Khảo sát về khai thác Itemset tiện ích cao

Philippe Fournier-Viger, Jerry Chun-Wei Lin, Tín Trường Chi, Roger Nkambou

Tóm tắt Khai thác mẫu tiện ích cao là một nhiệm vụ khoa học dữ liệu mới nổi, bao gồm việc khám phá các mẫu có tầm quan trọng cao trong cơ sở dữ liệu. Tiện ích của một mô hình có thể được đo lường theo các tiêu chí khách quan khác nhau như lợi nhuận, tần suất và trọng số của nó. Trong số các loại mẫu hữu ích cao có thể được tìm thấy trong cơ sở dữ liệu, tập mục hữu ích cao được nghiên cứu nhiều nhất. Tập mục hữu ích cao là tập hợp các giá trị xuất hiện trong cơ sở dữ liệu và có tầm quan trọng cao đối với người dùng, được đo bằng hàm tiện ích. Khai thác tập mục có tiện ích cao khái quát hóa vấn đề khai thác tập mục thường xuyên bằng cách xem xét số lượng và trọng lượng của mục. Một ứng dụng phổ biến của khai thác tập mục có tính tiện ích cao là khám phá tất cả các tập hợp các mục được khách hàng mua cùng nhau mang lại lợi nhuận cao. Chương này giới thiệu về khai thác tập mục có tính tiện ích cao, xem xét các thuật toán tiên tiến, các phần mở rộng, ứng dụng của chúng và thảo luận về các cơ hội nghiên cứu. Chương này nhắm đến cả những người mới tham gia vào lĩnh vực khai thác tập mục có tiện ích cao cũng như các nhà nghiên cứu làm việc trong lĩnh vực này.

1 Giới thiệu

Mục tiêu của khai thác dữ liệu là trích xuất các mẫu hoặc đào tạo mô hình từ cơ sở dữ liệu để hiểu về quá khứ hoặc dự đoán tương lai. Các loại thuật toán khai thác dữ liệu

Viện Công nghệ Philippe Fournier-Viger Harbin (Thâm Quyến), Thâm Quyến, Trung Quốc, e-mail: philfv8@yahoo.com Jerry Chun-Wei Lin

Khoa Máy tính, Toán và Vật lý, Đại học Khoa học Ứng dụng Tây Na Uy (HVL), Bergen, Na Uy e-mail: jerrylin@ieee.org

Đại học Tín Trường Chi Đà Lạt, Đà Lạt, Việt Nam, e-mail: tintc@dlu.edu.vn Roger Nkambou

Đại học Quebec ở Montreal, e-mail: nkambou.roger@uqam.ca

đã được đề xuất để phân tích dữ liệu [1, 38]. Một số thuật toán tạo ra các mô hình hoạt động như hộp đen. Ví dụ, một số loại mạng lưới thần kinh được thiết kế để thực hiện các dự đoán rất chính xác nhưng con người không thể dễ dàng giải thích được. Để trích xuất kiến thức từ dữ liệu mà con người có thể hiểu được, các thuật toán khai thác mẫu được thiết kế [27, 28]. Mục tiêu là khám phá các mẫu dữ liệu thú vị, hữu ích và/hoặc bất ngờ. Ưu điểm của việc khai thác mẫu so với một số phương pháp khai thác dữ liệu khác là việc khám phá các mẫu là một kiểu học không giám sát vì nó không yêu cầu dữ liệu được dán nhãn. Các mẫu có thể được trích xuất trực tiếp từ dữ liệu thô và sau đó được sử dụng để hiểu dữ liệu và hỗ trợ việc ra quyết định. Các thuật toán khai thác mẫu đã được thiết kế để trích xuất các loại mẫu khác nhau, mỗi loại cung cấp thông tin khác nhau cho người dùng và để trích xuất các mẫu từ các loại dữ liệu khác nhau. Các loại mẫu phổ biến là các mẫu tuần tự [27], tập mục [28], cụm, xu hướng, ngoại lệ và cấu trúc biểu đồ [38].

Nghiên cứu về các thuật toán khai thác mẫu đã bắt đầu từ những năm 1990 với các thuật toán khám phá các mẫu phổ biến trong cơ sở dữ liệu [2]. Thuật toán đầu tiên để khai thác mẫu thường xuyên là Apriori [2]. Nó được thiết kế để khám phá các tập mục thường xuyên trong cơ sở dữ liệu giao dịch của khách hàng. Cơ sở dữ liệu giao dịch là một tập hợp các bản ghi (giao dịch) cho biết các mặt hàng được khách hàng mua tại các thời điểm khác nhau. Tập mục thường xuyên là một nhóm các giá trị (mục) được khách hàng mua thường xuyên (xuất hiện trong nhiều giao dịch) của cơ sở dữ liệu giao dịch. Ví dụ, một tập mục thường xuyên trong cơ sở dữ liệu có thể là nhiều khách hàng mua món mì với món nước sốt cay. Những mô hình như vậy dễ hiểu đối với con người và có thể được sử dụng để hỗ trợ việc ra quyết định. Ví dụ: mẫu {mì, sốt cay} có thể được sử dụng để đưa ra các quyết định tiếp thị, chẳng hạn như đồng quảng cáo mì với sốt cay. Việc khám phá các tập phổ biến là một nhiệm vụ khai thác dữ liệu được nghiên cứu kỹ lưỡng và có ứng dụng trong nhiều lĩnh vực. Có thể xem đây là nhiệm vụ chung của việc phân tích cơ sở dữ liệu để tìm các giá trị (mục) cùng xảy ra trong một tập hợp các bản ghi cơ sở dữ liệu (giao dịch) [10, 16, 20, 37, 61, 64, 65, 66].

Mặc dù việc khai thác mẫu thường xuyên là hữu ích nhưng nó dựa trên giả định rằng các mẫu thường xuyên là thú vị. Nhưng giả định này không đúng cho

nhiều ứng dụng. Ví dụ: trong cơ sở dữ liệu giao dịch, mẫu {sữa, bánh mì} có thể rất thường xuyên nhưng có thể không thú vị vì nó thể hiện hành vi mua hàng phổ biến và có thể mang lại lợi nhuận thấp. Mặt khác, một số mô hình như {caviar, sâm panh} có thể không thường xuyên nhưng có thể mang lại lợi nhuận cao hơn. Do đó, để tìm ra các mẫu thú vị trong dữ liệu, có thể xem xét các khía cạnh khác như lợi nhuận hoặc tiện ích.

Để giải quyết hạn chế này của việc khai thác tập mục thường xuyên, một lĩnh vực nghiên cứu mới nổi là khám phá các mẫu tiện ích cao trong cơ sở dữ liệu [31, 52, 56, 58, 59, 83, 87, 94]. Mục tiêu của việc khai thác tiện ích là khám phá các mẫu có tiện ích cao (có tầm quan trọng cao đối với người dùng), trong đó tiện ích của mẫu được thể hiện dưới dạng hàm tiện ích. Chức năng tiện ích có thể được xác định theo các tiêu chí như lợi nhuận được tạo ra khi bán một mặt hàng hoặc thời gian dành cho các trang web. Nhiều loại mô hình tiện ích cao đã được nghiên cứu. Chương này khảo sát nghiên cứu về loại phổ biến nhất, đó là các tập mục có tính tiện ích cao [83]. Khai thác các tập mục có tiện ích cao có thể được coi là sự tổng quát hóa của vấn đề khai thác tập mục thường xuyên trong đó đầu vào là cơ sở dữ liệu giao dịch trong đó mỗi mục có trọng số biểu thị tầm quan trọng của nó và

nơi các mặt hàng có thể có số lượng không nhị phân trong giao dịch. Việc xây dựng bài toán tổng quát này cho phép mô hình hóa các nhiệm vụ khác nhau như khám phá tất cả các tập mục (bộ mục) mang lại lợi nhuận cao trong cơ sở dữ liệu giao dịch, tìm các tập hợp trang web mà người dùng dành nhiều thời gian hoặc tìm tất cả các mẫu thường xuyên như trong truyền thống. khai thác mẫu thường xuyên. Khai thác tập mục có tiện ích cao là một lĩnh vực nghiên cứu rất tích cực. Chương này cung cấp một bản khảo sát toàn diện về lĩnh vực này, vừa là phần giới thiệu vừa là hướng dẫn về những tiến bộ gần đây và các cơ hội nghiên cứu.

