BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

VÕ TẤN ANH KIỆT

KHAI THÁC TẬP MỤC LỢI ÍCH CAO

LUẬN VĂN THẠC SĨ

Chuyên ngành: Công Nghệ Thông Tin

Mã ngành: 60480201

TP. HỒ CHÍ MINH, tháng 10 năm 2015

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

VÕ TẤN ANH KIỆT

KHAI THÁC TẬP MỤC LỢI ÍCH CAO

LUẬN VĂN THẠC SĨ

Chuyên ngành: Công Nghệ Thông Tin

Mã ngành: 60340102

Cán bộ hướng dẫn khoa học: PGS. TS LÊ HOÀI BẮC

TP. HỒ CHÍ MINH, tháng 10 năm 2015

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HỒ CHÍ MINH

Cán bộ hướng dẫn khoa học:

PGS. TS LÊ HOÀI BẮC

Luận văn Thạc sĩ được bảo vệ tại Trường Đại học Công nghệ TP. HCM

ngày 17 tháng 10 năm 2015.

Thành phần Hội đồng đánh giá Luận văn Thạc sĩ gồm:

TT	Họ và Tên	Chức danh Hội đồng
1	PGS. TSKH. Nguyễn Xuân Huy	Chủ tịch
2	PGS. TS. Quản Thành Thơ	Phản biệ n 1
3	TS. Nguyễn Thị Thúy Loan	Phản biệ n 2
4	TS. Võ Đình Bảy	Ủy viên
5	TS. Cao Tùng Anh	Ủy viên, Thư ký

Xác nhận của Chủ tịch Hội đồng đánh giá luận văn sau khi luận văn đã sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM
PHÒNG QLKH – ĐTSĐH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

TP. HCM, ngày 03 *tháng 04 năm 2015*

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên : Võ Tấn Anh Kiệ t Giới tính: Nam.

Ngày, tháng, năm sinh : 12 – 06 – 1976 Nơi sinh: TP. Hồ Chí Minh.

Chuyên ngành : Công Nghệ Thông Tin MSHV : 1341860042

I- Tên đề tài:

KHAI THÁC TẬP MỤC LỢI ÍCH CAO

II- Nhiệm vụ và nội dung:

- Nghiên cứu về khám phá tri thức và khai thác dữ liệu cho Cơ Sở Dữ Liệu lớn có lợi ích đi kèm.
- Nghiên cứu và triển khai các thuật toán khai thác itemset lợi ích.
- Lập trình kiểm thử và so sánh hai thuật toán HUI-Miner và FHM.

III- Ngày giao nhiệm vụ: 03/04/2015

IV- Ngày hoàn thành nhiệm vụ: 07/09/2015

V- Cán bộ hướng dẫn: Phó Giáo Sư . Tiến Sĩ. Lê Hoài Bắc

CÁN BỘ HƯỚNG DẪN **KHOA QUẢN LÝ CHUYÊN NGÀNH**

PGS. TS LÊ HOÀI BẮC

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi.Các số liệu, kết quả đánh giá, nhận xét và các đề xuất cải tiến mới nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện luận văn này cũng như các trích dẫn hay tài liệu học thuật tham khảo đã được cảm ơn đến tác giả hay ghi rõ ràng nguồn gốc thông tin trích dẫn trong Luận văn.

Học viên thực hiện Luận văn

Võ Tấn Anh Kiệt

LỜI CÁM ƠN

Trước hết, cho tôi được gửi lời cảm ơn đến sự hướng dẫn và giúp đỡ tận tình của PGS.TS Lê Hoài Bắc.

Xin cảm ơn các Thầy/Cô Khoa Công Nghệ Thông Tin Đại Học Công Nghệ TP. HCM đã sát cánh và cung cấp cho tôi những kiến thức quí báu trong suốt thời gian học tập và nghiên cứu thực hiện luận văn.

Tôi cũng xin gởi lời cảm ơn đến gia đình, bạn bè và những người thân đã luôn quan tâm và giúp đỡ tôi trong suốt thời gian học tập và nghiên cứu hoàn thành luận văn này.

Luận văn không thể tránh khỏi những sai sót, rất mong nhận được ý kiến đóng góp của mọi người cho luận văn được hoàn thiện hơn.

Tôi xin chân thành cảm ơn.

TP. Hồ Chí Minh, tháng 10 năm 2015

Võ Tấn Anh Kiệt

TÓM TẮT

Khai thác tập có ích cao là một nhiệm vụ mang tính thử thách trong khai thác mẫu tuần tự, lĩnh vực có nhiều ứng dụng rộng rãi. Thuật toán điển hình là HUI-Miner[7]. Thuật toán này sử dụng phương pháp tìm kiếm theo chiều sâu để tìm ra các mẫu và tính toán lợi ích của chúng mà không tốn chi phí cho việc duyệt CSDL. Dù hướng tiếp cận này có hiệu quả, việc khai thác các tập có ích cao vẫn còn tốn kém vì HUI-Miner[7] phải thực hiện thao tác kết các item được tạo ra bằng thủ tục tìm kiếm .Trong luận văn này, tôi tập trung nghiên cứu một thuật toán khai thác các tập lợi ích cao với chiến lược cắt giảm không gian tìm kiếm có hiệu quả mà không phải thực hiện phép kết có tên là FHM[13]. Thuật toán này dễ triển khai và có hiệu quả hơn thuật toán trước đó là HUI-Miner[7]. Ba thuật toán có liên quan là Two-phase[8], TWU-Mining[12] và HUI-Miner[7] cũng được tìm hiểu.

ABTRACT

High utility itemset mining is a challenging task in frequent pattern mining, which has wide applications. The state-of-the-art algorithm is HUI-Miner[7]. It adopts a vertical representation and performs a depth first search to discover patterns and calculate their utility without performing costly database scans. Although, this approach is efective, mining high-utility itemsets remains computationally expensive because HUI-Miner[7] has to perform a costly join operation for each pattern that is generated by its search procedure. In this thesis, I address the algorithm of HUIM that named FHM[13] with the effective prunning stategy based on the analysis of item co-occurrences to reduce the number of join operations.

FHM[13] is easy to deploy and more efective than HUI-Miner[7]. Three related algorithms: Two- phase[8], TWU-Mining[12] và HUI-Miner[7] are also discovered.

Mục Lục

DANH MỤC CÁC TỪ VIẾT TẮT

Ký hiệu, viết tắt	Ý nghĩa tiếng Việt	Ý nghĩa tiếng anh
CSDL	Cơ sở dữ liệu	Data Base (DB)
EUCP	Phương pháp ước lượng giá trị lợi ích đồng thời	Estimated Utility Cooccurrence Pruning
EUCS	Cấu trúc ước lượng giá trị lợi ích đồng thời	Estimated Utility Co-occurrence Structure
FHM	Tên thuậ t toán khai thác tậ p mục lợi ích cao sử dụng phương pháp cắt tỉa đồng thời	Faster High-Utility Itemset Mining us Estimated Utility Co-occurrence Pruning
HUI	Tập mục lợi ích cao	High utility itemset
HUIM	Khai thác tập mục lợi ích cao	High utility itemset mining
ITEMSET	Tập mục	Itemset
ITEM	Mục	Item
KDD	Kỹ thuật khám phá tri thức và khai thác dữ liệu	Knowledge Discovery and Data Mining
KTDL	Khai thác dữ liệu	Data Mining
MIUT	Độ lợi ích item tối thiểu	Minimum item utility
MINULTI	Giá trị ngưỡng	Min utility
TID	Giao tác	Transaction Item Database
TU	Độ lợi ích của giao tác	Transaction Utility
TWDCP	Trọng số giao dịch đóng giảm	Transaction-weighted Downward Closure Property
TWU	Trọng số độ lợi ích của giao tác	Transaction – Weighted Utilization
UL	Danh sách giá trị lợi ích	Utility-list
UP – Growth	Thuật toán UP – Growth	Utility Pattern Growth
UP – Tree	Cây Up – tree	Utility Pattern Tree
WIT – Tree	Cây WIT – Tree	Weighted Itemset – Tidset Tree
TWD	Giao dịch có trọng số giảm	Transaction-Weighted-

		Downward
TWU – Mining	Thuật toán TWU – Mining	Transaction Weighted Utility Mining

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

CHƯƠNG 1
GIỚI THIỆU TỔNG QUAN

1.1 GIỚI THIỆU ĐỀ TÀI

Chúng ta đang sống trong kỷ nguyên của công nghệ thông tin. Ngoài sự phát triển của Internet thì sự phát triển nhanh chóng của các kỹ thuật tiên tiến về lưu trữ dữ liệu lớn cũng như khối dữ liệu khổng lồ phát sinh từ các doanh nghiệp, chính phủ và các tổ chức khoa học. Vấn đề là làm sao chúng ta có thể khai thác được các thông tin có giá trị từ nguồn dữ liệu đa dạng đó thành thông tin có ích. Do đó, việc khai thác dữ liệu (data mining) là quá trình giúp chúng ta có được những *tri thức* từ kho dữ liệu phát sinh hàng giờ.

Khai thác tập phổ biến (FIM – Frequent Itemset Mining) là công việc phổ biến trong khai thác dữ liệu, rất cần thiết trong nhiều ứng dụng. Cho 1 CSDL giao tác, FIM khám phá tập phổ biến, tức là nhóm các item phổ biến xuất hiện trong các giao tác [1]. Tuy nhiên, một hạn chế chủ yếu của FIM là giả định rằng mỗi item không thể xuất hiện nhiều hơn một lần trong giao tác và tất cả các item quan trọng như nhau (cân nặng, lợi nhuận hay giá trị). Những giả định thường không phù hợp với các ứng dụng thực tế. Chẳng hạn, xét 1 CSDL giao tác khách hàng có chứa các thông tin về số lượng các item trong mỗi giao tác và lợi ích của mỗi item. Các thuật toán khai thác FIM sẽ bỏ qua các thông tin này và có thể dẫn đến việc khám phá ra nhiều các itemset ít phổ biến với lợi ích thấp và điều đó dẫn đến thất bại trong việc khám phá ra các tập phổ biến có lợi ích cao.

Bài toán FIM được định nghĩa lại bằng High-Utility Itemset Mining (HUIM) để xem xét các trường hợp mà các item có thể xuất hiện nhiều hơn một lần trong mỗi giao tác và nơi mà mỗi giao tác có đánh trọng số (chẳng hạn như lợi nhuận

một mặt hàng). Mục đích của HUIM là khám phá các tập có lợi ích cao. HUIM có những ứng dụng rộng rãi như website phân tích và các ứng dụng y sinh học [2,7,10]. HUIM cũng được đưa vào những nhiệm vụ khai thác dữ liệu quan trọng khác như khai thác mẫu tuần tự và khai thác lớp dữ liệu có ích cao [9].

