liên quan trên các trang thương mại điện tử và đưa ra các chính sách bán hàng hiệu quả. Các hệ thống gợi ý thường hiện thêm khoảng 5 đến 7 sản phẩm tương tự hoặc có liên quan để giúp người dùng lựa chọn các sản phẩm cần mua. Trong các nghiên cứu trước đây, việc tìm các tập hữu ích cao thường tốn thời gian do xét nhiều tổ hợp các mục hàng trong một giao dịch. Trong bài báo này, chúng tôi đưa ra một thuật toán nhanh cho khai thác các tập hữu ích cao chứa k mục. Một cấu trúc danh sách nhỏ gọn được dùng để lưu thông tin về các tập mục chứa *k* mục xuất hiện trong cơ sở dữ liệu giao dịch. Đầu tiên, thực hiện phân mảnh dọc cơ sở dữ liệu giao dịch để sinh ra các phân mảnh con. Tiếp theo, khai thác các tập hữu ích cao trên từng phân mảnh dọc bằng cách thống kê các tập mục và tính số tiền lãi của tập mục đó. Các thực nghiệm được làm trên các cơ sở dữ liệu chuẩn. Kết quả thực nghiệm cho thấy cách tiếp cận đề xuất giảm đáng kể cả bộ nhớ và thời gian tính toán. Hơn nữa, thuật toán đề xuất còn tốt hơn thuật toán được so sánh.

DOI: https://doi.org/10.34238/tnu-jst.4691

[*] Corresponding author. *Email: nongthihoa@duytan.edu.vn*

1. Giới thiệu

Ngày nay, nhiều cơ sở dữ liệu khổng lồ từ các tập đoàn kinh doanh, chính phủ và tổ chức đã được xây dựng. Thông tin có giá trị trong các cơ sở dữ liệu này cần khai thác để hỗ trợ cho việc ra quyết định. Các quy tắc kết hợp chứa trong cơ sở dữ liệu giao dịch trình bày mối quan hệ giữa các mục. Các quy tắc này được dùng để lập kế hoạch kinh doanh hoặc phát triển các hệ thống gợi ý. Tập hữu ích cao là các tập mục được bán ra mang lại mức lãi cao hơn một ngưỡng cho trước. Do đó, khai thác tập hữu ích cao tìm ra các tập mục tạo ra nhiều lợi nhuận.

Để giảm cả thời gian và bộ nhớ cho việc tính toán, một số thuật toán nhanh được đề xuất để khám phá các tập hữu ích cao chứa tối đa số lượng mục nhất định. Các phương pháp này sử dụng cấu trúc cây hoặc danh sách để lưu trữ thông tin về cơ sở dữ liệu. Cấu trúc cây có thể được tái cấu trúc hoặc áp dụng các chiến lược cắt nhánh để giảm bớt số tập mục ứng cử [1][2][3]. Cấu trúc danh sách tạo ra các tập mục ứng cử có triển vọng và ước tính lãi của các tập mục mà không cần quét lại cơ sở dữ liệu [4][5][6][7]. Tuy nhiên, các nghiên cứu trước đây vẫn tốn nhiều thời gian và bộ nhớ do chúng đã xử lý tất cả các mục trong mỗi giao dịch. Hơn nữa, những người ra quyết định lập kế hoạch kinh doanh hiệu quả hơn với tập hữu ích cao có chứa số lượng mục thích hợp. Tập hữu ích cao với số ít mục không đủ thông tin và tập hữu ích cao có nhiều mục lại yêu cầu người ra quyết định xem xét mức độ quan trọng của từng mục. Trong bài báo này, chúng tôi đề xuất một danh sách nhỏ gọn và một thuật toán nhanh cho khai thác các tập hữu ích cao chứa *k* mục để đáp ứng nhu cầu của những người ra quyết định và giảm đồng thời cả thời gian và bộ nhớ cho việc tính toán. Danh sách đề xuất lưu trữ các tập mục và tiền lãi của tập mục xuất hiện trong cơ sở dữ liệu. Mỗi dòng gồm các mục, tiền lãi của một tập mục. Danh sách này lưu trữ thông tin trong quá trình duyệt cơ sở dữ liệu và khai thác các tập hữu ích cao chứa *k* mục. Danh sách này được cập nhật hoặc thêm mới sau khi phát hiện ra một tập mục xuất hiện trong các giao dịch liền kề. Chúng tôi trình bày một thuật toán nhanh hiệu quả để khai thác các tập hữu ích cao chứa *k* mục. Đầu tiên, cơ sở dữ liệu hiện tại được phân đoạn theo chiều dọc tạo thành các phân đoạn con. Mọi dòng trong mỗi phân đoạn đều chứa *k* mục. Đối với mỗi phân đoạn con, các tập mục xuất hiện trong các giao dịch được khai thác và lưu trữ trong một danh sách chung. Từ danh sách chung này, xuất ra các tập hữu ích cao chứa k mục dựa vào tiền lãi của tập mục có lớn hơn ngưỡng lãi cho trước. Cách tiếp cận của chúng tôi có được hai ưu điểm mạnh bao gồm không tạo các tập mục ứng cử, không duyệt lại cơ sở dữ liệu để tìm tập hữu ích cao khi thay đổi ngưỡng lãi. Các thử nghiệm được tiến hành trên cơ sở dữ liệu chuẩn với đủ sự đa dạng về số lượng mẫu, tổng số mục và số mục dài nhất trong một giao dịch. Chúng tôi đánh giá hiệu suất của thuật toán đề xuất về cả thời gian và bộ nhớ. Kết quả cho thấy, cách tiếp cận của chúng tôi hiệu quả trong khai thác các cơ sở dữ liệu thực và tốt hơn thuật toán được so sánh.