Phần còn lại của chương này được tổ chức như sau. Phần 2 giới thiệu vấn đề khai thác tập mục có tính tiện ích cao, các thuộc tính chính của nó và cách khái quát hóa việc khai thác tập mục thường xuyên. Phần 3 khảo sát các kỹ thuật phổ biến để khám phá hiệu quả các tập mục hữu ích cao trong cơ sở dữ liệu. Phần 4 trình bày các phần mở rộng chính của việc khai thác tập mục có tính tiện ích cao. Phần 5 thảo luận về các cơ hội nghiên cứu. Phần 6 trình bày các triển khai nguồn mở. Cuối cùng, phần 7 đưa ra kết luận.

2 Định nghĩa vấn đề

Phần này trước tiên giới thiệu vấn đề khai thác tập mục thường xuyên [2], sau đó giải thích cách nó được khái quát hóa như khai thác tập mục có tính tiện ích cao [31, 52, 56, 58, 59, 83, 87, 94]. Sau đó, các thuộc tính chính của bài toán khai thác tập mục có tính tiện ích cao được trình bày và đối chiếu với các đặc tính của việc khai thác tập mục thường xuyên.

2.1 Khai thác tập mục thường xuyên

Vấn đề khai thác tập mục thường xuyên bao gồm việc trích xuất các mẫu từ cơ sở dữ liệu giao dịch. Trong cơ sở dữ liệu giao dịch, mỗi bản ghi (được gọi là giao dịch) là một tập hợp các mục (ký hiệu). Về mặt hình thức, cơ sở dữ liệu giao dịch D được định

nghĩa như sau. Giả sử có tập I của tất cả các mục (ký hiệu) I = {i1, i2, . . . , im } xảy ra trong cơ sở dữ liệu. Cơ sở dữ liệu giao dịch D là một tập hợp các bản ghi, gọi là giao dịch, ký hiệu là D = {T0, T1, . . . , Tn }, trong đó mỗi giao dịch Tq là một tập hợp các mục (tức là Tq ⊆ I) và có một mã định danh duy nhất q được gọi là TID (Mã định danh giao dịch) của nó. Ví dụ: hãy xem xét cơ sở dữ liệu giao dịch khách hàng được hiển thị trong Bảng 1. Cơ sở dữ liệu trong Bảng 3 chứa năm giao dịch được ký hiệu là T0, T1, T3 và T4. Giao dịch T2 chỉ ra rằng các mặt hàng a, c và d đã được khách hàng mua cùng nhau trong giao dịch đó.

Mục tiêu của việc khai thác tập mục thường xuyên là khám phá các tập mục (bộ mục) có độ hỗ trợ cao (xuất hiện thường xuyên). Về mặt hình thức, tập mục X là tập hữu hạn các mục sao cho X ⊆ I. Đặt ký hiệu |X | biểu thị số phần tử của tập hợp hay nói cách khác là số phần tử trong một tập mục X. Một tập mục X được gọi là có độ dài k hoặc một tập mục k nếu nó chứa k mục (|X | = k). Ví dụ: {a, b, c} là tập có 3 mục và {a, b} là tập có 2 mục. Biện pháp hỗ trợ được xác định như sau.

Bảng 1: Cơ sở dữ liệu giao dịch

Giao dịch TID	T0
a, b, c, d, e	
T1	b, c, d, e
T2	a, c, d
T3	a, c, e
T4	b, c, e

Định nghĩa 1 (Biện pháp hỗ trợ). Độ hỗ trợ (tần số) của tập mục X trong cơ sở dữ liệu giao dịch D được ký hiệu là sup(X) và được định nghĩa là sup(X) = |{T |X ⊆ T ∧ T ∈ D}|, đó là số lượng giao dịch chứa X.

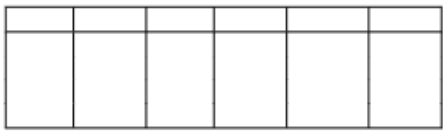
Ví dụ, độ hỗ trợ của tập mục {a, c} trong cơ sở dữ liệu của Bảng 3 là 3, vì tập mục này xuất hiện trong ba giao dịch (T0, T2 và T3). Định nghĩa về thước đo hỗ trợ này được gọi là mức hỗ trợ tương đối. Một định nghĩa tương đương khác là thể hiện mức hỗ trợ dưới dạng phần trăm trên tổng số giao dịch (được gọi là mức hỗ trợ tuyệt đối). Ví dụ: độ hỗ trợ tuyệt đối của {a,c} là 60% vì nó xuất hiện ở 3 trên 5 giao dịch. Bài toán khai thác tập mục thường xuyên được định nghĩa như sau: Định nghĩa 2 (Tập mục thường xuyên). Giả sử có một ngưỡng minsup > 0, do người dùng xác định. Một tập mục X là tập phổ biến nếu độ hỗ trợ sup(X) của nó không nhỏ hơn ngưỡng tối thiểu đó (tức là sup(X) ≥ minsup). Ngược lại, X là tập mục không phổ biến. Định nghĩa 3 (Định nghĩa vấn đề). Vấn đề của việc khai phá tập phổ biến là khám phá tất cả các tập mục phổ biến trong cơ sở dữ liệu giao dịch D, với ngưỡng tối thiểu do người dùng đặt.

Ví dụ, hãy xem xét cơ sở dữ liệu của Bảng 3 và minsup = 3. Có 11 tập phổ biến được liệt kê trong Bảng 2.

Bảng 2: Tập phổ biến cho minsup = 3

Hỗ trợ tập mục		Hỗ trợ tập mục		Hỗ trợ tập mục	
{Một }	3	{e }	4	{là }	3
{b }	3	{a, c }	3	{c, e }	4
{c }	5	{b, c }	3	{b, c, e }	3
{d }	3	{đĩa CD }	3		

Vấn đề khai thác tập mục thường xuyên đã được nghiên cứu trong hơn hai thập kỷ. Nhiều thuật toán đã được đề xuất để khám phá các mẫu phổ biến một cách hiệu quả, bao gồm Apriori [2], FP-Growth [39], Eclat [91], LCM [81] và H-Mine [69]. Mặc dù việc khai thác tập mục thường xuyên có nhiều ứng dụng, giả định chắc chắn về việc khai thác tập mục thường xuyên là các mẫu phổ biến hữu ích hoặc thú vị đối với người dùng, điều này không phải lúc nào cũng đúng. Để giải quyết hạn chế quan trọng này của khai thác mẫu thường xuyên truyền thống, nó đã được khái quát hóa là khai thác tập mục có tính tiện ích cao, trong đó các mục được chú thích bằng các giá trị số và các mẫu được chọn dựa trên hàm tiện ích do người dùng xác định.