Các vấn đề của HUIM gặp nhiều khó khăn hơn so với các vấn đề của FIM. Đối với FIM, thuộc tính bao đóng giảm chỉ ra độ hỗ trợ (support) của một itemset không có tính đơn điệu (anti-monotonic), điều đó có nghĩa là các tập cha của một tập không phổ biến thì không phổ biến và các tập con của một tập phổ biến thì phổ biến. Tính chất này giúp cắt giảm không gian tìm kiếm mạnh mẽ. Đối với HUIM, lợi ích của các itemset thì cũng không đơn điệu hay phản đơn điệu, điều đó có nghĩa là các tập có ích có thể có tập cha hay tập con với lợi ích thấp hơn, bằng hay cao hơn chính nó.Vì vậy, kỹ thuật làm giảm không gian tính toán trong FIM không thể ứng dụng trực tiếp vào HUIM.

Nhiều nghiên cứu đã thực hiện các thuật toán có hiệu quả trên HUIM [2, 6-8,10]. Một hướng tiếp cận phổ biến với HUIM là tìm ra các tập có ích cao bằng 2 pha dựa vào mô hình giao dịch có trọng số giảm TWD (Transaction-Weighted-Downward) [8, 2, 10]. Hướng tiếp cận này sử dụng các thuật toán Two - phase[8], IHUP [2] và UPGrowth [10]. Các thuật toán trước hết tạo ra tập các ứng viên có lợi ích cao bằng đánh giá lợi ích của chúng ở pha 1. Sau đó, trong đó trong pha 2, thuật toán thực thi việc quét cơ sở dữ liệu để đánh giá chính xác lợi ích của các ứng viên và lọc ra các itemset có lợi ích thấp. Gần đây, có nhiều thuật toán hiệu quả hơn được đề xuất để khai thác các tập có ích cao bằng việc sử dụng chỉ 1 pha duy nhất. HUI-Miner[7] làm tốt hơn các thuật toán trước đây và được xem là thuật toán tốt nhất hiện nay cho HUIM [7].Tuy nhiên, công việc khai thác tập có ích cao vẫn còn tốn nhiều thời gian thực thi.Vì vậy, nó vẫn là 1 thách thức quan trọng để thiết kế nhiều thuật toán hiệu quả hơn cho công việc này. Giải thuật FHM[13] tập trung vào thách thức này. Đề xuất của các tác giả dựa trên sự quan sát rằng mặc dù thuật toán HUI-Miner[7] thực hiện 1 pha và vì vậy nó không tạo ra các ứng viên như đối với định nghĩa của mô hình 2 pha, HUI-Miner[7] khám phá không gian tìm kiếm của các

itemset bằng việc tạo ra các itemset và tốn chi phí cho thao tác kết để tính lợi ích của mỗi itemset. Để giảm số lượng phép kết, tác giả để xuất 1 chiến lược cắt giảm có hiệu quả mà không phải thực hiện phép kết.

1.2 TỔNG QUAN VỀ KHAI THÁC DỮ LIỆU

Các khái niệm

Tri thức: là các thông tin tích hợp, bao gồm các sự kiện và mối quan hệ giữa chúng, đã được nhận thức, khám phá, hoặc nghiên cứu.Tri thức có thể được xem như là dữ liệu trừu tượng và tổng quát ở mức độ cao.

Khám phá tri thức: là việc rút trích ra các tri thức chưa được nhận ra, tiềm ẩn trong các tập dữ liệu lớn một cách tự động. Khám phá tri thức trong CSDL là một quá trình gồm một loạt các bước phân tích dữ liệu nhằm rút ra được các thông tin có ích, xác định được các giá trị, quy luật tiềm ẩn trong các khuôn mẫu hay mô hình dữ liệu.

Khai thác dữ liệu: Là quá trình khám phá (rút trích) các tri thức mới và các tri thức có ích ở dạng tiềm ẩn trong lượng lớn dữ liệu được lưu trữ trong các CSDL, kho dữ liệu... Khai thác dữ liệu được dùng kết hợp với kho dữ liệu giúp cho quá trình ra quyết định được chắc chắn hơn.

Khai thác dữ liệu là một bước của quá trình khám phá tri thức (KDP).

1.3 KHÁM PHÁ TRI THỨC VÀ KHAI THÁC DỮ LIỆU

Khám phá tri thức là quá trình tìm ra những tri thức, đó là những mẫu tiềm ẩn, trước đó chưa biết và là thông tin hữu ích đáng tin cậy. Mục đích của khám phá tri thức và KTDL chính là tìm ra các mẫu hoặc mô hình đang tồn tại trong các CSDL nhưng vẫn còn bị che khuất bởi hàng núi dữ liệu.

Khám phá tri thức từ CSDL là một quá trình sử dụng các phương pháp và công cụ tin học, trong đó con người là trung tâm của quá trình. Do đó, con người cần phải có kiến thức cơ bản về lĩnh vực cần khám phá để có thể chọn được tập con

dữ liệu tốt, từ đó phát hiện các mẫu phù hợp với mục tiêu đề ra. Đó chính là tri thức, được rút ra từ CSDL, thường để phục vụ cho việc giải quyết một loạt nhiệm vụ nhất định trong một lĩnh vực nhất định. Tuy vậy, quá trình khám phá tri thức mang tính chất hướng nhiệm vụ vì không phải là mọi tri thức tìm được đều áp dụng vào thực tế được. Để có được những thông tin quý báu chúng ta phải tìm ra các mẫu có trong tập CSDL trước. Việc đánh giá các mẫu được tìm thấy cũng là một điều thú vị và tất yếu có tính chất quyết định đến sự sử dụng hay không sử dụng chúng.

Người ta thường chia quá trình khám phá tri thức gồm các bước sau :

Bước 1: Xác định và định nghĩa vấn đề:

- Tìm hiểu lĩnh vực ứng dụng và nhiệm vụ đề ra, xác định các tri thức đã có và các mục tiêu của người sử dụng.

- Tạo và chọn lựa cơ sở dữ liệu.

Bước 2: Thu nhập và tiền xử lý dữ liệu:

- Xử lý và làm sạch dữ liệu trước: Bỏ đi các dữ liệu tạp bao gồm các lỗi và các dạng không bình thường. Xử lý dữ liệu bị mất, chuyển đổi dữ liệu phù hợp.

- Rút gọn kích thước dữ liệu nhận được: Nhận ra các thuộc tính hữu ích cho quá trình phát hiện tri thức.

Bước 3: Khai thác dữ liệu:

- Chọn nhiệm vụ khai thác dữ liệu.

- Lựa chọn các phương pháp khai thác dữ liệu.

- Khai thác dữ liệu để rút ra các mẫu, các mô hình.

Bước 4: Giải thích kết quả và đánh giá các mẫu, các mô hình tìm được ở bước 3.

Bước 5: Sử dụng tri thức phát hiện được.

- Các tri thức phát hiện được tích hợp chặt chẽ trong hệ thống. Tuy nhiên để sử dụng được tri thức đó đôi khi cần đến các chuyên gia trong các lĩnh vực quan tâm

vì tri thức rút ra có thể chỉ có tính chất hỗ trợ quyết định.

- Tri thức tìm được có thể được sử dụng cho một quá trình khám phá tri thức khác.

Như vậy khám phá tri thức gồm 5 bước chính, trong đó khai thác dữ liệu là bước quan trọng nhất, nhờ đó có thể tìm được các thông tin tiềm ẩn trong cơ sở dữ liệu. Ngoài ra chúng ta cũng thấy được sự khác biệt giữa khám phá tri thức và khai thác dữ liệu.Trong khi khám phá tri thức là nói đến quá trình tổng thể phát hiện tri thức lợi ích từ dữ liệu. Còn KTDL chỉ là một bước trong quá trình khám phá tri thức, các công việc chủ yếu là xác định được bài toán khai thác, tiến hành lựa chọn phương pháp KTDL phù hợp với dữ liệu có được và tách ra các tri thức cần thiết.

Quá trình khai phá dữ liệu

Khai thác dữ liệu (DM - Data mining): là một giai đoạn quan trọng trong quá trình phát hiện tri thức. Về bản chất nó là giai đoạn duy nhất tìm ra được thông tin mới, thông tin tiềm ẩn có trong cơ sở dữ liệu chủ yếu phục vụ cho mô tả và dự đoán. Quá trình khai thác dữ liệu bao gồm các bước chính sau:

+ Xác định nhiệm vụ: Xác định chính xác các vấn đề cần giải quyết.

+ Xác định các dữ liệu liên quan dùng để xây dựng giải pháp.

+ Thu thập và tiền xử lý dữ liệu: Thu thập các dữ liệu liên quan và tiền xử lý chúng thành dạng sao cho thuật toán khai thác dữ liệu có thể hiểu được. Ở đây có thể gặp phải một số vấn đề: dữ liệu phải được sao ra nhiều bản (nếu được chiết xuất vào các tệp), quản lý các tệp dữ liệu, phải lặp đi lặp lại nhiều lần toàn bộ quá trình (nếu mô hình dữ liệu thay đổi, ...).

+ Thuật toán khai thác dữ liệu: Chọn thuật toán khai thác dữ liệu thích hợp và thực hiện việc khai thác dữ liệu: nhằm tìm được các mẫu có ý nghĩa dưới dạng biểu diễn tương ứng với các ý nghĩa đó.

Các loại dữ liệu có thể khai thác

Khai thác dữ liệu có khả năng chấp nhận một số kiểu dữ liệu khác nhau, điển hình là:

- Cơ sở dữ liệu quan hệ (Relational database): Cơ sở dữ liệu tác nghiệp được tổ chức theo mô hình dữ liệu quan hệ. Hầu hết các hệ quản trị cơ sở dữ liệu hiện nay đều hỗ trợ dạng này như MS SQL Server, Oracle, .v.v.

- Cơ sở dữ liệu đa chiều (Multidimensional structures, data warehouses, data mart) là các kho dữ liệu được tập hợp, chọn lọc từ nhiều nguồn dữ liệu khác nhau. Dạng dữ liệu này mang tính lịch sử (tức có tính thời gian) và chủ yếu phục vụ cho quá trình phân tích cũng như là khám phá tri thức nhằm hỗ trợ ra quyết định.

- Cơ sở dữ liệu dạng giao tác (Trasactional database): Là dạng cơ sở dữ liệu tác nghiệp nhưng các bản ghi thường là các giao tác. Dạng dữ liệu này thường phổ biến trong lĩnh vực thương mại và ngân hàng.

- Cơ sở dữ liệu quan hệ - hướng đối tượng (Object-relational database): Là dạng cơ sở dữ liệu lai giữa 2 mô hình quan hệ và hướng đối tượng.

- Dữ liệu không gian và thời gian (Spatial, temporal and time-series data): là dạng dữ liệu có tích hợp thuộc tính về không gian (ví dụ dữ liệu về bản đồ), dữ liệu thời gian (dữ liệu thị trường chứng khoán…).

- Cơ sở dữ liệu đa phương tiện (Multimedia database): Là dạng dữ liệu âm thanh (audio), hình ảnh (Images), phim ảnh (video), Text & WWW,…Dạng dữ liệu này hiện đang rất phổ biến trên Internet do sự ứng dụng rộng rãi của nó.

Các ứng dụng của khai thác dữ liệu

Khai thác dữ liệu có nhiều ứng dụng trong thực tiễn, các ứng dụng điển hình có thể liệt kê như là:

- Phân tích dữ liệu và hỗ trợ ra quyết định.
- Điều trị trong y học: Mối liên hệ giữa triệu chứng, chẩn đoán và phương pháp điều trị.
- Phân lớp văn bản, tóm tắt văn bản và phân lớp các trang Web.
- Tin sinh học: Tìm kiếm, đối sánh các hệ Gene và thông tin di truyền, mối

liên hệ giữa một số hệ Gene và một số bệnh di truyền.