Phần còn lại của bài báo được tổ chức như sau: trong phần II, chúng tôi trình bày thuật toán đề xuất cho khai thác nhanh các tập hữu ích cao chứa k mục. Phần III trình bày kết quả thực nghiệm trên các tập dữ liệu chuẩn. Cuối cùng, một số kết luận và hướng phát triển được nêu ra trong phần IV.

2. Thuật toán đề xuất cho khai thác các tập hữu ích cao chứa k mục

Chúng tôi nhận thấy các cơ sở dữ liệu giao dịch chứa phần lớn các tập hữu ích cao chứa một số ít các mục và chứa rất ít các tập hữu ích cao chứa nhiều mục. Hơn nữa, việc lập kế hoạch bán hàng hay gợi ý các mặt hàng liên quan thực hiện hiệu quả hơn dựa trên tập hữu ích cao chứa một số ít các mục. Do đó, chúng tôi đề xuất một thuật toán nhanh hiệu quả để khai thác các tập hữu ích cao chứa *k* mục.

2.1. Nguyên tắc tìm các tập hữu ích cao chứa k mục

Phát biểu bài toán: Cho một cơ sở dữ liệu giao dịch, số mục phải có trong mỗi tập hữu ích cao và một ngưỡng lãi, tìm các tập mục xuất hiện trong cơ sở dữ liệu mà có mức lãi lớn hơn hoặc

bằng ngưỡng lãi. Một cơ sở dữ liệu giao dịch gồm số lượng từng mục hàng đã mua trong mỗi giao dịch và một bảng kê lãi của từng mặt hàng.

Đầu tiên, thực hiện phân đoạn cơ sở dữ liệu hiện tại theo chiều dọc để tạo thành các phân đoạn con. Mỗi dòng trong các phân đoạn đều chứa k mục. Hình 1 mô tả việc phân đoạn dọc một cơ sở dữ liệu với *k*=5. Tiếp theo, khai thác các tập hữu ích cao chứa *k* mục từ mỗi phân đoạn con.

Để tiết kiệm bộ nhớ, một danh sách với cấu trúc đơn giản được dùng để lưu trữ thông tin về các tập mục trong quá trình duyệt cơ sở dữ liệu. Mỗi dòng chứa thông tin về tập mục chứa k mục xuất hiện trong các phân đoạn con. Cấu trúc của danh sách có hai phần gồm các mục và tiền lãi của tập mục. So sánh tiền lãi với ngưỡng lãi để đưa ra các tập hữu ích cao. Đối với các giao dịch có số mục không chia hết cho k và các mục được đánh số từ 1, 2, ..., *n*, thực hiện thêm mục 0 vào cuối để đạt đủ k mục.