2.2 Khai thác tập mục tiện ích cao

Nhiệm vụ khai thác tập mục có tiện ích cao [31, 52, 56, 58, 59, 87, 94] bao gồm khám phá các mẫu trong một loại cơ sở dữ liệu giao dịch tổng quát được gọi là cơ sở dữ liệu giao dịch định lượng, nơi cung cấp thông tin bổ sung, đó là số lượng các mục trong các giao dịch và trọng số cho thấy tầm quan trọng tương đối của từng mục đối với người dùng.

Về mặt hình thức, cơ sở dữ liệu giao dịch định lượng D được định nghĩa như sau. Giả sử có tập I của tất cả các mục I = {i1, i2, . . . Tôi }. Cơ sở dữ liệu giao dịch định lượng D là tập hợp các giao dịch, ký hiệu là D = {T0, T1, . . . , Tn }, trong đó mỗi giao dịch Tq là một tập hợp các mục (tức là Tq ⊆ I) và có một mã định danh duy nhất q được gọi là TID (Mã định danh giao dịch) của nó. Mỗi mục i ∈ I được liên kết với một số dương p(i), được gọi là tiện ích bên ngoài của nó. Tiện ích bên ngoài của một vật phẩm là một con số dương thể hiện tầm quan trọng tương đối của nó đối với người dùng. Hơn nữa, mọi mục i xuất hiện trong giao dịch Tc đều có số dương q(i,Tc), được gọi là tiện ích nội tại của nó, đại diện cho số lượng i trong giao dịch Tc.

Để minh họa các định nghĩa này, hãy xem xét một cơ sở dữ liệu giao dịch khách hàng mẫu được mô tả trong Bảng 3, sẽ được sử dụng làm ví dụ chạy. Trong ví dụ này, tập hợp các mục là I = {a, b, c, d, e}. Nó có thể được coi là đại diện cho các sản phẩm khác nhau được bán trong một cửa hàng bán lẻ như táo, ngũ cốc, thịt vịt và trứng. Cơ sở dữ liệu trong Bảng 3 chứa năm giao dịch (T0, T1, . . T4). Giao dịch T3 chỉ ra rằng các mặt hàng a, c và e được mua với số lượng mua (hữu ích bên trong) lần lượt là 2, 6 và 2. Bảng 4 cung cấp tiện ích bên ngoài của các mặt hàng, thể hiện lợi nhuận đơn vị của chúng. Giả sử rằng đồng đô la ($) được sử dụng làm tiền tệ. Việc bán một đơn vị mặt hàng a, b, c, d và e mang lại lợi nhuận lần lượt là 5$, 2$, 1$, 2$ và 3$.

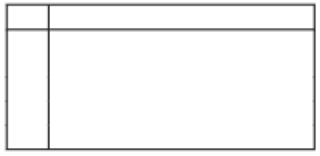
Bảng 3: Cơ sở dữ liệu giao dịch định lượng

Giao dịch TID	
T0	(a, 1), (b, 5), (c,

1), (d, 3), (e, 1)
T1 (b, 4), (c, 3), (d, 3), (e, 1)
T2 (a, 1), (c, 1), (d, 1)
T3 (a, 2), (c, 6), (e, 2)
T4 (b, 2), (c, 2), (e, 1)

Bảng 4: Giá trị tiện ích bên ngoài

Mục	Tiện ích bên ngoài
Một	5
b	2
c	1
d	2
e	3



Mục tiêu của việc khai thác tập mục có tính tiện ích cao là khám phá các tập mục (bộ mục) xuất hiện trong cơ sở dữ liệu định lượng và có tính hữu dụng cao (ví dụ: mang lại lợi nhuận cao). Tiện ích của một tập mục là thước đo tầm quan trọng của nó trong cơ sở dữ liệu, được tính toán bằng hàm tiện ích. Biện pháp hữu ích thường được xác định theo định nghĩa sau, mặc dù các biện pháp thay thế đã được đề xuất [83] (sẽ được xem xét trong Phần 4). Trong ví dụ đang chạy, thước đo hữu ích được hiểu là số lợi nhuận được tạo ra bởi mỗi bộ mặt hàng.

Định nghĩa 4 (Thước đo hữu ích). Tiện ích của mục i trong giao dịch Tc được ký hiệu là u(i, Tc) và được định nghĩa là p(i) × q(i, Tc). Trong bối cảnh phân tích các giao dịch của khách hàng, nó thể hiện lợi nhuận được tạo ra từ việc bán mặt hàng i trong giao dịch Tc. Tiện ích của tập mục X trong giao dịch Tc được ký hiệu là u(X, Tc) và được định nghĩa là u(X, Tc) = Ði ∈X u(i, Tc) nếu X ⊆ Tc . Ngược lại u(X, Tc) = 0. Tiện ích của tập mục X trong cơ sở dữ liệu D được ký hiệu là u(X) và được định nghĩa là u(X) = ÕTc ∈g(X) u(X, Tc), trong đó g (X) là tập hợp các giao dịch chứa X. Nó thể hiện lợi nhuận được tạo ra từ việc bán tập mục X trong cơ sở dữ liệu.

Ví dụ, tiện ích của mục a trong giao dịch T2 là u(a, T2) = 5 × 2 = 10. Tiện ích của tập mục {a, c} trong T2 là u({a, c}, T2) = u (a, T2)+u(c, T2) = 5×2+1×6 = 16. Tiện ích của tập mục {a, c} trong cơ sở dữ liệu là u({a, c}) = u(a) + u(c) = u(a, T0) + u(a, T2) + u(a, T3) + u(c, T0) + u(c, T2) + u(c, T3) = 5 + 5 + 10 + 1 + 1 + 6 = 28. Như vậy, hữu dụng của {a, c} trong cơ sở dữ liệu có thể hiểu là tổng lợi nhuận mà các mặt hàng a và c tạo ra khi chúng được mua cùng nhau. Bài toán khai phá tập mục có ích lợi cao được định nghĩa như sau:

Định nghĩa 5 (Tập mục có tính tiện ích cao). Một tập mục X là một tập mục có ích cao nếu tiện ích u(X) của nó không nhỏ hơn ngưỡng tiện ích tối thiểu do người dùng chỉ định tối thiểu do người dùng đặt ra (tức là u(X) ≥ minutil). Ngược lại, X là tập mục có lợi ích thấp.

Định nghĩa 6 (Định nghĩa vấn đề). Vấn đề của việc khai thác tập mục hữu ích cao là khám phá tất cả các tập mục hữu ích cao, đưa ra một ngưỡng tối thiểu do người dùng đặt ra [83].

Lưu ý rằng trong một số nghiên cứu, tiện ích của một tập mục được biểu thị bằng phần trăm của tổng tiện ích trong cơ sở dữ liệu. Việc khám phá các mẫu sử dụng định nghĩa này được gọi là tiện ích tuyệt đối [79], tương đương với việc sử dụng định nghĩa trên và dẫn đến việc tìm ra cùng một tập hợp các mẫu.

Khai thác tập mục có tiện ích cao có nhiều ứng dụng. Đối với ứng dụng phân tích giỏ thị trường, vấn đề khai thác tập mục có tính tiện ích cao có thể được hiểu là tìm tất cả các tập mục đã tạo ra lợi nhuận lớn hơn hoặc bằng minutil. Ví dụ: đối với ví dụ đang chạy, nếu minutil = 25, tập hợp HUI được hiển thị trong Bảng 5. Một số thuật toán đã được đề xuất để khám phá các tập mục có tiện ích cao (được xem xét trong phần tiếp theo).