- Nhận dạng.
- Tài chính và thị trường chứng khoán: Phân tích tình hình tài chính và dự báo giá của các cổ phiếu.
- Bảo hiểm.

CHƯƠNG 2
KHAI THÁC TẬP MỤC LỢI ÍCH CAO

2.1 Khai thác dữ liệu truyền thống

Thuật toán về luật kết hợp được được Agarwal và Srikant đề xuất vào năm 1994 mang tên là Apriori. Thuật toán được thiết kế để tính toán trên các cơ sở dữ liệu chứa các giao dịch (ví dụ như dữ liệu bán hàng trong siêu thị hay dữ liệu về các địa chỉ trang web được truy cập).

Apriori dùng cách tiếp cận "bottom up", các tập con phổ biến được sinh ra từ một mục. Mục đích của thuật toán Apriori là tìm ra được tất cả các tập phổ biến có thể có trong CSDL giao tác, Apriori hoạt động theo nguyên tắc quy hoạch động do đó nó sinh ra rất nhiều tập phổ biến và phải duyệt CSDL nhiều lần.

Bảng 2.1: Bảng mô tả các bước thực hiện giải thuật Apriori

1	Duyệt toàn bộ cơ sở dữ liệu để có được độ hỗ trợ S của 1 itemset, so sánh S với độ hỗ trợ nhỏ nhất min_sup, để có được 1-itemset (L_1)
2	Sử dụng L_{k-1} nối L_{k-1} để sinh ra candidate k-itemset. Loại bỏ các itemsets không phải là frequent itemsets thu được k-itemset
3	Duyệt cơ sở dữ liệu giao dịch để có được độ support của mỗi candidate k-itemset, so sánh S với min_sup để thu được frequent k-itemsets (L_k)
4	Lặp lại từ bước 2 cho đến khi Candidate set (C) trống (không tìm thấy frequent itemsets)
5	Với mỗi frequent itemset I, sinh các tập con s không rỗng của I

Sau đó, các hướng nghiên cứu dựa trên cấu trúc cây như FP-Growth đã được đề xuất bởi *Han* cùng các cộng sự. FP-Growth được công nhận rộng rãi là có hiệu suất tốt hơn hướng tiếp cận Apriori do nó tìm tập phổ biến mà không cần phát sinh bất kỳ tập ứng viên nào và chỉ duyệt CSDL gốc hai lần.

Tuy nhiên, trong các luật kết hợp được tìm ra ở Apriori hay FP-Growth thì ***giá trị lợi ích của các item đối với người dùng chưa được xem xét đến***. Vì vậy, đã xuất hiện các phương pháp để khai thác giá trị lợi ích này từ CSDL và được gọi là itemset lợi ích cao, chẳng hạn như các thuật toán Two-Phase[8], TWU-Mining[12], HUI-Miner[7], ….

2.2 Lịch sử phát triển của khai thác tập lợi ích cao

KHAI THÁC TẬP LỢI ÍCH CAO	Năm	Nghiên cứu liên quan	Tác giả
	2014	FHM[13]	Philippe Fournier-Viger
	2012	HUI-Miner	Liu et al
	2009	Efficient Tree Structures for HUI	Ahmed et al
	2008	Based on FP-tree	Erwin et al
	2007	FP-tree	Erwin et al
	2006	Framework	Hamilton et al
	2005	TWU	Liu et al
	2004	Phát biểu bài toán	Hamilton et al

2.3 Giới thiệu bài toán khai thác tập lợi ích cao

Trong mô hình khai thác itemset lợi ích cao, giá trị của mục dữ liệu trong giao tác là một số (chẳng hạn như số lượng đã bán của mặt hàng, gọi là giá trị khách quan), ngoài ra còn có bảng lợi ích cho biết lợi ích mang lại khi bán một đơn vị hàng đó (gọi là giá trị chủ quan, do người quản lý kinh doanh xác định). Lợi ích của một itemset là số đo lợi nhuận của itemset đó trong CSDL, nó có thể là tổng lợi nhuận, là tổng chi phí của itemset.

Khai thác itemset lợi ích cao là khai thác tất cả các itemset X có lợi ích, không nhỏ hơn giá trị ngưỡng tối thiểu quy định bởi người sử dụng. Có thể coi bài toán cơ bản khai thác itemset phổ biến là trường hợp đặc biệt của bài toán khai thác itemset lợi ích cao, trong đó tất cả các item đều có giá trị khách quan bằng 0 hoặc 1

và giá trị chủ quan bằng 1. Các thuật toán khám phá itemset phổ biến được xây dựng theo phương pháp tìm kiếm từng bước. Cơ sở của các thuật toán này là tính chất Apriori hay còn gọi là tính chất phản đơn điệu (anti monotone) của itemset phổ biến. Tính chất đó là "tập con khác rỗng của một itemset phổ biến phải là tập phổ biến". Điều này có nghĩa các (k+1)-itemset phổ biến chỉ có thể sinh ra từ các k itemset phổ biến. Tính chất Apriori cho phép loại bỏ được các tổ hợp item không phổ biến ra khỏi không gian tìm kiếm tại mỗi bước. Đáng tiếc là các itemset lợi ích cao không thỏa mãn tính chất Apriori. Do đó, việc rút gọn không gian tìm kiếm, phát hiện itemset lợi ích cao không thể thực hiện được như trong khai thác itemset phổ biến.

Vì thế thách thức trong việc khai thác lợi ích là giới hạn kích thước của tập ứng viên và đơn giản việc tính toán để tính lợi ích.

2.4. Các cách tiếp cận trong khai thác tập lợi ích cao

- Dựa vào biên trên của độ có ích : Hamilton et al phát triển công thức chặn trên của độ có ích (upper bound utility) và từ đó hình thành nên thuật toán UMining, đồng thời phát triển thuật toán UMining_H để tìm tập lợi ích cao.

- Dựa vào định nghĩa về TWU, các tác giả chứng minh được TWU thỏa tính chất Apriori để từ đó hình thành nên thuật toán Two-Phase dựa trên thuật toán Apriori.

- Dựa trên TWU và FP-tree, Erwin et al phát triển thuật toán khai thác hiệu quả HUIs dựa vào FP-tree.

- Dựa trên TWU và phương pháp cắt tỉa bằng cách ước lượng giá trị lợi ích đồng thời mang tên EUCP (Estimated Utility Cooccurrence Pruning), phương pháp này dùng phép kết mà sinh ra cấu trúc Utility List và EUCS giúp giảm đi phép kết (dùng trong thuật toán HUI-Miner[7] và FHM[13]).

2.5 Các định nghĩa và quy ước trong khai thác tậ p mục lợi ích cao

2.5.1 Định nghĩa 1 (cơ sở dữ liệu giao tác)

Gọi I là 1 tập hợp các item. Một CSDL giao tác là 1 tập các giao tác D = { T_1, T_2, ... , T_n } mà mỗi giao tác T_c, $T_c \in$ I và T_c có 1 định dang c gọi là Tid. Mỗi item i $\in$ I được kết hợp với 1 số dương p(i) gọi là lợi ích bên ngoài. Với mỗi giao tác T_c mà i $\in$ T_c, một số nguyên dương q(i, T_c) được gọi là lợi ích bên trong của i.

Ví dụ: Xét CSDL ở bảng 2.2, CSDL này chứa 5 giao tác (T, T_2, T_3, T_4, T_5). Giao tác T_2 chỉ ra rằng a, c, e và g xuất hiện trong giao tác này với 1 lợi ích bên trong lần lượt là 2, 6, 2 và 5. Bảng 2.2 chỉ ra rằng lợi ích bên ngoài của các các item này lần lượt là 5, 1, 3 và 1

Bảng 2.2: Biểu diễn CSDL giao tác

Tid	Các giao dịch
T_1	(a,1)(c,1)(d,1)
T_2	(a,2)(c,6)(e,2)(g,5)
T_3	(a,1)(b,2)(c,1)(d,6),(e,1),(f,5)
T_4	(b,4)(c,3)(d,3)(e,1)
T_5	(b,2)(c,2)(e,1)(g,2)

Bảng 2.3: Biểu diễn giá trị lợi nhuậ n của các mục

Item	**A**	**b**	**c**	**d**	**e**	**f**	**G**
Lợi nhuận	5	2	1	2	3	1	1

2.5.2 Định nghĩa 2 (lợi ích của itemset trong CSDL)

Lợi ích của 1 item I trong CSDL giao tác T_c được ký hiệu bằng u(i,T_c) và được định nghĩa bằng p(i) $\times$ q(i,T_c). Lợi ích của 1 itemset X (một nhóm các item X $\subseteq$ I) trong 1 giao tác T_c được kí hiệu là u(X,T_c)và được định nghĩa u(X,T_c) = $\sum_{i \in X} u(i,T_c)$

Ví dụ: Lợi ích của 1 item a trong T_2 là u(a,T_2) = 5 $\times$ 2 =10. Lợi ích của itemset { a,c } trong T_2 là u({ a,c },T_2) = u(a,T_2)+ u(c,T_2) = 5 $\times$ 2+ 1 $\times$ 6=16

2.5.3. Định nghĩa 3 (Lợi ích của 1 itemset trong CSDL)

Lợi ích của 1 itemset X được kí hiệu bằng u(X) và định nghĩa bằng u(X) = $\sum_{T_c \in g(X)} u(X,T_c)$ với g(X) là tập giao tác có chứa X.

Ví dụ: Lợi ích của tập giao tác { a, c } là u({ a,c }) = u(a)+u(c) = u(a,T_1) + u(a,T_2)+u(a,T_3)+u(c,T_1) +u(c,T_2)+u(c,T_3)= 5+10+5+1+6+1 = 28.

2.5.4. Định nghĩa 4 (định nghĩa vấn đề)

Bài toán khai thác itemset có ích cao là đi tìm tất cả các itemset có ích cao. Một tập X là một tập có ích cao khi nếu lợi ích của nó u(X) thì không thấp hơn một ngưỡng lợi ích tối thiểu minuntil cho người dùng đưa ra. Ngược lại X là 1 tập có lợi ích thấp.

Ví dụ: Nếu minutil = 30, tập có ích cao trong cơ sở dữ liệu khi chạy là { b,d },

{ a,c,e }, { b,c,d }, { b,c,e }, { b,d,e }, { b,c,d,e } với lợi ích lần lượt là 30, 31, 34, 36,40, 30.

Điều này chứng tỏ rằng lợi ích có nghĩa là không đơn điệu hoặc bất đơn điệu. Ngược lại, 1 itemset có thể có lợi ích thấp hơn, bằng hay cao hơn lợi ích của tập con của nó. Vì vậy, chiến lược là sử dụng FIM để cắt giảm không gian tìm kiếm dựa trên tính chất bất đơn điệu của độ hỗ trợ có thể không ứng dụng trực tiếp để tìm ra itemset có ích. Một vài thuật toán HUIM hạn chế vấn đề này bằng cách đánh giá cao lợi ích của các itemset sử dụng 1 độ đo gọi là Transaction- Weighted Utilization (TWU) [2, 8, 10], chính là tính chất bất đơn điệu. Độ đo TWU được định nghĩa như sau.