Giao dịch	Các mặt hàng đã mua trong từng giao dịch														
	Phân đoạn 1					Phân đoạn 2					Phân đoạn 3				
505	1	2	3	4	5	6	7	8	9	10	11	12	13	0	0
506	3	4	5	6	7	8	9	10	11	12	13	14	0	0	0
507	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
508	1	2	3	4	5	6	7	8	9	10	0	0	0	0	0

Hình 1. *Phân đoạn dọc một cơ sở dữ liệu với k=5*

Danh sách này được cập nhật hay thêm mới trong quá trình duyệt cơ sở dữ liệu. Xét từng phân đoạn con, duyệt các giao dịch từ trên xuống. Nếu một tập mục xuất hiện trong các giao dịch liền kề thì thực hiện tính lãi của tập mục trong các giao dịch liền kề vừa xét. Kiểm tra tập mục hiện tại trong danh sách. Nếu tập mục chưa có thì thêm một dòng mới vào danh sách. Nếu tập mục đã tồn tại thì cập nhật tiền lãi của tập mục bằng tiền lãi cũ cộng với tiền lãi trong các giao dịch liền kề vừa xét.

Sau khi duyệt hết cơ sở dữ liệu, đưa ra các tập hữu ích cao chứa k mục từ danh sách trên.

2.2. Thuật toán khai thác các tập hữu ích cao chứa k mục

Các thông tin vào, thông tin ra và các bước của thuật toán được mô tả chi tiết như sau:

Input:

- DB là cơ sở dữ liệu giao dịch
- *min_util* là ngưỡng lãi
- *k* là số mục có trong tập hữu ích cao

Output: Các tập hữu ích cao chứa *k* mục

Danh sách các biến

- **L** là danh sách lưu các tập mục xuất hiện trong cơ sở dữ liệu
- **P** là một phân đoạn con của cơ sở dữ liệu
- **T** là giao dịch
- **I** tập mục đang xét
- **e** là tập mục của giao dịch đang xét

Nội dung của thuật toán được mô tả chi tiết như sau:

Bước 1: Phân đoạn DB theo chiều dọc để mọi dòng của mỗi P đều chứa *k* mục. Thực hiện thêm vào mục 0 với các dòng chưa có đủ *k* mục.

Bước 2: Thực hiện duyệt các giao dịch từ trên xuống dưới cho từng phân đoạn **P**.

Bước 2.1: Đặt **I** là tập mục trong giao dịch đầu tiên của **P** đang xét.

Bước 2.2: Với từng giao dịch **T** trong P, thực hiện kiểm tra:

Nếu **e** = **I** thì cộng dồn lãi cũ của **I** với lãi của **I** trong **T**.

Nếu **e** ≠ **I** thì duyệt **L** để kiểm tra **I** đã có trong **L** hay chưa. Nếu **I** *có trong* **L** thì cập nhật tiền lãi của **I** trong **L.** Nếu **I** *chưa có trong* **L** thì thêm một dòng mới vào **L** để lưu các thông tin về **I**.

Bước 3: Đưa ra các tập mục trong **L** có tiền lãi lơn hơn ngưỡng *min_util.*

Các ưu điểm của thuật toán đề xuất gồm:

- Không tạo ra các tập mục ứng cử.
- Khi thay đổi *min_util* thì vẫn khai thác được các tập hữu ích cao mà không cần quét lại cơ sở dữ liệu.
- Chỉ duyệt cơ sở dữ liệu trong một lần.
- Cung cấp một danh sách nhỏ gọn để tổng hợp tất cả thông tin trong quá trình khai thác tập hữu ích cao.
- Dễ hiểu và dễ thực hiện.

3. Kết quả thực nghiệm

Các thực nghiệm được làm trên các cơ sở dữ liệu chuẩn và có cùng số mục trong mỗi giao dịch (cơ sở dữ liệu dày đặc). Ba cơ sở dữ liệu dày đặc được sử dụng gồm *mushroom* (8124 giao dịch, có 119 mục và có 23 mục trong mỗi giao dịch), *chess* (3196 giao dịch, có 75 mục và có 39 mục trong mỗi giao dịch), *connect* (67557 giao dịch, có 129 mục và có 43 mục trong mỗi giao dịch). Chúng được tải xuống từ FIMI Repository [8]. Hầu hết các cơ sở dữ liệu không cung cấp danh sách lãi của từng mục và số lượng của từng mục trong mỗi giao dịch. Giống như một số nghiên cứu trước đây [4], tiền lãi của mỗi mục được tạo từ 0,01 đến 10 bằng cách sử dụng phân phối chuẩn log và số lượng của từng mục trong mỗi giao dịch được tạo ngẫu nhiên trong khoảng từ 1 đến 10.