Điều thú vị cần lưu ý là vì vấn đề khai thác tập mục có ích cao là tổng quát hơn vấn đề khai thác tập mục thường xuyên, nên bất kỳ thuật toán nào để khám phá các tập mục có ích cao cũng có thể được sử dụng để khám phá các tập mục thường xuyên trong cơ sở dữ liệu giao dịch. Để làm được điều đó, có thể áp dụng các bước sau:

Bảng 5: Các tập mục có tính tiện ích cao cho minutil = 25

Bộ vật phẩm	Tính thiết thực	Tập mục	Tiện ích	Bộ mục tiện ích	
{a, c }	28	{b, c, d }	34	{b, d, e }	36
{a, c, e }	31	{b, c, d, e }	40	{là }	31
{a,b,c,d,e }	25	{b, c, e }	37	{c, e }	27
{b, c }	28	{b, d }	30		

1. Cơ sở dữ liệu giao dịch được chuyển đổi thành cơ sở dữ liệu giao dịch định lượng. Với mỗi mục i ∈ I, giá trị tiện ích bên ngoài của i được đặt thành 1, tức là p(i) = 1 (để biểu thị rằng tất cả các mục đều quan trọng như nhau). Hơn nữa, với mỗi mục i và giao dịch Tc , nếu i ∈ Tc , đặt q(i, Tc) = 1. Ngược lại, đặt q(i, Tc) = 0.

2. Sau đó, thuật toán khai thác tiện ích cao được áp dụng trên cơ sở dữ liệu giao dịch định lượng kết quả với minutil được đặt thành minsup, để thu được các tập phổ biến.

Ví dụ, cơ sở dữ liệu của Bảng 1 có thể được chuyển đổi thành cơ sở dữ liệu định lượng. Kết quả là cơ sở dữ liệu giao dịch của Bảng 6 và 7. Sau đó, các tập mục phổ biến có thể được khai thác từ cơ sở dữ liệu này bằng thuật toán khai thác tập mục có tính tiện ích cao. Tuy nhiên, mặc dù thuật toán khai thác tập mục có tính tiện ích cao có thể được sử dụng để khai thác các tập mục thường xuyên, nhưng có thể nên sử dụng thuật toán khai thác tập mục thường xuyên khi hiệu suất là quan trọng vì những thuật toán này được tối ưu hóa cho nhiệm vụ này.

Bảng 7: Các giá trị tiện ích bên ngoài cho cơ sở dữ liệu Bảng 6

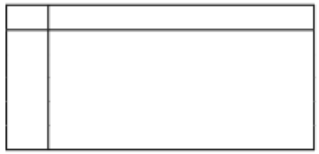
Giao dịch TID	
T0	(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)
T1	(b, 1), (c, 1), (d, 1), (e, 1)
T2	(a, 1), (c, 1), (d, 1)
T3	(a, 1), (c, 1), (e, 1)
T4	(b, 1), (c, 1), (e, 1)

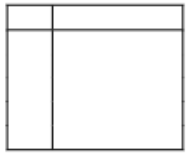
Mục	Tiện ích bên ngoài
Một	1
b	1
c	1
d	1
e	1

2.3 Các đặc tính chính của bài toán khai thác tập mục có tính tiện ích cao

Đối với một cơ sở dữ liệu định lượng và ngưỡng tiện ích tối thiểu nhất định, bài toán khai thác tập mục có tiện ích cao luôn có một giải pháp duy nhất. Đó là liệt kê tất cả các mẫu có tiện ích lớn hơn hoặc bằng ngưỡng tiện ích tối thiểu do người dùng chỉ định.

Vấn đề khai thác tập mục có tiện ích cao là khó khăn vì hai lý do chính. Lý do đầu tiên là số lượng tập mục cần xem xét có thể rất lớn để tìm ra những tập mục có tính hữu ích cao. Nói chung, nếu một cơ sở dữ liệu chứa m mục riêng biệt thì có 2m − 1 tập mục có thể có (không bao gồm tập trống). Ví dụ: nếu I = {a, b, c} thì các tập mục có thể là {a}, {b}, {c}, {a, b}, {a, c}, {b, c} và { a, b, c}. Như vậy, có





là 23 − 1 = 7 tập mục, có thể được hình thành với I = {a, b, c}. Một cách tiếp cận đơn giản để giải quyết vấn đề khai thác tập mục có tiện ích cao là đếm tiện ích của tất cả các tập mục có thể có bằng cách quét cơ sở dữ liệu, sau đó giữ lại các tập mục có ích cao. Mặc dù cách tiếp cận này mang lại kết quả chính xác nhưng nó không hiệu quả. Lý do là số lượng các tập mục có thể có có thể rất lớn. Ví dụ: nếu một cửa hàng bán lẻ có 10.000 mặt hàng trên kệ (m = 10.000), thì nên tính toán tiện ích của 210.000 − 1 tập mặt hàng có thể có, điều này khó quản lý được khi sử dụng cách tiếp cận đơn giản. Cần lưu ý rằng vấn đề khai thác tập mục có tiện ích cao có thể rất khó khăn ngay cả đối với cơ sở dữ liệu nhỏ. Ví dụ: một cơ sở dữ liệu chứa một giao dịch gồm 100 mục có thể tạo ra 2100 − 1 tập mục có thể. Do đó, kích thước của không gian tìm kiếm (số lượng tập mục có thể có) có thể rất lớn ngay cả khi có ít giao dịch trong cơ sở dữ liệu. Trên thực tế, kích thước của không gian tìm kiếm không chỉ phụ thuộc vào kích thước của cơ sở dữ liệu mà còn phụ thuộc vào mức độ tương tự của các giao dịch trong cơ sở dữ liệu, giá trị tiện ích lớn đến mức nào và cũng phụ thuộc vào mức độ thấp của ngưỡng minutil được đặt bởi người dùng.

Lý do thứ hai khiến vấn đề khai thác tập mục hữu ích cao gặp khó khăn là vì các tập mục hữu ích cao thường nằm rải rác trong không gian tìm kiếm. Vì vậy, nhiều tập mục phải được xem xét bằng một thuật toán trước khi nó có thể tìm ra các tập mục có tính tiện ích cao thực tế. Để minh họa điều này, Hình 1 cung cấp sự trình bày trực quan về không gian tìm kiếm cho ví dụ đang chạy, dưới dạng sơ đồ Hasse. Sơ đồ Hasse là một biểu đồ trong đó mỗi tập mục có thể được biểu diễn dưới dạng một nút và một mũi tên được vẽ từ tập mục X đến tập mục khác Y khi và chỉ khi X ⊆ Y và |X | + 1 = |Y |. Trong Hình 1, các tập mục có tiện ích cao được mô tả bằng các nút màu xám nhạt, trong khi các tập mục có tiện ích thấp được biểu diễn bằng các nút màu trắng. Giá trị tiện ích của mỗi tập mục cũng được chỉ định. Một quan sát quan trọng có thể được rút ra từ hình đó là tiện ích của một tập mục có thể lớn hơn, cao hơn hoặc bằng tiện ích của bất kỳ tập hợp con/tập hợp con nào của nó. Ví dụ: tiện ích của tập mục {b, c} là 28, trong khi tiện ích của tập siêu {b, c, d} và {a, b, c, d, e} của nó lần lượt là 34 và 25. Do đó, về mặt hình thức, người ta nói rằng biện pháp hữu ích không đơn điệu cũng không phản đơn điệu.