2.5.5. Định nghĩa 5 (Lợi ích của giao tác)

Lợi ích giao tác (TU) của một giao tác T_c là tổng lợi ích của các các item từ T_c trong T_c, tức la $TU(T_c) = \sum_{x \in T_c} u(x, T_c)$.

Ví dụ:

Bảng 2.4: Giá trị TU của các giao tác T_1, T_2,T_3, T_4,T_5 khi thực thi

TID	TU
T1	8
T2	27
T3	30
T4	20
T5	11

Bảng 2.5: Giá trị TWU của các item khi thực thi

ITEM	TWU
a	65
b	61
c	96
d	58
e	88
f	30
g	38

2.5.6. Định nghĩa 6 (Lợi ích trọng số của giao dịch)

Lợi ích trọng số giao dịch (TWU) của 1 itemset X được định nghĩa bằng tổng lợi ích giao tác của các giao tác chứa X, tức là $TWU(X) = \sum_{T_c \in g(X)} TU(T_c)$.

Ví dụ : Bảng 2.4 chỉ ra TWU của các item đơn a, b, c, d, e, f, g. Xét item a.$TWU(A) = TU(T_1)+TU(T_2)+TU(T_3) = 8+27+30 = 65$.

Độ đo TWU có 3 tính chất quan trọng được dùng để cắt giảm không gian tìm kiếm.

Tính chất 1 (overestimation) TWU của 1 itemset X thì cao hơn hoặc bằng lợi ích của nó, tức là $TWU(X) \geq u(X)$ [8].

Tính chất 2 (antimonotonicity) TWU thì có tính chất bất đơn điệu. Lấy X và Y là 2 itemset.Nếu $X \subset Y$ thì $TWU(X) \geq TWU(Y)$ [8].

Tính chất 3 (pruning) Lấy X là 1 tập giao tác. Nếu TWU(X) < minutil, khi ấy itemset X cũng như các tập con của nó là 1 itemset có lợi ích thấp cũng như các tập con của nó.

2.5.7. Định nghĩa 7 (danh sách giá trị lợi ích UL)

Gọi $\succ$ là 1 thứ tự toàn phần trên các item của I. UL của một itemset X trong CSDL D là 1 tập các bộ bao gồm (tid, iutil, rutil) cho mỗi giao tác T_{tid} chứa X.

+ **Thành phần iutil trong UL** là 1 bộ là lợi ích của X trong T_{tid}. Tức là $u(X,T_{tid})$.

+ **Thành phần rutil** của 1 bộ được định nghĩa bằng $\sum_{i \in T_{tid} \wedge i \notin X} U(i,T_{tid})$.

Ví dụ: UL của { a } là { (T_1, 5, 3)(T_2, 10, 17)(T_3, 5, 25) }.UL của { e } là { (T_2, 6, 5),(T_3, 3, 5)(T_4, 3, 0) }. Utility-list của { a, e } là { (T_2, 16, 5),(T_3, 8, 5) }.

Tính chất 4 (tổng của iutil). Lấy X là một itemset. Nếu tổng của giá trị iutil trong UL của x cao hơn hoặc bằng minutil, khi ấy X là itemset có lợi ích cao. Ngược lại, nó là itemset có lợi ích thấp [7].

Tính chất 5 (tổng của iutil và rutil).Lấy X là một itemset. Lấy một mở rộng của X là các itemset có thể thu được bằng việc thêm vào một item y vào X sao cho $y \succ I$ cho tất cả các item i trong X. Nếu tổng của iutil và riutil trong UL của x nhỏ hơn minutil, tất cả các mở rộng của X và các mở rộng bắc cầu cảu nó là các itemset có lợi ích thấp [7].

2.6 Thuật toán Two-phase [8]

2.6.1 Giới thiệu

Thuật toán khai thác các luật kết hợp truyền thống không đáp ứng yêu cầu phát sinh từ các ứng dụng thực tế. Bằng cách xem các giá trị của từng item là các giá trị lợi ích, thuật toán này tập trung định nghĩa các tập item có chất lượng tốt hơn. Thuật toán Two-phase có thể làm tinh gọn số phần tử của tập dự tuyển hiệu quả hơn và từ đó thu được các tập item có chất lượng tốt hơn. Liu và các cộng sự đã đề xuất mô hình khai thác dữ liệu dựa trên trọng số lợi ích (*transaction-weighted utilization mining*). Mô hình này sử dụng tính chất về ***trọng số giao dịch đóng giảm*** (*Transaction-weighted Downward Closure Property* - TWDCP).

2.6.2 Thuật toán Two-phase

Thuật toán xử lý qua hai giai đoạn

*** Giai đoạn 1:**

- Tìm tất cả tập item có giá trị lợi ích lớn hơn giá trị ngưỡng do người dùng định nghĩa dựa trên trọng số lợi ích của giao dịch (*Transaction-Weighted Utilization - TWU*). Trong giai đoạn 1 chỉ có những kết hợp của những tập item có trọng số giao dịch có độ lợi ích cao mới được thêm vào tập ứng viên trong suốt quá trình tìm kiếm thông minh trên mỗi mức. Tuy các tập item có độ lợi ích thấp có thể được đánh giá cao nhưng thuật toán lại không đánh giá thấp bất kỳ tập item nào.

*** Giai đoạn 2:**

Thuật giải duyệt lại CSDL để lọc lại các tập item được đánh giá cao nhờ vào tính chất TWDCP.

2.6.3 Nhận xét

So với các thuật toán khai thác lợi ích cao hiện nay, thuật toán Two-phase gặp vấn đề là một số lượng rất lớn các tập ứng viên được tạo ra nhưng hầu hết các ứng cử viên được sinh ra là lợi ích không cao sau khi các lợi ích này được tính chính xác ở giai đoạn 2 của thuật toán.

2.7 Thuật toán TWU-Mining [12]

2.7.1 Giới thiệu

Thuật toán TWU-Mining gồm hai giai đoạn xử lý và được gọi là phase 1 và phase 2: Trong phase 1 thuật toán sẽ tìm tất cả các itemset có trọng số độ lợi ích TWU theo giao tác có độ lợi ích không nhỏ hơn *minutil*. Trong phase 2 thuật toán sẽ tìm tất cả các itemset có độ lợi ích cao trên từng giao tác.

2.7.2 Thuậ t toán TWU-Mining

Thuật toán TWU-Mining được xây dựng trên nền tảng cây WIT-Tree, cấu trúc WIT-Tree chi tiết được trình bày như sau.

Ví dụ: xét CSDL A như sau :

Bảng 2.6 CSDL A

Item / Giao tác	A	B	C	D	E	F	G	H
T_1	1	0	10	1	0	0	0	0
T_2	2	0	6	0	2	0	5	0
T_3	2	2	0	6	2	1	0	0
T_4	0	4	13	3	1	0	0	0
T_5	0	2	4	0	1	0	2	0
T_6	1	1	1	1	0	0	0	2

Bảng 2.7 Lợi nhận của các item trong CSDL A

Item	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Lợi nhuận	5	2	1	2	3	5	1	1

Trong WIT-Tree,mỗi nút***N*** bao gồm ***N.name***, ***N.tidset***, ***N.twu***, với :

N.name là tên item của nút trong cây.

N.tidset $N.tidset$ là tập số thứ tự của các giao tác chứa nút.

N N , ***N.twu*** $N.twu$ là tổng độ lợi ích của các item trên tất cả các giao tác. Giá trị của TWU đã được tính chi tiết theo.

Bảng 2.8 Trọng số độ lợi ích TWU theo giao tác.

Item	A	B	C	D	E	F	G	H
TWU	93	92	99	96	107	37	40	12

Nguyên tắc sinh cây WIT-Tree

Xây dựng các k-itemset trong đó k có giá trị từ 1 đến m, với m là số item trong CSDL giao tác. Ở mỗi bước của quá trình là tạo ra các itemset sao cho các item chung trong itemset phải cùng tham gia trong giao tác của CSDL A. Lợi ích của mỗi nút trong cây WIT-Tree sẽ là tổng độ lợi ích của giao tác mà item có tham gia như Bảng 2 .9, các bước thực hiện như sau:

Bước 1: Với k = 1 (*1-itemset*)

Bảng 2.9 WIT-Tree với 1-itemset

1-itemset	TWU	Các giao tác tham gia
{ A }	93	1,2,3,6
{ B }	92	3,4,5,6
{ C }	99	1,2,4,5,6
{ D }	96	1,3,4,6
{ E }	107	2,3,4,5
{ F }	37	3
{ G }	40	2,5
{ H }	12	6

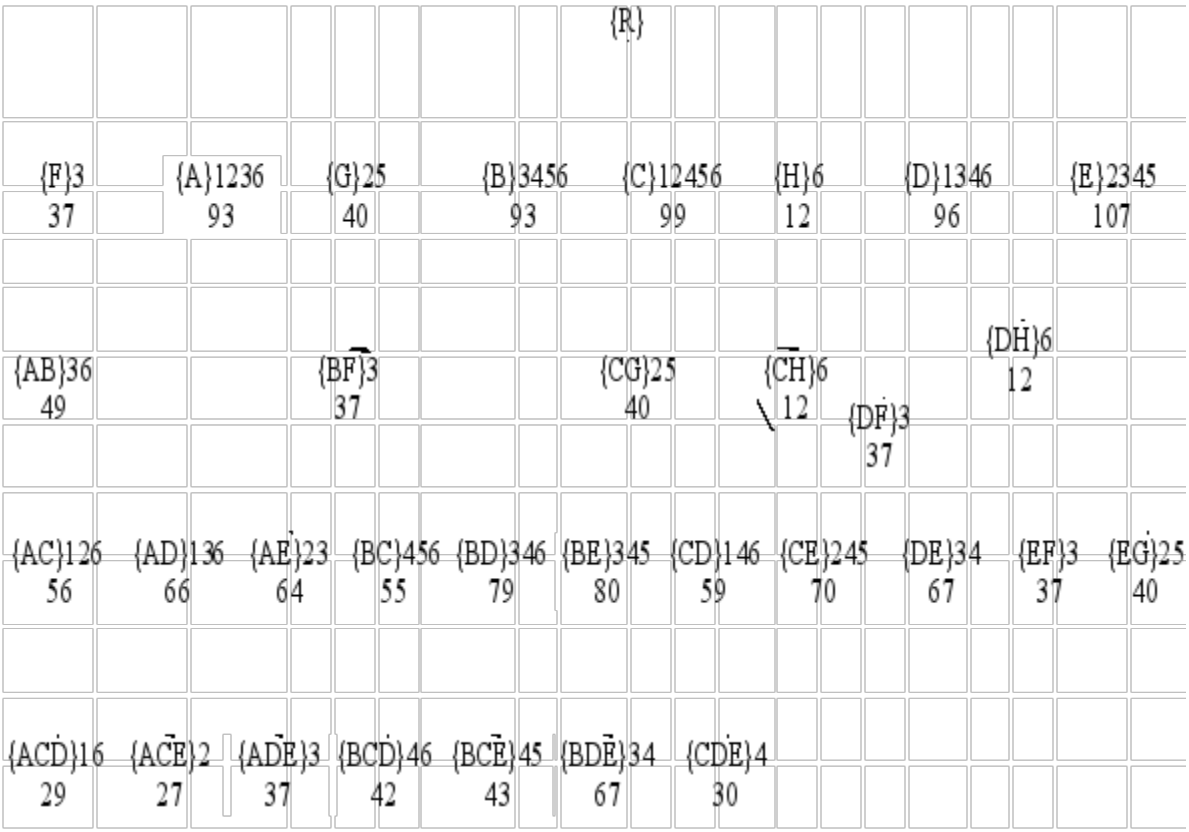
– **Bước 2**: Với k = 2 (*2-itemset*)

Bảng 2.10 WIT-Tree với 2-itemset

1-itemset	TWU	Các giao tác tham gia
{ AB }	49	3,6
{ AC }	56	1,2,6
{ AD }	66	1,3,6

{ AE }	64	2,3
…		

❖ Sau đó cây WIT-Tree hoàn chỉnh được tạo ra theo



Hình 2.1 Cây WIT-Tree với minutil = 50

Thuật toán TWU-Mining [12]

Thuật toán TWU-Mining được phát triển trên cơ sở nền tảng cây WIT-Tree để tìm ra itemset lợi ích cao với trọng số độ lợi ích theo giao tác *minutil*.