Chúng tôi đã tiến hành thử nghiệm trên một máy tính được trang bị Bộ xử lý Intel 64 bit, Core i3 3,8 GHz, bộ nhớ chính 32 GB và chạy Windows 10. Chúng tôi làm các thực nghiệm với ngưỡng lãi thay đổi ứng với ba giá trị của k là 5, 6, 7. Chúng tôi chọn các ngưỡng lãi nhỏ giúp sinh ra nhiều tập hữu ích cao hơn. Vì vậy, thời gian chạy và bộ nhớ sẽ lớn hơn. Điều này giúp đánh giá hiệu quả hiệu suất của thuật toán đề xuất.

Chúng tôi chọn một phần của *mushroom* (từ giao dịch 2000 đến giao dịch 8124), một phần của *chess* (từ giao dịch 1000 đến giao dịch 3196) và một phần của *connect* (từ giao dịch 60000 đến giao dịch 67557) để giới hạn thời gian chạy. Các giao dịch phía cuối được chọn vì các giao dịch sau mang tính thời sự hơn các giao dịch trước.

3.1. Phân tích kết quả thực nghiệm theo từng tập dữ liệu

Bảng 1 thể hiện các số liệu thu được khi khai thác tập hữu ích cao từ *mushroom*. Bảng 1 cho thấy *k*=7 là giá trị tốt nhất cho khai thác tập hữu ích cao từ *mushroom* do thời gian tốn ít nhất, số lượng tập hữu ích cao thu được và dung lượng bộ nhớ sử dụng cũng tương tự với *k*=6.

Dữ liệu thu được khi khai thác tập hữu ích cao từ *chess* được trình bày trong Bảng 2. Chúng tôi nhận thấy *k*=7 là giá trị thích hợp nhất cho khai thác tập hữu ích cao từ *chess*. So với kết quả của *k*=6 thì thời gian giảm nhanh khi giảm ngưỡng lãi, bộ nhớ sử dụng tăng lên ít (khoảng 0,3 MB) và số lượng tập hữu ích cao cao hơn (nhiều hơn khoảng 1380 tập hữu ích cao).

Bảng 1. *Kết quả thu được từ mushroom*

Ngưỡng lãi	Thời gian (ms)			Bộ nhớ (MB)			Số tập hữu ích cao		
	k=5	k=6	k=7	k=5	k=6	k=7	k=5	k=6	k=7
500	237,2	156,2	140,2	0,353	0,353	0,353	92	591	106
400	200,0	140,4	143,6	0,353	0,353	0,353	226	820	266
300	200,2	137,6	122,1	0,353	0,353	0,353	598	1265	643
200	143,6	119,0	93,8	0,353	0,353	0,353	1722	2512	2104
100	97,0	59,2	52,8	0,353	0,353	0,353	6561	6747	6224

Bảng 2. *Kết quả thu được từ chess*

Ngưỡng lãi	Thời gian (ms)			Bộ nhớ (MB)			Số tập hữu ích cao		
	k=5	k=6	k=7	k=5	k=6	k=7	k=5	k=6	k=7
500	631,4	927,8	968,2	0,820	0,353	0,353	708	993	1023
400	659,6	893,8	1024,8	1,053	0,353	0,353	929	1439	1552
300	668,6	806,2	953,2	1,053	0,353	0,353	1364	2266	2522
200	524,8	797,0	775,0	0,596	0,353	0,353	2155	3926	4485
100	387,6	531,4	484,4	0,879	0,353	0,681	4239	7989	9326

Dữ liệu trong Bảng 3 thể hiện kết quả thu được khi khai thác tập hữu ích cao từ *connect*. Đây là cơ sở dữ liệu lớn, mới và phức tạp. Với *k*=7, thời gian xử lý nhiều nhất nhưng lại giảm nhanh hơn so với *k*=6 khi giảm ngưỡng lãi. Hơn nữa, bộ nhớ sử dụng cũng giảm nhanh khi giảm ngưỡng lãi trong khi *k*=6 lại tăng bộ nhớ sử dụng. Ngoài ra, số tập hữu ích cao tìm được ở mức ngưỡng thấp lại nhiều hơn *k*=6 khoảng hơn 1000 tập hữu ích cao.