Tính chất 1 (Biện pháp hữu ích không đơn điệu cũng không phản đơn điệu). Giả

sử có hai tập mục X và Y sao cho $X \subset Y$. Mối quan hệ giữa ích lợi của X và Y là $u(X) < u(Y)$, $u(X) > u(Y)$, hoặc $u(X) = u(Y)$ [83].

Do đặc tính này, các tập mục có ích cao xuất hiện rải rác trong không gian tìm kiếm, như có thể thấy trong Hình 1. Đây là lý do chính tại sao vấn đề khai thác tập mục có ích cao lại khó khăn hơn vấn đề khai thác tập mục thường xuyên [2]. Trong khai thác tập mục thường xuyên, thước đo hỗ trợ có đặc tính tốt là đơn điệu [2], nghĩa là độ hỗ trợ của một tập mục luôn lớn hơn hoặc bằng tần số của bất kỳ tập thay thế nào của nó.

Tính chất 2 (Biện pháp hỗ trợ mang tính đơn điệu). Giả sử có hai tập mục X và Y sao cho $X \subset Y$. Suy ra $sup(X) \geq sup(Y)$ [2].

Ví dụ: trong cơ sở dữ liệu của Bảng 1, độ hỗ trợ của {b, c} là 3, trong khi độ hỗ trợ của các tập con {b, c, d} và {a, b, c, d, e} của nó là 2 và 1 , tương ứng. Tính đơn điệu của thước đo hỗ trợ giúp dễ dàng tìm thấy các mẫu thường xuyên vì nó

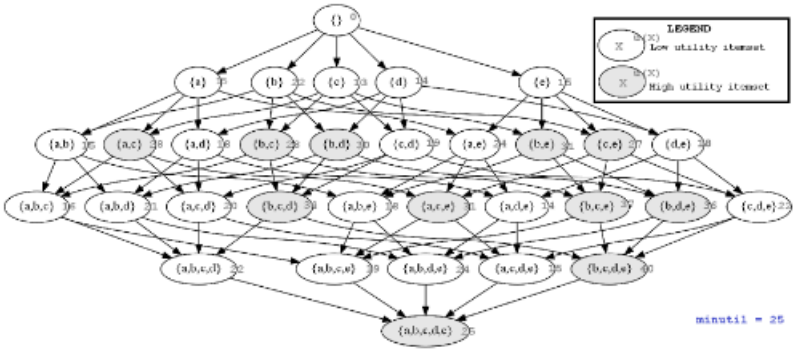
đảm bảo rằng tất cả các siêu tập của một tập mục không thường xuyên cũng không thường xuyên [2]. Do đó, thuật toán khai thác tập phổ biến có thể loại bỏ tất cả các siêu tập của tập phổ biến khỏi không gian tìm kiếm. Ví dụ: nếu một thuật toán nhận thấy tập mục {a, d} không phổ biến, nó có thể loại bỏ trực tiếp tất cả các siêu tập hợp của {a, d} khỏi việc khám phá thêm, do đó làm giảm đáng kể không gian tìm kiếm. Không gian tìm kiếm cho cơ sở dữ liệu mẫu của Bảng 1 được minh họa trong Hình 2. Tính chống đơn điệu của điểm hỗ trợ có thể được quan sát rõ ràng trong hình này khi một đường được vẽ để phân tách rõ ràng các tập mục phổ biến với các tập mục không thường xuyên. Tính chất 2 còn được gọi là tính chất đóng xuống, tính chất chống đơn điệu hoặc tính chất Apriori [2]. Mặc dù nó đúng với thước đo hỗ trợ nhưng nó không đúng với thước đo tiện ích được sử dụng trong khai thác tập mục có tiện ích cao. Kết quả là, trong Hình 1, không thể vẽ một đường rõ ràng để phân biệt các tập mục có ích lợi thấp với các tập mục có ích lợi cao.

Hình 1: Không gian tìm kiếm khai thác tập mục có tiện ích cao cho ví dụ đang chạy và minutil = 25

Do không gian tìm kiếm lớn trong khai phá tập mục có tính tiện ích cao, do đó điều quan trọng là phải thiết kế các thuật toán nhanh có thể tránh xem xét tất cả các tập mục có thể có trong không gian tìm kiếm và xử lý từng tập mục trong không gian tìm kiếm một cách hiệu quả nhất có thể, trong khi vẫn tìm thấy tất cả các tập mục có giá trị cao. tập mục tiện ích. Hơn nữa, do thước đo tiện ích không đơn điệu cũng không chống đơn điệu, nên các chiến lược hiệu quả để giảm không gian tìm kiếm được sử dụng trong khai thác tập mục phổ biến không thể được sử dụng trực tiếp để giải quyết vấn đề khai thác tập mục hữu ích cao. Phần tiếp theo giải thích các ý tưởng chính được sử dụng bởi các thuật toán khai thác tập mục có tiện ích cao hiện đại để giải quyết vấn đề một cách hiệu quả.

3 thuật toán

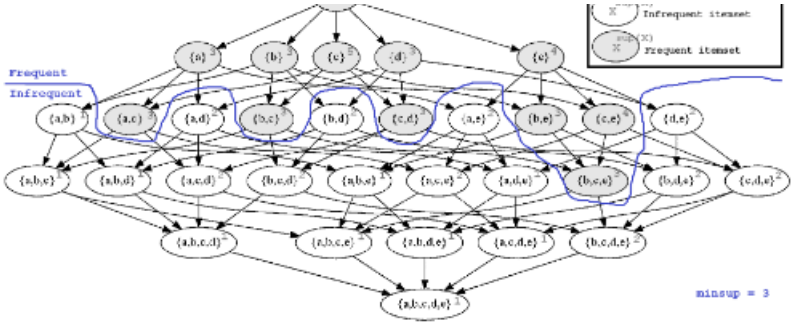
Một số thuật toán khai thác tập mục hữu ích cao đã được đề xuất như UMining [82], Two-Phase [59], IHUP [5], UP-Growth [79], HUP-Growth [52], MU-



Hình 2: Không gian tìm kiếm khai thác tập phổ biến cho cơ sở dữ liệu Bảng 1 và minsup = 3

Tăng trưởng [87], HUI-Miner [58], FHM [31], ULB-Miner [17], HUI-Miner* [71] và EFIM [94]. Tất cả các thuật toán này đều có cùng đầu vào và đầu ra giống nhau. Sự khác biệt giữa các thuật toán này nằm ở cấu trúc dữ liệu và chiến lược được sử dụng để tìm kiếm các tập mục có tính tiện ích cao. Cụ thể hơn, các thuật toán khác nhau ở (1) chúng sử dụng tìm kiếm theo chiều sâu hay theo chiều rộng, (2) kiểu biểu diễn cơ sở dữ liệu mà chúng sử dụng bên trong hay bên ngoài, (3) cách chúng tạo ra hoặc xác định các tập mục tiếp theo. được khám phá trong không gian tìm kiếm và (4) cách chúng tính toán tiện ích của các tập mục để xác định xem chúng có thỏa mãn ràng buộc tiện ích tối thiểu hay không. Những lựa chọn thiết kế này ảnh hưởng đến hiệu suất của các thuật toán này về mặt thời gian thực hiện, mức sử dụng bộ nhớ và khả năng mở rộng cũng như mức độ dễ dàng triển khai và mở rộng các thuật toán này cho các tác vụ khai thác dữ liệu khác. Nói chung, tất cả các thuật toán khai thác tập mục hữu ích cao đều được lấy cảm hứng từ các thuật toán khai thác tập mục phổ biến cổ điển, mặc dù chúng cũng đưa ra những ý tưởng mới để đối phó với thực tế là thước đo tiện ích không đơn điệu cũng không phản đơn điệu.