Hình 2.2 Thuật toán TWU-Miningđược phát biểu như sau [10]:

TWU-Mining()

HUIs = **[]** = { *it*(*i*)| *i I twu*(*i*)*minulti* }

TWU-Mining-Extend(**[]**, *minulti*)

TWU-Mining-Extend (**[***P***]**, *minulti*)

// Phase 1

for all $l_i \geq$ **[***P***]** do

1. **[***P*$_i$**]** = ∈

2. for all $l_j \varnothing$ **[***P***]**, with j > i do

$X = l_i \in l_j$

$Y = Tidset(l_i) \cup Tidset(l_j)$

if *twu*(*X*) ∩*minulti* then

$[P_i] = [P_i] \geq \{ \frac{X \times Y}{twu(X)} \}$

3. **TWU-Mining-Extend**(**[***P*$_i$**]**, *minulti*)

// Phase 2

foreach itemset *s* in **[***P***]** do

4. If *u*(*s*) ∪*minulti* then

HUIs = HUIs ≥ s

Trong giai đoạn phase 1 thực hiện duyệt CSDL sẽ tạo được cây WIT-Tree, trong phase 2 sẽ duyệt cây WIT-Tree để phát sinh cát PHUI và tìm ra các tập HUI thỏa ngưỡng minulti.

Với PHUI sau giai đoạn 1 gồm 15 itemset { { A }, { B }, { C }, { D }, { E }, { AC }, { AD }, { AE }, { BC }, { BD }, { BE }, { CD }, { CE }, { DE }, { BDE }.

Với TWU-Mining các itemset trong PHUI thu được lớn nhưng số lần duyệt CSDL ít hơn Two-Phase. Sau khi tính lại TWU thực sự của các itemset trong PHUI, itemset thực sự đạt ngưỡng minulti là $\{BDE\}$.

2.8 Thuật toán HUI-Miner[7]

2.8.1 Giới thiệu thuật toán

Để xác định tập lợi ích cao, hầu hết các thuật toán đầu tiên tạo ra tập ứng cử viên từ cách đánh giá các lợi ích cao và sau đó tính toán các lợi ích chính xác của các ứng cử viên. Các thuật toán này gặp những vấn đề là tạo ra một số lượng lớn tập ứng viên nhưng hầu hết các ứng cử viên được sinh ra là lợi ích không cao sau khi các lợi ích được tính chính xác. HUI- Miner (High Utility Itemset Miner) sử dụng một cấu trúc mới, được gọi là danh sách lợi ích, để lưu trữ tất cả các thông tin hữu ích về một tập và tìm ra thông tin để cắt tỉa không gian tìm kiếm của HUI- Miner. Bằng cách tránh tạo ra các tập ứng viên thế hệ và tính toán lợi ích của nhiều tập ứng viên, HUI- Miner hiệu quả hơn vì có thể khai thác tập lợi ích cao từ danh sách lợi ích (utility list)

2.8.2 Thuật toán HUI-Miner[7]

Giải thuật có 2 bước xử lý chính sau:

- Bước 1: Tạo danh sách các tiện ích UL
- Bước 2: Tìm các tập mục thỏa giá trị ngưỡng *minutil*

Bước 1: Tạo danh sách các tiện ích UL

Đầu vào:

1. ***P.UL***: UL của tập mục *P*;

2. ***Px.UL***: UL của tậ p mục*Px*;

3.***Py.UL***: UL của tậ p mục*Py*.

Đầu ra: ***Pxy.UL***, UL của tập mục *Pxy*.

1 *Pxy.UL = NULL*;

2 foreach element *Ex* ∈ *Px.UL* **do**

3 **if** ∃*Ey*∈*Py.UL* **and** *Ex.tid==Ey.tid* **then**

4 **if** *P.UL* is not empty **then**

5 search such element *E* ∈*P.UL* that E.*tid ==Ex.tid*;

6 *Exy =<Ex.tid, Ex.iutil +Ey.iutil -E.iutil, Ey.rutil >;*

7 **else**

8 *Exy =<Ex.tid, Ex.iutil +Ey.iutil, Ey.rutil >;*

9 **end**

10 append *Exy to Pxy.UL;*

11 **end**

12 end

13 return *Pxy.UL;*

Bước 2: Tìm các tập mục thỏa giá trị ngưỡng *minutil*

Đầu vào:

1. *P.UL*: UL của tập item *P*, khởi gán ban đầu rỗng,

2. ULs : UL của tập P trong tham số đầu vào thứ 1

3. *minutil :* giá trị ngưỡng.

Đầu ra: Tất cả tập mục có giá trị lợi ích cao

1 foreach utility-list *X* in *ULs* **do**

2 **if** SUM(*X.iutil*)≥*minutil***then**
3 xuất ra giá trị của item tương ứng trong tậ p*X*;
4 **end**
5 **if** SUM(*X.iutil*)+SUM(*X.rutil*)≥*minutil***then**
6 *exULs = NULL*;
7 **foreach** utility-list *Y* after *X* in *ULs***do**
8 *exULs= exULs* +Construct(*P.UL, X, Y*);
9 **end**
10 HUI-Miner(*X, exULs, minutil*);
11 **end**
12end

Ví dụ:

Xét cơ sở dữ liệu giao dịch minh họa ở bảng 2.1 và giá trị lợi ích ở bảng 2.2

Bước 1: Tạo danh sách UL :

+ Tính giá trị UL của tập item có 1 phần tử :

* Giá tri UL của item a

Bảng 2.11 : giá trị UL của { a }

Tid	Giá trị iulti	Giá trị rulti
T1	5	3
T2	10	17
T3	5	25
Tổng	**20**	**45**

* Giá tri UL của item b

Bảng 2.12 : giá trị UL của { b }

Tid	Giá trị iulti	Giá trị rulti
T3	4	21
T4	8	12
T5	4	7
Tổng	**16**	**40**

* Giá tri UL của item c

Bảng 2.13 : giá trị UL của { c }

Tid	Giá trị iulti	Giá trị rulti
T1	1	2
T2	6	11
T3	1	20
T4	3	9
T5	2	5
Tổng	**13**	**47**

* Giá tri UL của item d

Bảng 2.14 : giá trị UL của { d }

Tid	Giá trị iulti	Giá trị rulti
T1	2	0
T3	12	8
T4	6	3
Tổng	**20**	**11**

* Giá tri UL của item e

Bảng 2.15 : giá trị UL của { e }

Tid	Giá trị iulti	Giá trị rulti
T2	6	5
T3	3	5
T4	3	0
T5	3	2
Tổng	**15**	**12**

* Giá tri UL của item f

Bảng 2.16 : giá trị UL của { f }

Tid	Giá trị iulti	Giá trị rulti
T3	5	0
Tổng	**5**	**0**

* Giá tri UL của item g

Bảng 2.17 : giá trị UL của ({ g })

Tid	Giá trị iulti	Giá trị rulti
T2	5	0
T5	2	0
Tổng	**7**	**0**

+ Tính giá trị UL của tập item có 2 phần tử:

* Giá trị UL của { d,b }

Bảng 2.18 : giá trị UL của { d,b }

Tid	Giá trị iulti	Giá trị rulti
T3	16	8
T4	14	3
Tổng	**30**	**11**

* Giá trị UL của { d,a }

Bảng 2.19 : giá trị UL của { d,a }

Tid	Giá trị iulti	Giá trị rulti
T1	7	0
T3	17	8
Tổng	**24**	**8**

* Giá trị UL của { d,e }

Bảng 2.20 : giá trị UL của { d,e }

Tid	Giá trị iulti	Giá trị rulti
T3	15	5
T4	9	0

Tổng	**26**	**5**

* Giá trị UL của { d,c }

Bảng 2.21 : giá trị UL của { d,c }

Tid	Giá trị iulti	Giá trị rulti
T1	3	0
T3	13	8
Tổng	**26**	**8**

+ UL của itemset k phần tử (k ≥ 3)

Để xây dựng các danh sách lợi ích của k- itemset { $i_1 \cdots i_{(k-1)} i_k$ } (k ≥ 3), chúng ta có thể trực tiếp lấy giao của danh sách lợi ích { $i_1 \cdots i_{(k-2)} i_{(k-1)}$ } và { $i_1 \cdots i_{(k-2)} i_k$ } như chúng ta xây dựng các danh sách lợi ích từ tập 2-itemsets.

Ví dụ, để xây dựng các danh sách lợi ích { d,b,a }, chúng ta có thể lấy phần giao các danh sách lợi ích của itemset { d,b } và itemset { d,a } như trong bảng 2.18 và 2.19 và tập { d,b,a } xuất hiện trong T3 được tính như bảng 2.22

Bảng 2.22 : giá trị UL của { d,b,a }

Tid	Giá trị iulti	Giá trị rulti
T3	21	8
Tổng	**21**	**8**

Bước 2: Tìm những item thỏa giá trị ngưỡng minulti:

* Không gian tìm kiếm những itemset được tạo ra như sau:

Ta xét bảng 2.4 thể hiện giá trị TWU của các item. Giả sử cho **minulti ≥ 50**, ta được các item ban đầu thỏa điều kiện và được sắp xếp theo thứ tự tăng dần như sau: **d < b < a < e < c**

* Lần lượt tính các giá trị tổng của iulti và rulti của tập item có 1 phần tử, 2 phần tử và k phần tử (k≥3) để tìm ra các tập tiện ích cao.

Ví dụ: tìm tập tiện ích cao ở CSDL giao tác minh họa ở Bảng 2.2 với **minulti ≥ 50**

- Đầu tiên xét item d: có giá trị tổng iulti = 20 và rulti = 45, ta có iulti + rulti = 65 ≥ minulti nên:

+ Gán tập *exULs = NULL*

+ Lần lượt hội item d với các phần tử đứng phía sau d trong quan hệ thứ tự **d < b < a < e < c** để tạo ra tập item mới**:** { d,b }, { d,a },{ d,e },{ d,c }, tính UL cho các tập mới phát sinh này.