Bảng 3. *Kết quả thu được từ connect*

Ngưỡng lãi	Thời gian (ms)			Bộ nhớ (MB)			Số tập hữu ích cao		
	k=5	k=6	k=7	k=5	k=6	k=7	k=5	k=6	k=7
500	762,4	896,8	2074,4	0,354	0,354	28,005	3666	4618	4554
400	784,4	915,6	1837,0	0,354	0,354	28,005	4612	5805	5844
300	656,0	893,2	1540,4	0,354	0,354	28,005	6000	7579	7703
200	577,8	840,6	1212,0	0,354	11,337	11,414	8260	10428	10653
100	797,0	634,8	853,0	5,845	11,477	5,954	12318	15468	16504

Qua các phân tích kết quả thực nghiệm trên, chọn *k*=7 là giá trị thích hợp nhất cho cả ba cơ sở dữ liệu trên. Với ngưỡng lãi thấp nhất (100) khi khai thác từ *connect*, thời gian tính toán là 853 (ms) và bộ nhớ đã dùng là gần 6 MB. Vì vậy, thuật toán đề xuất có thể khai thác hiệu quả trên các cơ sở dữ liệu lớn.

3.2. So sánh với nghiên cứu liên quan

Chúng tôi so sánh thuật toán của chúng tôi với EFIM-Closed [1]. EFIM-Closed thực hiện tìm các tập hữu ích đóng có thời gian thực hiện nhanh nhất. Kết quả của EFIM-Closed được lấy từ chương trình nằm trong thư viện SPMF **Error! Reference source not found.**. Các thực nghiệm được làm trên tập *mushroom* để hạn chế thời gian tính toán. Do ngưỡng lãi càng thấp thì thời gian chạy càng nhiều. Chúng tôi chọn ngưỡng lãi của EFIM-Closed cao hơn 100 lần ngưỡng lãi dùng trong thuật toán đề xuất để so sánh. Nghĩa là, thuật toán đề xuất chạy mất nhiều thời gian hơn nhiều lần so với EFIM-Closed. Chúng tôi chọn *k*=7 để chạy với *mushroom*.

Bảng 4 thể hiện thời gian xử lý và bộ nhớ của thuật toán đề xuất đã dùng cho *mushroom* với k=7. Các mức ngưỡng lãi được chọn là 50, 100, 150, 200, 250. Số tập hữu ích cao chứa 7 mục khai thác được từ 106 tập đến 6224 tập. Thời gian xử lý từ 52,8 (ms) đến 143,6 (ms) và bộ nhớ ổn định ở mức 0,35 MB.

Bảng 5 thể hiện thời gian xử lý và bộ nhớ của thuật toán EFIM-Closed đã dùng cho *mushroom*. Các mức ngưỡng lãi được chọn là 5000, 10000, 15000, 20000, 25000. Số tập hữu ích cao đóng khai thác được từ 21772 tập đến 75248 tập. Thời gian xử lý từ 5277,3 (ms) đến 8628,0 (ms). Bộ nhớ thay đổi từ 181,27 MB đến 242,21 MB.

Bảng 4. *Thời gian xử lý và bộ nhớ của thuật toán đề xuất đã dùng cho mushroom với k=7*

Ngưỡng lãi	Số tập hữu ích cao chứa 7 mục	Thời gian (ms)	Bộ nhớ (MB)
250	106	**140,2**	0,35309
200	266	143,6	**0,35323**
150	643	122,1	0,35301
100	2104	93,8	0,35291
50	6224	52,8	0,35330

Bảng 5. *Thời gian xử lý và bộ nhớ của thuật toán EFIM-Closed đã dùng cho mushroom với k=7*

Ngưỡng lãi	Số tập hữu ích cao đóng	Thời gian (ms)	Bộ nhớ (MB)
25000	21772	**5277,3**	**181,272**
20000	27591	5816,3	212,261
15000	35186	6516,7	242,218
10000	48855	7378,3	194,346
5000	75248	8628,0	216,416

Dữ liệu từ Bảng 4 và 5 cho thấy, thời gian tính toán của EFIM-Closed cao hơn **37** lần so với thuật toán đề xuất và dung lượng bộ nhớ của EFIM-Closed cao hơn **513** lần so với thuật toán đề xuất. Nghĩa là, sự thực hiện của thuật toán đề xuất là tốt hơn thuật toán EFIM-Closed cả về bộ nhớ và thời gian tính toán.