Các thuật toán ban đầu cho vấn đề khai thác tập mục hữu ích cao là các thuật toán chưa hoàn chỉnh, không thể tìm thấy tập mục hữu ích cao hoàn chỉnh do sử dụng chiến lược heuristic để giảm không gian tìm kiếm. Ví dụ, đây là trường hợp của thuật toán UMining và UMining_H [82]. Trong phần còn lại của phần này, các thuật toán hoàn chỉnh sẽ được xem xét, đảm bảo tìm thấy tất cả các tập mục có tính tiện ích cao. Cũng rất thú vị khi lưu ý rằng thuật ngữ khai thác tập mục có tiện ích cao đã được sử dụng lần đầu tiên vào năm 2003 [11], mặc dù định nghĩa vấn đề được hầu hết các nhà nghiên cứu hiện nay sử dụng và sử dụng trong chương này đã được đề xuất vào năm 2005 [83].



3.1 Thuật toán hai pha

Các thuật toán hoàn chỉnh đầu tiên để tìm các tập mục có tiện ích cao thực hiện hai giai đoạn và do đó được gọi là thuật toán hai giai đoạn. Điều này bao gồm các thuật toán như Hai pha [59], IHUP [5], UP-Growth [79], HUP-Growth [52] và MU-Growth [87]. Ý tưởng đột phá đã truyền cảm hứng cho tất cả các thuật toán này đã được giới thiệu trong Two-Phase [59]. Có thể xác định thước đo đơn điệu là giới hạn trên của thước đo tiện ích và sử dụng thước đo đó để giảm không gian tìm kiếm một cách an toàn mà không bỏ sót bất kỳ tập mục hữu ích cao nào. Biện pháp được đề xuất trong thuật toán Hai pha là biện pháp TWU (Mức sử dụng trọng số giao dịch), được xác định như sau:

Định nghĩa 7 (Thước đo TWU). Tiện ích giao dịch (TU) của giao dịch Tc là tổng hữu dụng của tất cả các phần tử trong Tc. tức là TU(Tc) = Ox ∈Tc u(x, Tc). Mức sử dụng theo trọng số giao dịch (TWU) của tập mục X được định nghĩa là tổng tiện ích giao dịch của các giao dịch chứa X, tức là TWU(X) = OTc ∈g(X) TU(Tc).

Chẳng hạn, tiện ích giao dịch của T0, T1, T2, T3 và T4 lần lượt là 25, 20, 8, 22 và 9. TWU của các mục a, b, c, d, e lần lượt là 55, 54, 84, 53 và 76. TWU của tập mục {c, d} là TWU({c, d}) = TU(T0) + TU(T1) + TU(T2) = 25 + 20 + 8 = 53. Độ đo TWU được cho là giới hạn trên của độ đo tiện ích đơn điệu. Ý tưởng này được chính thức hóa như là thuộc tính tiếp theo.

Thuộc tính 3 (TWU là giới hạn trên đơn điệu của thước đo tiện ích). Giả sử có một tập mục X. TWU của X không nhỏ hơn tiện ích của nó (TWU(X) ≥ u(X)). Hơn nữa, TWU của X không kém gì tiện ích của các siêu tập hợp của nó (TWU(X) ≥ u(Y)∀Y ⊃ X). Chứng minh được đưa ra trong [59]. Theo trực giác, vì TWU của X là tổng tiện ích của các giao dịch trong đó X xuất hiện nên TWU của nó phải lớn hơn hoặc bằng tiện ích của X và bất kỳ tập hợp siêu nào của nó.

Biện pháp TWU rất thú vị vì nó có thể được sử dụng để giảm không gian tìm kiếm. Với mục đích này, tính chất sau đây đã được đề xuất.

Thuộc tính 4 (Cắt bớt không gian tìm kiếm bằng TWU). Đối với bất kỳ tập mục X nào, nếu TWU(X) < minutil thì X là tập mục có lợi ích thấp cũng như tất cả các tập siêu của nó. Điều này trực tiếp theo sau Thuộc tính 3.

Ví dụ: tiện ích của tập mục {a, b, c, d} là 20 và TWU({a, b, c, d}) = 25. Do đó, theo Thuộc tính 4, người ta biết rằng bất kỳ tập con nào của {a, b, c, d} không thể có TWU và tiện ích lớn hơn 25. Kết quả là, nếu người dùng đặt ngưỡng minutil thành giá trị lớn hơn 25, thì tất cả các tập hợp thay thế của {a, b, c, d} có thể bị loại khỏi không gian tìm kiếm vì Thuộc tính 4 đã biết rằng tiện ích của chúng không thể lớn hơn 25.

Các thuật toán như IHUP [5], PB [47], Two-Phase [59], UP-Growth [79], HUP-Growth [52] và MU-Growth [87] sử dụng Thuộc tính 4 làm thuộc tính chính để cắt tỉa tìm kiếm không gian. Chúng hoạt động theo hai giai đoạn:

1. Trong giai đoạn đầu tiên, các thuật toán này tính toán TWU của các tập mục trong không gian tìm kiếm. Đối với tập mục X, nếu TWU(X) < X thì X và các tập siêu của nó không thể là tập mục có tính tiện ích cao. Vì vậy, chúng có thể được loại bỏ khỏi không gian tìm kiếm và TWU của chúng không cần phải tính toán. Ngược lại, X và các tập siêu của nó có thể là các tập mục có tính tiện ích cao. Do đó, X được lưu giữ trong bộ nhớ như một tập mục có tính tiện ích cao và các tập siêu của nó có thể được khám phá.

2. Trong giai đoạn thứ hai, tiện ích chính xác của từng tập mục hữu ích cao ứng cử viên X được tìm thấy trong giai đoạn 1 được tính toán bằng cách quét cơ sở dữ liệu. Nếu u(X) ≥ minutil thì X là đầu ra vì đây là tập mục có tính tiện ích cao.

Quá trình hai giai đoạn này đảm bảo rằng chỉ các tập mục có tiện ích thấp mới được lược bỏ khỏi không gian tìm kiếm. Do đó, thuật toán hai pha có thể tìm thấy tất cả các tập mục có tiện ích cao đồng thời giảm không gian tìm kiếm để cải thiện hiệu suất của chúng. Thuật toán hai pha đại diện là Hai pha [59]. Nó được mô tả tiếp theo và sau đó những hạn chế của nó sẽ được thảo luận.