UL1({ d,b }) = (30,11): có iulti + rulti = 41 < 50 không đưa vào *exULs*

UL2({ d,a }) = (24,8) : có iulti + rulti = 32 < 50 không đưa vào *exULs*

UL3({ d,e }) = (26,5) : có iulti + rulti = 31 < 50 không đưa vào *exULs*

UL4({ d,c }) = (26,8) : có iulti + rulti = 34 < 50 không đưa vào *exULs*

Các tập { d,b,a }, { d,b,e },{ d,b,c }: không đưa vào UL do UL1({ d,b }) = (30,11): có iulti + rulti = 41 < 50. Tương tự cũng không xét tập { d,a,e },{ d,a,c }, { d,e,c }.

- Xét item b: có giá trị giá trị iulti = 16 và rulti = 40, ta có iulti + rulti = 56 ≥ minulti nên:

+ Lần lượt hội item b với các phần tử đứng phía sau b trong quan hệ thứ tự **d < b < a < e < c** để tạo ra tập item mới**:** { b,a }, { b,e },{ b,c }, tính UL cho các tập mới phát sinh này.

+ Xét tập item 2 phần tử :

UL11({ b,a }) = (28,8): có iulti + rulti = 36 < 50 không đưa vào *exULs*

UL12({ b,e }) = (25,7): có iulti + rulti = 32 < 50 không đưa vào *exULs*

UL13({ b,c }) = (22,31): có iulti + rulti = 53 > 50. Tuy nhiên ta không thể mở rộng tập UL13({ b,c }) vì phía sau không còn phần tử nào.

- Xét item a: có giá trị giá trị iulti = 20 và rulti = 45, ta có iulti + rulti = 65 ≥ minulti ta ần lượt hội item a với các phần tử đứng phía sau b trong quan hệ thứ tự **d < b < a < e < c** để tạo ra tập item mới**:** { a,e }, { a,c }, tính UL cho các tập mới phát sinh này

UL14({ a,e }) = (24,10): có iulti + rulti = 34 < 50 không đưa vào *exULs*

UL15({ a,c }) = (28,33): có iulti + rulti = 61 > 50. Tuy nhiên ta không thể mở rộng tập UL15({ a,c }) vì phía sau không còn phần tử nào.

- Xét item e: có giá trị giá trị iulti = 15 và rulti = 12, ta có iulti + rulti = 27 < minulti, nên không mở rộng các phần tử sau item a.

- Xét item c: có giá trị giá trị iulti = 13 và rulti = 47, ta có iulti + rulti = 60. Tuy nhiên ta không thể mở rộng tập item { c } vì phía sau không còn phần tử nào.

2.9 Thuật toán FHM[13]

2.9.1 Thủ tục chính

Giải thuật 1

Đầu vào:

1. ***D***: tập cơ sở giao dịch.

2. ***minutil***: giá trị ngưỡng

Đầu ra: tập chứa các tập mục có giá trị lợi ích cao

1. Quét cơ sở dữ liệu để tính giá trị của TWU cho từng mục.

2. Lưu các mục i có giá trị TWU < *mintil* vào tập I^*

3. Tạo quan hệ thứ tự toàn phần trên I^* với giá trị của các phần tử trong I được sắp tăng dần theo TWU

4. Quét qua D để tạo UL của mỗi mục $i \in I^*$ và xây dựng cấu trúc EUCS

5. Thực hiện thủ tục tìm kiếm các mục lợi ích cao

Search(Ø, I^*, minutil,EUCS)

Thuật toán trước tiên quét CSDL để tính TWU cho mỗi item. Sau đó, thuật toán nhận ra tập I^* của tất cả các item có TWU không nhỏ hơn minutil (các item khác được bỏ đi vì chúng không phải là thành phần của các tập có lợi ích cao bằng tính chất 3). Giá trị TWU của các item sau đó được sử dụng để thiết lập 1 thứ tự toàn phần $\succ$ trên các item, đây là thứ tự giảm dần của các TWU (như gợi ý trong bài báo [7]). Lần quét CSDL thứ hai được tiến hành. Trong suốt quá trình quét CSDL này, các item trong giao tác được sắp xếp lại theo thứ tự toàn phần $\succ$, utility-list của mỗi item $I \in I^*$ được xây dựng và cấu trúc đó được đặt tên là EUCS

(Estimated Utility Co-occurrence Structure). Cấu trúc mới này được định nghĩa như 1 bộ 3 có dạng (a, b, c) $\in$ $I^* x I^* x \mathbb{R}$. Bộ 3 (a, b, c) chỉ ra rằng TWU({ a,b }) = c. EUCS có thể được triển khai như ma trận 3 chiều được mô tả trong bảng Bảng 2.24 hoặc như 1 bảng băm với các bộ có dạng (a, b, c) sao cho c $\neq$ 0 được giữ lại. Tác giả đã sử dụng cấu trúc mới này để đạt hiệu quả về bộ nhớ cao hơn vì tác giả quan sát thấy rằng có ít các item cùng xuất hiện với các item khác. Xây dựng EUCS rất nhanh và chiếm ít bộ nhớ, giới hạn khoảng $|I^*| x |I^*|$, dù thực tiễn kích thước nhỏ hơn nhiều vì số lượng giới hạn của các cặp item xuất hiện đồng thời trong các giao tác. Sau khi xây dựng EUCS, thuật toán tìm kiếm theo chiều sâu bắt đầu gọi thủ tục đệ quy Search với itemset rỗng = $\varnothing$, tập các item đơn I^*, minutil và cấu trúc EUCS.

Giải thuật 2: Thuật toán Search

Đầu vào:

1. ***P***: tập item *P*
2. ***ExtensionofP***: tập các phần mở rộng của *P*
3. ***minutil***: giá trị ngưỡng.
4. ***EUCS***: cấu trúc EUCS

Đầu ra: Tất cả tập mục có giá trị lợi ích cao

1. Với mỗi P_x thuộc phần mở rộng của tập P ta tính các giá trị *iulti,* nếu P_x.UL. *iulti* $\geq$ *minulti* : xuất ra kết quả là P_x

2. Nếu SUM(P_x.UL. *iulti)* + SUM(P_x.UL. *rulti)* $\geq$ *minutil:*

   2.1 Gán *ExtensionofP* = $\varnothing$

   2.2 Duyệt qua các phần tử Py $\in$ *ExtensionofP:*

   + *Nếu* $\exists(x, y, c) \in EUCS$ *mà* $\geq minutil$:

   - P_x <- P_x $\cup$ P_y

- Tính UL của P_{xy}: *Construct(P, P_x, P_y)*

- $ExtensionofP_x = ExtensionofP_x \cup P_{xy}$

2.3 Thực hiện tìm trong tập $ExtensionofP_x$ các mục thỏa minutil

3. Thực hiện thủ tục search: (PX, $ExtensionofP_x$, minutil)

Thủ tục Search (Thuật toán 2) nhận đầu vào là (1) itemset P, (2) mở rộng của P có dạng Pz nghĩa là Pz thu được trước đó bằng việc thêm 1 item z vào P, (3) minutil và (4) EUCS. Thủ tục search thực hiện như sau:

Với mỗi mở rộng Px của P, nếu tổng giá trị iutil của UL của Px nhỏ hơn minutil, điều đó có nghĩa là mở rộng của Px nên được khám phá. Điều này được thực hiện bằng việc gộp Px với tất cả các mở rộng Py của P sao cho $y \succ x$ để tạo nên dạng mở rộng Pxy chứa |Px| +1 item. UL của Pxy được xây dựng như thuật toán HUI-Miner[7] bằng việc gọi thủ tục Construct (Thuật toán 3) để kết utility-list của P, Px và Py. Thủ tục này cũng giống với HUI-Miner[7]. Sau đó, 1 lời gọi đệ quy gọi thủ tục Search để hoàn thành việc tính toán lợi ích của Pxy và khám phá các tập mở rộng của nó. Vì thủ tục Search bắt đầu từ các item đơn, nó khám phá đệ quy không gian tìm kiếm của các itemset bằng việc thêm vào các item đơn và nó chỉ cắt giảm không gian tìm kiếm dựa vào tính chất 5. Có thể dễ dàng thấy trong tính chất 4 và 5 rằng thủ tục này thì chính xác và hoàn toàn tìm ra các itemset có lợi ích cao.

<u>Giải thuật 3</u>: Giải thuật Contruct

Đầu vào:

1. ***P***: tập item *P*

2. $\boldsymbol{P_x}$: tập các phần mở rộng của P với item x

2. $\boldsymbol{P_y}$: tập các phần mở rộng của P với item y

Đầu ra: Tất cả tập mục có giá trị lợi ích cao

1 UtilityListOf$\boldsymbol{P_{xy}$ □ ∅ ;

2 **foreach** *tuple ex* ∈ $\boldsymbol{P_x}$.*utilitylist* do

3 **if** ∋*ey* ∈ $\boldsymbol{P_y}$.*utilitylist a and ex.tid = exy.tid then*

4 **if** *P.utilitylist* ≠ 0 **then**

5 Search element e ∈ P.*utilitylist* such that *e.tid* = *ex.tid*.;

6 *exy* ⇐ *(ex.tid,ex.iutil - e.iutil, ey.rutil);*

7 **end**

8 **else**

9 *exy* ⇐ *(ex.tid,ex.iutil + ey.iutil, ey.rutil);*

10 **end**

11 *UtilityListOf*$\boldsymbol{P_{xy}$ ⇐ *UtilityListOf*$\boldsymbol{P_{xy}$ ∪ *{ exy };*

12 **end**

13 **end**

14 Return UtilityListOf$\boldsymbol{P_{xy;}$

2.9.2. Ví dụ minh họa thuật toán FHM[13]

Cơ sở dữ liệu giao dịch sử dụng như trong bảng 2.2 và các giá trị lợi ích ngoài như bảng 2.3.

*** Các bước thực hiện thuật toán 1**

Cho giá trị minulti = 20

1. Quét CSDL giao dịch

2. Tính TWU:

Bảng 2.23 : kết quả tính TWU cho các item

ITEM	TWU
a	65
b	61
c	96
d	58
e	88
f	30
g	38

3. Lấy các item có TWU ≥ 20 và sắp tăng dần theo TWU: f < g < d < b < a < e < c

Ta có tập I*({ f,g,d,b,a,e,c })

4. Xây dựng cấu trúc EUCS

EUCS được triển khai như ma trận như sau:

Lấy lần lượt tính TWU của các item đơn của dòng và cột trong ma trận tam giác dưới.

Ví dụ : TWU({ g,b }) = 11, TWU({ b,e }) = 61

Bảng 2.24 : kết quả tính bảng EUCS

	f	g	d	b	a	e
g	0					
d	30	0				
b	30	11	50			
a	30	27	38	30		
e	30	38	50	61	27	
c	30	38	58	61	65	88

*** Các bước thực hiện thuật toán 2 (thuật toán Search)**

1. Gán tập chứa kết quả Q = ∅

2. Ta lần lượt tính giá trị UL của các item

UL({ f }) = (5,0) có *iulti = 5 < minulti* nên không khám phá phần mở rộng của item f.