4. Kết luận

Trong bài báo này, chúng tôi đề xuất một danh sách nhỏ gọn và một thuật toán nhanh mới để khai thác tập hữu ích cao chứa k mục. Một danh sách duy nhất được dùng để lưu trữ thông tin của các tập mục xuất hiện trong quá trình duyệt cơ sở dữ liệu. Các tập hữu ích cao chứa k mục được trích xuất từ danh sách dựa vào sự so sánh tiền lãi với ngưỡng lãi cho trước. Đầu tiên, một phân đoạn dọc được thực hiện trên cơ sở dữ liệu để tạo thành các phân đoạn con. Tiếp theo, các tập hữu ích cao chứa k mục được khai thác từ mỗi phân đoạn con. Cách tiếp cận đề xuất có được các ưu điểm mạnh gồm không tạo các tập mục ứng cử, không cần duyệt lại cơ sở dữ liệu khi thay đổi ngưỡng lãi. Các thử nghiệm được thực hiện trên ba cơ sở dữ liệu dày đặc chuẩn. Thuật toán đề xuất tiêu tốn một khoảng thời gian nhỏ (tối đa khoảng 2s) và chiếm bộ nhớ rất nhỏ (tối đa khoảng 28 MB). Kết quả thực nghiệm cho thấy cách tiếp cận của chúng tôi phù hợp với các ứng dụng thực và tốt hơn thuật toán được so sánh.

Chúng tôi sẽ tiếp tục nghiên cứu để giảm dung lượng bộ nhớ và tiến hành thử nghiệm trên các cơ sở dữ liệu lớn trong tương lai.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] P. Fournier-Viger, S. Zida, J. C.-W. Lin, C.-W. Wu, V. S. Tseng, "EFIM-Closed: Fast and memory efficient discovery of closed high-utility itemsets," In: P*roceedings of MLDM 2016, LNCS*, Springer, USA, 2016, vol. 9729, pp. 199-213.

[2] Y. Unil, K. Donggyu, Y. Eunchul, F. Hamido, "Damped window based high average utility pattern mining over data streams," *Knowledge-Based Systems*, vol. 144, pp. 188-205, 2018.

[3] K. Donggyu and Y. Unil, "Efficient algorithm for mining high average-utility itemsets in in-cremental transaction databases," *Applied Intelligence*, vol. 47, no. 1, pp. 114-131, 2017.

[4] C. W. Wu, P. Fournier-Viger, J. Y. Gu, V. S. Tseng, "Mining closed high utility itemsets without candidate generation," In: *Proceeding of Technologies and Applications of Artificial Intelligence (TAAI)*, Taiwan, 2015, pp. 187-194.

[5] P. Fournier-Viger, J. Lin, R. Nkambou, B. Vo, V. S. Tseng, "Mining Compact High Utility Itemsets Without Candidate Generation," *High-Utility Pattern Mining: Theory, Algorithms and Applications*, vol. 51, pp. 282-307, 2019.

[6] L. T. Dam, R. Heri, N. Kjetil, and H. Q. Duong, "Towards efficiently mining closed high utility itemsets from incremental databases," *Knowledge-Based Systems*, vol. 165, pp. 13-29, 2019.

[7] Y. Until, K. Donggyu, "Mining of high average-utility itemsets using novel list structure and pruning strategy," *Future Generation Computer Systems*, vol. 68, pp. 346-360, 2017.

[8] Frequent Itemset Mining Dataset Repository, 2012. [Online]. Available: http://fimi.uantwerpen.be/. [Accessed May 25, 2021].

[9] P. Fournier-Viger, A. Gomariz, A. Soltani, H. Lam, and T. Gueniche, "Spmf: Open-source data mining platform," 2014. [Online]. Available: http://www.philippe-fournier-viger.com/spmf/. [Accessed May 15, 2021].