3.1.1 Thuật toán hai pha

Thuật toán hai pha tổng quát hóa thuật toán Apriori, được đề xuất để khai thác tập mục thường xuyên [2]. Hai pha khám phá không gian tìm kiếm của các tập mục bằng cách sử dụng tìm kiếm theo chiều rộng. Thuật toán tìm kiếm theo chiều rộng trước tiên sẽ xem xét các mục đơn lẻ (bộ 1 mục). Trong ví dụ đang chạy, đó là {a}, {b}, {c}, {d} và {e}. Sau đó, Two-Phase tạo ra 2 tập mục như {a, b}, {a, c}, {a, d}, rồi đến tập 3 mục, v.v., cho đến khi nó tạo ra tập mục lớn nhất {a, b, c, d, e} chứa tất cả các mục. Hai pha [59] lấy cơ sở dữ liệu giao dịch định lượng và ngưỡng tối thiểu làm đầu vào. Hai pha sử dụng cách biểu diễn cơ sở dữ liệu tiêu chuẩn, như trong Bảng 3, còn được gọi là cơ sở dữ liệu theo chiều ngang. Mã giả của Hai pha được đưa ra trong Thuật toán 1. Trong giai đoạn 1, Hai pha quét cơ sở dữ liệu để tính TWU của mỗi tập hợp 1 mục (dòng 1). Sau đó, Two-Phase sử dụng thông tin này để xác định tập hợp tất cả các mục có tính ứng dụng cao, được ký hiệu là P1 (dòng 2). Một tập mục X được gọi là tập mục có ích cao nếu TWU(X) ≥ minutil. Sau đó, Two-Phase thực hiện tìm kiếm theo chiều rộng để tìm các tập mục có ứng viên có ích cao hơn (dòng 4 đến 10). Trong quá trình tìm kiếm, Two-Phase sử dụng các tập mục có tính tiện ích cao có thể có độ dài k

- 1 cho trước (ký hiệu là Pk-1) để tạo ra các tập mục có độ dài k (ký hiệu là Pk). Điều này được thực hiện bằng cách kết hợp các cặp tập mục hữu ích cao có độ dài k có chung tất cả trừ một mục (dòng 5). Ví dụ: nếu các tập mục 1 mục có tiện ích cao ứng cử viên là {a}, {b}, {c} và {e}, thì hãy kết hợp hai pha của các tập mục này để thu được các tập mục 2 sau: {a, b}, {a, c}, {a, e}, {b, c}, {b, e} và {c, e}. Sau khi tạo các tập mục có độ dài k, Two-Phase kiểm tra xem các tập con (k − 1) của mỗi tập mục có phải là tập mục có ích cao hay không. Nếu một tập mục X có một tập mục con (k − 1) không phải là tập mục có ích cao ứng viên thì X không thể là tập mục có ích cao (nó sẽ vi phạm Thuộc tính 4) và do đó nó bị loại khỏi tập k-mục. Sau đó, Two-Phase quét cơ sở dữ liệu để tính toán TWU của tất cả các tập mục còn lại trong Pk (dòng 7). Mỗi tập mục có TWU không nhỏ hơn minutil sẽ được thêm vào tập Pk gồm k-itemset có tính hữu ích cao ứng cử viên (dòng 8). Quá trình này được lặp lại cho đến khi không còn ứng viên nào có tính hữu dụng cao

các tập mục có thể được tạo ra. Sau đó, giai đoạn thứ hai được thực hiện (dòng 12 đến 13). Hai pha quét cơ sở dữ liệu để tính toán tiện ích chính xác của từng tập mục có tiện ích cao ứng viên. Tập hợp tất cả các tập mục hữu ích cao ứng cử viên có tiện ích không nhỏ hơn minutil là các tập mục hữu ích cao. Chúng được trả lại cho người dùng (dòng 13).

Thuật toán 1: Thuật toán hai pha

đầu vào : D: cơ sở dữ liệu giao dịch theo chiều ngang, minutil: đầu ra ngưỡng do người dùng chỉ định: tập hợp các tập mục tiện ích cao

1 Quét cơ sở dữ liệu để tính TWU của tất cả các mục trong I; // GIAI ĐOẠN 1

2 P1 = {i |i ∈ I ∧ sup({i }) ≥ minsup } ; // P1 : ứng viên có độ hữu dụng cao 1-tập 3 k = 2;

4 trong khi Pk , ∅ do 5 Pk = itemsetGeneration (Pk−1) ; // Pk : k-itemset

6 Loại bỏ từng ứng cử viên X ∈ Pk có chứa tập mục (k − 1) không thuộc Pk−1;

7 Quét cơ sở dữ liệu để tính TWU của từng ứng viên X ∈ Pk ;

8 Pk = {X |X ∈ Pk ∧ TWU(X) ≥ minutil } ; // Pk : ứng viên cao tiện ích k-mục

9 k = k + 1;

10 kết thúc

11 P = –k=1...k Pk ; // P : tất cả các tập mục có ích cao ứng cử viên

12 Quét cơ sở dữ liệu để tính toán tiện ích của từng tập mục trong P ; // GIAI ĐOẠN 2

13 trả về mỗi tập mục X ∈ P sao cho u(X) ≥ minutil;

Hai pha là một thuật toán quan trọng, vì nó là một trong những thuật toán khai thác tập mục có tiện ích cao hoàn chỉnh đầu tiên và nó đã giới thiệu giới hạn trên TWU, được sử dụng bởi hầu hết các thuật toán khai thác tập mục có tiện ích cao sau đó. Tuy nhiên, Two-Phase có những hạn chế quan trọng về hiệu suất. Đầu tiên là vì Two-Phase tạo các tập mục bằng cách kết hợp các tập mục mà không cần xem cơ sở dữ liệu, nên nó có thể tạo ra một số mẫu thậm chí không xuất hiện trong cơ sở dữ liệu. Do đó, Two-Phase có thể dành một lượng lớn thời gian để xử lý các tập mục không tồn tại trong cơ sở dữ liệu. Hạn chế thứ hai là Two-Phase liên tục quét cơ sở

dữ liệu để tính toán TWU và tiện ích của các tập mục, điều này rất tốn kém. Hạn chê thứ ba là việc sử dụng tìm kiếm theo chiều rộng có thể khá tốn kém về mặt bộ nhớ vì nó đòi hỏi phải giữ trong trường hợp xấu nhất tất cả các tập mục k và tập mục (k − 1) trong bộ nhớ (với k > 1).). Hơn nữa, Two-Phase phải giữ tất cả các tập mục hữu ích cao ứng viên trong bộ nhớ trước khi thực hiện giai đoạn thứ hai. Nhưng Two-Phase có thể tạo ra một lượng lớn ứng viên để tìm ra một số tập mục có tính tiện ích cao [79]. Lý do là TWU có giới hạn trên lỏng lẻo về tiện ích của các tập mục. Trong các nghiên cứu tiếp theo, các giới hạn trên chặt chẽ hơn đã được thiết kế và các kỹ thuật để giảm các giới hạn trên này. Về độ phức tạp, Two-Phase dựa trên Apriori. Một phân tích phức tạp rất chi tiết của thuật toán Apriori đã được Hegland thực hiện [40]. Tóm lại, độ phức tạp về thời gian của Apriori là O(m2n), trong đó m là số mục riêng biệt và n là số lượng giao dịch. Về độ phức tạp, điểm khác biệt chính giữa Apriori và Two-Phase là

sau này thực hiện giai đoạn thứ hai trong đó tiện ích chính xác của từng mẫu được tính toán bằng cách quét cơ sở dữ liệu. Các tối ưu hóa khác nhau có thể được sử dụng để giảm chi phí của giai đoạn thứ hai, chẳng hạn như lưu trữ các tập mục trong cây băm để tránh so sánh từng tập mục với mỗi giao dịch [2]. Tuy nhiên, giai đoạn thứ hai vẫn còn rất tốn kém [79, 90].