UL({ g }) = (7,0) có *iulti = 7 < minulti* nên không khám phá phần mở rộng của item g.

UL({ d }) = (20,11) có *iulti = 20 ≥ minulti* nên tập item { d } là itemset lợi ích cao

→ *gán Q = ({ d })*

Ta thấy giá trị iulti+rulti của tập { d } = 31 ≥ minulti nên ta khám phá các tập mở rộng của tập { d } **lần lượt theo thứ tự là b,a,e,c**.

- Xét tập X ({ d,b }) trong bảng EUCS *có giá trị TWU = 50 ≥ minutil* nên ta tính giá trị utility list của tập X.

Ta có: UL(X) = (30,11) có *iulti = 30 ≥ minulti* nên tập X({ d,b }) là itemset lợi ích cao

→ ***thêm { d,b } vào tập Q: Q = ({ d },{ d,b })***

- Xét X ({ d,a }) trong bảng EUCS *có giá trị TWU = 38 ≥ minutil* nên ta tính giá trị utility list của tập X({ d,a }).

Ta có: UL({ d,a }) = (24,8) có *iulti = 24 ≥ minulti* nên tập item X({ d,a }) là itemset lợi ích cao.

→ ***thêm { d,a } vào tập Q: Q = ({ d },{ d,b },{ d,a })***

Xét X({ d,e }) trong bảng EUCS *có giá trị TWU = 50 ≥ minutil* nên ta tính giá trị utility list của tập X.

Ta có: UL({ d,e }) = (26,5) có *iulti = 26 ≥ minulti* nên tập item X là itemset lợi ích cao.

→ ***thêm { d,e } vào tập Q: Q = ({ d },{ d,b },{ d,a },{ d,e })***

Xét X({ d,c }) trong bảng EUCS *có giá trị TWU = 58 ≥ minutil* nên ta tính giá trị utility list của tập X.

Ta có: UL({ d,c }) = (26,8) có *iulti = 26 ≥ minulti* nên tập item X là itemset lợi ích cao

→ ***thêm { d,e } vào tập Q: Q = ({ d },{ d,b },{ d,a },{ d,e },{ d,c })***

- Bước kế tiếp ta tính UL của các tập gồm có 3 item là phần giao của tập 2 item lần lượt với các item đứng phía sau tập đang xét

Mở rộng tập X = ***{ d,b }*** với item a ta được K = ***{ d,b,a }***

Ta có: UL({ d,b,a }) = (21,8) có *iulti = 21 ≥ minulti* nên tập item K là itemset lợi ích cao

→ ***thêm K vào tập Q: Q = ({ d },{ d,b },{ d,a },{ d,e },{ d,c },{ d,b,a })***

- Các tập 3 item còn lại ***{ d,b,e },{ d,b,c }*** cũng có cách tìm HUI tương tự.

Lần lượt ta có thể tính được giá trị UL của các item còn lại là: b,a,e,c và tìm ra các itemset lợi ích cao từ các item và tập mở rộng tương ứng của các item này ta có tập Q như sau:

Q=({d},{d,b},{d,a},{d,e},{d,c},{d,b,a},{d,b,c},{d,a,e},{d,a,c},{d,e,c},{d,b,e,c},
{d,a,e,c},{b,e},{b,c},{a,e},{a,c},{b,e,c})

2.9.3 Nhận xét

Sự cắt tỉa dựa trên xuất hiện đồng thời (Co-occurrence-based Pruning) là tính mới của FHM[13] là cơ chế cắt giảm hiệu quả đặt tên là EUCP (Estimated Utility Co-occurrence Pruning), nó dựa trên 1 cấu trúc mới là EUCP. EUCP dựa trên quan sát rằng một trong những thao tác tốn kém của HUI-Miner[7] là thao tác kết. EUCP là 1 chiến lược cắt tỉa để loại bỏ trực tiếp các mở rộng Pxy có lợi ích thấp và tất cả các mở rộng bắc cầu mà không phải đi xây dựng utility-list của chúng. Điều này được thể hiện ở dòng 8 của thủ tục Search. Điều kiện để cắt bỏ là nếu có 1 bộ (x, y, c) trong EUCS sao cho $c \geq$ minutil, khi ấy Pxy và tất cả các tập cha của nó là các itemset có lợi ích thấp và không cần phải khám phá chúng.

Chiến lược này đúng vì chỉ cắt bỏ những itemset có lợi ích thấp. Chứng minh điều này bằng tính chất 3, nếu 1 itemset X chứa một itemset Y khác sao cho TWU(Y) < minutil, khi ấy X và các tập cha của nó là các itemset có lợi ích thấp.

CHƯƠNG 3
THỰC NGHIỆM – ĐÁNH GIÁ KẾT QUẢ

Để đo tính hiệu quả của các thuật toán HUI-Miner[7] và FHM[13], tôi tiến hành thực nghiệm trên 2 bộ dữ liệu tổng hợp Chess_utility gồm 20 item và bộ Retail 46 item.

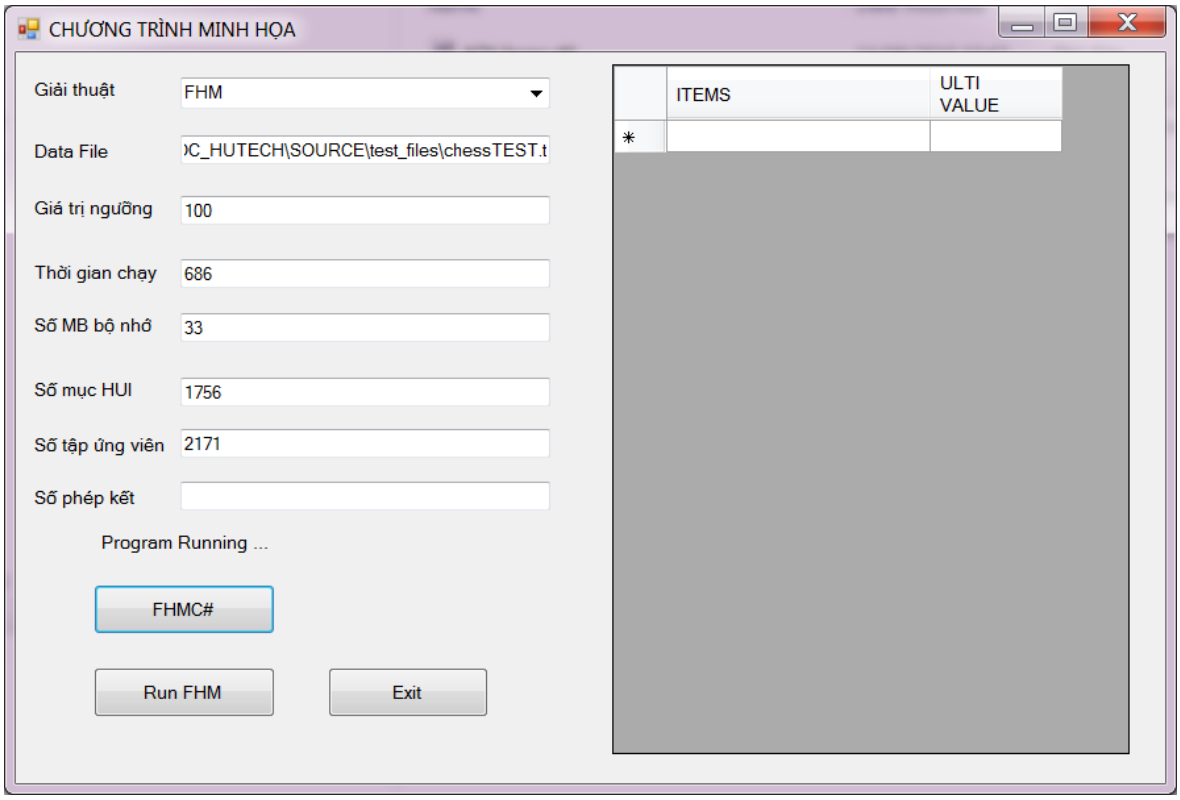
Thử nghiệm chạy trên máy tính: Sony 1.6 GHz bộ xử lý Core i7 và 6 GB bộ nhớ chính, chạy trên hệ điều hành Windows 7.

Chương trình lập trình được thử nghiệm trên ngôn ngữ lập trình C# trong bộ Microsoft Visual Studio 2010.

Đơn vị tính của kết quả thử nghiệm:

+ Thời gian : tính bằng giây (s)

+ Bộ nhớ : tính bằng megabytes (MB)



Hình 3.1: giao diện chương trình minh họa thuật giải HUI-MINER[7] và FHM[13]

3.1 Bộ dữ liệu

Bảng 3.1: liệt kê các đặc tính của 2 bộ dữ liệu thử nghiệm.

Bộ dữ liệu	Số giao dịch	Số mục
Retail	88162	46
Bộ Chess_utility	3196	20

3.2 Kết quả thử nghiệm

Kết quả thử nghiệm được lấy trung bình từ kết quả của 3 lần chạy khác nhau trên cùng bộ dữ liệu và giá trị ngưỡng minulti.

3.2.1 Kết quả thực nghiệm trên bộ dữ liệu Chess_utility

Bảng 3.2: kết quả thực nghiệm trên bộ Chess_utility

	MinUtility = 100		MinUtility = 500		MinUtility = 1000	
	Thời gian	Bộ Nhớ	Thời gian	Bộ Nhớ	Thời gian	Bộ Nhớ
FHM	840	537	754	513	793	258
Hui-Miner	977	61	859	104	948	99

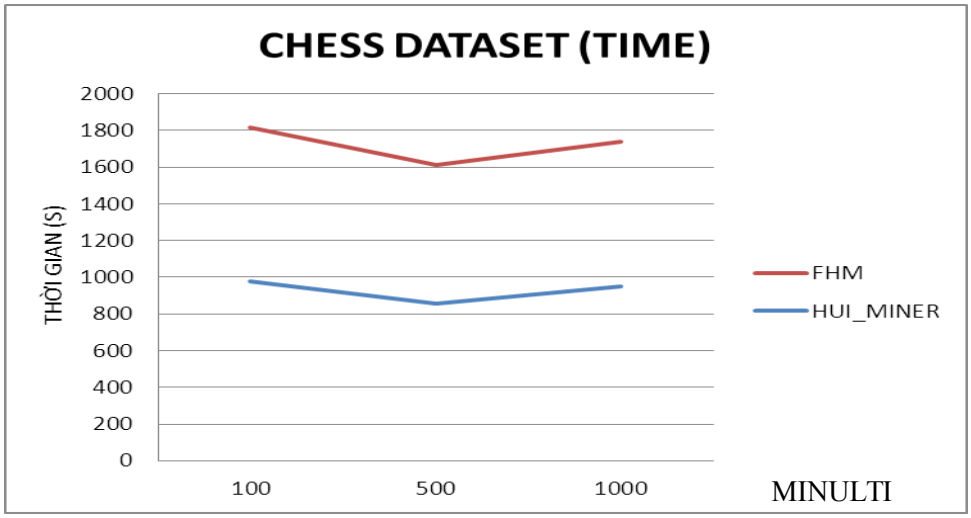
3.2.4 Kết quả thực nghiệm trên bộ dữ liệu Retail

Bảng 3.3: kết quả thực nghiệm trên bộ Retail

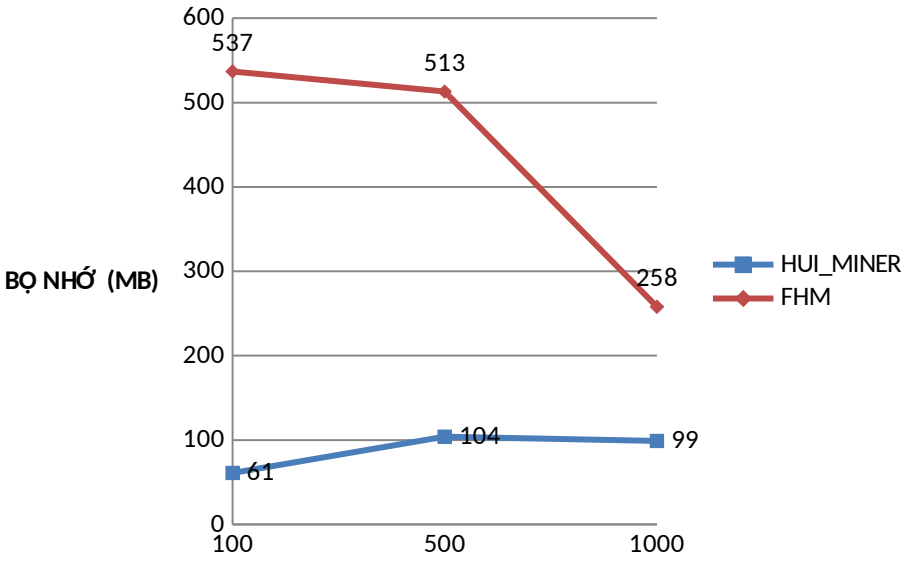
MinUlty	5.000		10.000		15.000		20.000		25.000		30.000	
	Thời gian	Bộ nhớ	Thời gian	Bộ Nhớ	Thời gian	Bộ nhớ	Thời gian	Bộ nhớ	Thời gian	Bộ Nhớ	Thời gian	Bộ nhớ
FHM	10.087	46	6.789	209	4.924	239	3.592	50	2.820	47	2.287	47
Hui-Miner	15.107	29	7.277	16	4.993	151	3.714	37	2.837	27	2.300	36

3.3 Biểu đồ so sánh

3.3.1 Trên dataset CHESS

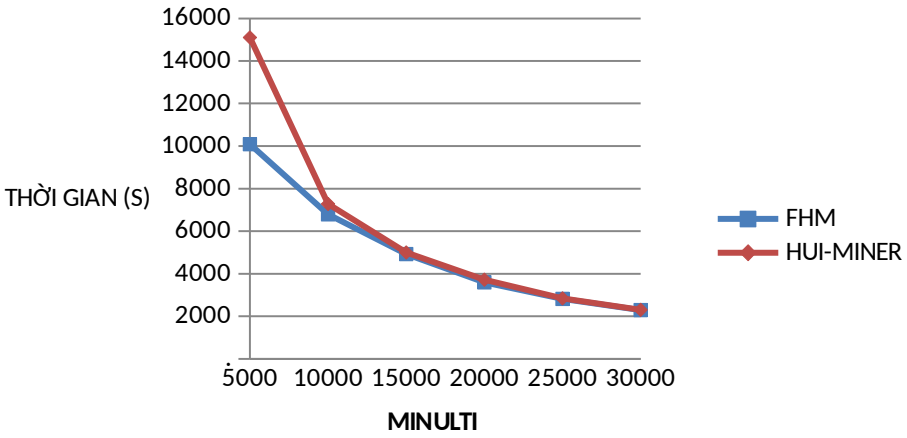


Hình 3.2 Biểu đồ so sánh về thời gian thực thi thuật toán trên bộ dữ liệu Chess_utility

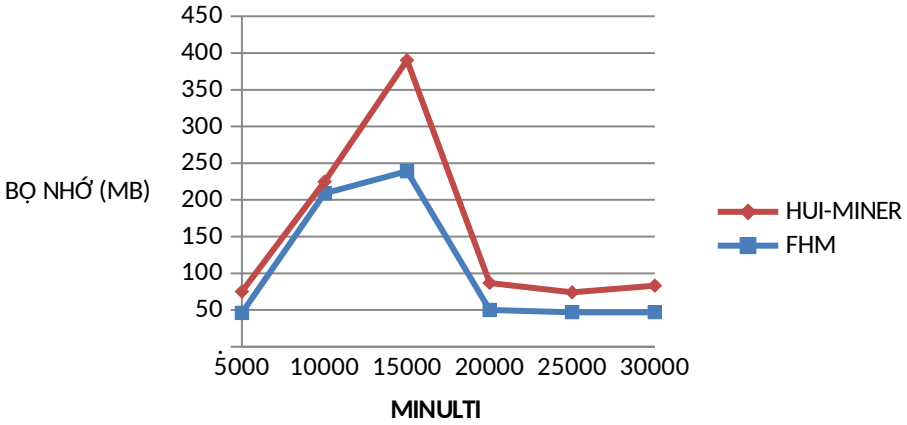


Hình 3.3 Biểu đồ so sánh về bộ nhớ trên bộ dữ liệu Chess_utility

3.3.2 Trên dataset RETAIL



Hình 3.4 Biểu đồ so sánh về thời gian thực thi thuật toán trên bộ dữ liệu Retail



Hình 3.5 Biểu đồ so sánh về bộ nhớ trên bộ dữ liệu Retail

3.4 Đánh giá

* Trên bộ dữ liệu Retail:

Thời gian thực thi trung bình của thuật toán FHM[13] nhanh hơn thời gian thực thi của thuật toán HUI-Miner[7] khoảng 19%. Tuy nhiên thuật toán FHM[13] chiếm bộ nhớ gấp đôi so với thuật toán HUI-Miner[7].

* Trên bộ dữ liệu Chess:

Thời gian thực thi trung bình của thuật toán FHM[13] nhanh hơn thời gian thực thi của thuật toán HUI-Miner[7] khoảng 17%.

Thời gian thực thi

Luận văn thực hiện thử nghiệm hai thuật toán FHM[13] và HUI-Miner[7] trên mỗi bộ dữ liệu trong khi giảm ngưỡng minutil cho đến khi các thuật toán trở nên quá tải, thoát ra khỏi bộ nhớ hoặc quan sát rõ ràng thuật toán nào chạy xuất ra được kết quả.

- Trên bộ dữ liệu Retail:

Thời gian thực thi trung bình của thuật toán FHM[13] nhanh hơn thời gian thực thi của thuật toán HUI-Miner[7] khoảng 19%.

- Trên bộ dữ liệu Chess:

Thời gian thực thi trung bình của thuật toán FHM[13] nhanh hơn thời gian thực thi của thuật toán HUI-Miner[7] khoảng 17%.

CHƯƠNG 4
KẾT LUẬN

4.1. Những kết quả chính của luận văn

Luận văn đã nghiên cứu về các cách tiếp cận khác nhau trong bài toán tìm itemset tiện ích cao. Luận văn đã khái quát vấn đề về khai thác itemset lợi ích cao, trình bày những khái niệm cơ bản và các cách tiếp cận để khai thác itemset lợi ích cao. Đồng thời trình bày chi tiết hai thuật giải điển hình dựa trên tính chất TWDCP và cấu trúc EUCS: thuật toán HUI-Miner[7] và FHM[13] cũng như một số thuật toán khai thác theo mô hình hai giai đoạn. Các thuật toán được minh họa qua ví dụ cụ thể và có nhận xét về tính hiệu quả. Phần thực nghiệm, luận văn đã xây dựng được chương trình đánh giá kết quả thực thi của hai thuật giải HUI-Miner[7] và FHM[13]. Nhìn chung thuật toán FHM[13] có thời gian xử lý cải thiện khoảng 17% so với thuật toán trước đó là HUI-Miner[7].

4.2. Hướng nghiên cứu tiếp theo

Trên cơ sở nghiên cứu đã được trình bày trong luận văn, học viên sẽ tiếp tục nghiên cứu sâu hơn các thuật toán khai thác itemset lợi ích cao chỉ sử dụng một giai đoạn xử lý, đặc biệt là cải tiến về thời gian xử lý và số các tập ứng viên sinh ra trong quá trình tìm tập lợi ích cao nhằm nâng cao hiệu quả của các thuật toán để áp dụng vào một số bài toán khai thác dữ liệu đang được áp dụng trong nhiều lĩnh vực, đặc biệt trong lĩnh vực kinh doanh.

TÀI LIỆU THAM KHẢO

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in largedatabases. In: Proc. Int. Conf. Very Large Databases, pp. 487-499, (1994)
2. Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., Lee, Y.-K.: Effcient Tree Structures for High-utility Pattern Mining in Incremental Databases. In: IEEE Trans. Knowl.Data Eng. 21(12), pp. 1708-1721 (2009)
3. Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast Vertical Sequential Pattern Mining Using Co-occurrence Information. In: Proc. 18th Pacific-AsiaConference on Knowledge Discovery and Data Mining, Springer, LNAI, (2014)
4. Fournier-Viger, P., Wu, C.-W., Gomariz, A., Tseng, V. S.: VMSP: Effient VerticalMining of Maximal Sequential Patterns. In: Proc. 27th Canadian Conference onArtificial Intelligence, Springer, LNAI, pp. 83-94 (2014)
5. Fournier-Viger, P., Nkambou, R., Tseng, V. S.: RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth. In: Proc. ACM 26th Symposium on Applied Computing, pp. 954- 959 (2011)
6. Li, Y.-C., Yeh, J.-S., Chang, C.-C.: Isolated items discarding strategy for discovering high utility itemsets. In: Data & Knowledge Engineering. 64(1), pp. 198-217 (2008)
7. Liu, M., Qu, J.:Mining High Utility Itemsets without Candidate Generation. In Proceedings of CIKM12, pp. 55-64 (2012)
8. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Proc. PAKDD 2005, pp. 689-695 (2005)
9. Shie, B.-E., Cheng, J.-H., Chuang, K.-T., Tseng, V. S.: A One-Phase Method for Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments.In: Proceedings of IEA/AIE12, pp. 616-626 (2012)

10. Tseng, V. S., Shie, B.-E., Wu, C.-W., Yu., P. S.: Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases. In: IEEE Trans. Knowl. DataEng. 25(8), pp. 1772-1786 (2013)

11. Yin, J., Zheng, Z., Cao, L.: USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. In: Proceedings of ACM SIG KDD12, pp. 660-668 (2012)

12. Bay Vo, Huy Nguyen, Bac Le: Mining High Utility Itemsets from Vertical Distributed Databases. In: Computing and Communication Technologies, RIVF '09. International Conference (2009)

13. Philippe Fournier-Viger1, Cheng-Wei Wu2, Souleymane Zida1, Vincent S: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning In: Volume 8502 of the series Lecture Notes in Computer Science, pp 83-92 (2014)