3.1.2 Thuật toán hai pha tăng trưởng mẫu

Để giải quyết một số nhược điểm của thuật toán Hai pha, một số thuật toán tăng trưởng mẫu đã được đề xuất như IHUP [5], UP-Growth [79], HUP-Growth [52], PB [47] và MU-Growth [87]. Khái niệm thuật toán tăng trưởng mẫu lần đầu tiên được sử dụng trong các thuật toán khai thác tập mục thường xuyên như FP-Growth [39], H-Mine [69] và LCM [81]. Ý tưởng chính của thuật toán tăng trưởng mẫu là quét cơ sở dữ liệu để tìm các tập mục và do đó tránh tạo ra các tập mục không xuất hiện trong cơ sở dữ liệu. Hơn nữa, để giảm chi phí quét cơ sở dữ liệu, các thuật toán tăng trưởng mẫu đã đưa ra các biểu diễn cơ sở dữ liệu nhỏ gọn và khái niệm cơ sở dữ liệu dự kiến để giảm kích thước cơ sở dữ liệu khi thuật toán khám phá các tập mục lớn hơn.

Tất cả các thuật toán tăng trưởng mẫu được thảo luận trong chương này đều sử dụng tìm kiếm theo chiều sâu thay vì tìm kiếm theo chiều rộng để khám phá không gian tìm kiếm của các tập mục. Ưu điểm của việc sử dụng cái trước thay vì cái sau là ít tập mục cần được lưu giữ trong bộ nhớ hơn trong quá trình tìm kiếm. Thuật toán tìm kiếm theo chiều sâu bắt đầu từ mỗi tập mục 1 và sau đó cố gắng nối các mục vào tập mục hiện tại để tạo ra các tập mục lớn hơn. Ví dụ: trong ví dụ đang chạy, thuật toán tìm kiếm theo chiều sâu điển hình sẽ khám phá các tập mục theo thứ tự đó: {a}, {a, b}, {a, b, c}, {a, b, c, d }, {a, b, c, d, e}, {a, b, c, e}, {a, b, d}, {a, b, d, e}, {a, b, e}, {a, c}, {a, c, d}, {a, c, d, e}, {a, c, e}, {a, d}, {a, d, e}, {a, e}, {b}, {b, c}, {b , c, d}, {b, c, d, e}, {b, c, e}, {b, d}, {b, d, e}, {b, e}, {c}, {c , d}, {c, d, e}, {c, e}, {d}, {d, e}, {e}.

Mã giả của thuật toán tăng trưởng mẫu hai pha điển hình để khai thác tập mục có tiện ích cao được hiển thị trong Thuật toán 2. Nó lấy đầu vào là cơ sở dữ liệu giao dịch định lượng D và ngưỡng minutil. Không mất tính tổng quát, giả sử rằng tồn tại một thứ

tự tông cộng trên các mục $\prec$ chăng hạn như thứ tự từ điên ($a \prec b \prec c \prec d \prec e$). Thuật toán tăng trưởng mẫu trước tiên tạo một tập P để lưu trữ các tập mục hữu ích cao ứng viên (dòng 1). Sau đó, thuật toán quét cơ sở dữ liệu D để tính toán tiện ích giao dịch, ký hiệu là TU (dòng 2). Sau đó, thuật toán khám phá không gian tìm kiếm bằng cách sử dụng tìm kiếm theo chiều sâu bằng cách thêm đệ quy các mục theo thứ tự $\prec$ vào các tập mục có ích cao ứng cử viên, để thu được các tập mục có ích cao ứng viên lớn hơn. Quá trình này được thực hiện bằng cách gọi thủ tục RecursiveGrowth, được mô tả trong Thuật toán 3. Lúc đầu, thủ tục RecursiveGrowth coi rằng tập mục X hiện tại là tập trống. Thủ tục quét cơ sở dữ liệu D để tìm tập Z của tất cả các mục trong D là các tập mục có ích cao ứng cử viên (dòng 1 và 2). Sau đó, với mỗi mục z như vậy, tập mục $X \cup \{z\}$ được lưu trong tập các tập mục hữu ích cao ứng cử viên P (dòng 4). Sau đó, quy trình tăng trưởng mẫu được gọi để thực hiện tìm kiếm theo chiều sâu để tìm các tập phổ biến lớn hơn là phần mở rộng của $X \cup \{z\}$ theo cách tương tự (dòng 6). Tuy nhiên, có thể thấy rằng không phải tất cả các mục trong D đều có thể được thêm vào $X \cup \{z\}$ để tạo ra các tập mục lớn hơn. Trong thực tế, tập mục $X \cup \{z\}$ có thể không

thậm chí xuất hiện trong tất cả các giao dịch của cơ sở dữ liệu D. Vì lý do này, thuật toán tăng trưởng mẫu sẽ tạo cơ sở dữ liệu dự kiến của tập mục X ∪ {z} (dòng 5) và sẽ sử dụng cơ sở dữ liệu này để thực hiện tìm kiếm theo chiều sâu (dòng 6). Điều này sẽ cho phép giảm chi phí quét cơ sở dữ liệu. Sau khi thực hiện đệ quy tìm kiếm theo chiều sâu cho tất cả các mục, tập hợp tất cả các tập mục có ích cao ứng cử viên P đã được tạo.

Sau đó, Thuật toán 2 thực hiện giai đoạn thứ hai theo cách tương tự như thuật toán Hai giai đoạn được mô tả trước đó. Cơ sở dữ liệu được quét để tính toán độ thỏa dụng chính xác của từng tập mục có độ thỏa dụng cao ứng viên (dòng 4). Những thứ có tiện ích không nhỏ hơn ngưỡng minutil sẽ được trả lại cho người dùng (dòng 5).

Bây giờ, hãy minh họa các bước này chi tiết hơn bằng một ví dụ. Xem xét cơ sở dữ liệu trong Bảng 3 và 4 và giả sử minutil = 25. Trong giai đoạn 1, thuật toán quét cơ sở dữ liệu và tìm thấy các tập 1 mục {a}, {b}, {c} và {e}, có các giá trị TWU là lần lượt là 55, 54, 84, 53 và 76. Do đó, các tập mục này là các tập mục có tính tiện ích cao. Đầu tiên, thuật toán xem xét mục a để cố gắng tìm các tập mục ứng cử viên lớn hơn bắt đầu bằng tiền tố {a}. Sau đó, thuật toán xây dựng cơ sở dữ liệu dự kiến của {a} như trong Bảng 8. Cơ sở dữ liệu dự kiến của mục i được định nghĩa là tập hợp các giao dịch mà tối xuất hiện, nhưng trong đó mục i và các mục trước i theo thứ tự ≺ có đã được gỡ bỏ. Sau đó, để tìm các tập mục ứng viên bắt đầu bằng {a} chứa thêm một mục nữa, thuật toán sẽ quét cơ sở dữ liệu dự kiến của {a} và đếm TWU của tất cả các mục xuất hiện trong cơ sở dữ liệu đó. Ví dụ: TWU của các mục trong cơ sở dữ liệu dự kiến của {a} là: {b} : 25, {c} : 55 và {e} : 47. Điều này có nghĩa là TWU của {a, b} là 25, nghĩa là TWU của {a, c} là 55 và TWU của {a, e} là 47. Vì ba tập mục này có TWU không nhỏ hơn minutil nên các tập mục này là các tập mục có ích cao ứng cử viên và là tập tiếp theo được sử dụng để cố gắng tạo ra các tập mục lớn hơn bằng cách thực hiện tìm kiếm theo chiều sâu bắt đầu từ mỗi tập mục. Tập mục {a, c} được xem xét đầu tiên. Thuật toán xây dựng cơ sở dữ liệu dự kiến của {a, c} từ cơ sở dữ liệu dự kiến của {a}. Cơ sở dữ liệu dự kiến của {a, c} được hiển thị trong Bảng 9. Sau đó, thuật toán quét cơ sở dữ liệu dự kiến của {a, c} để tìm các mục có TWU không nhỏ hơn minutil trong cơ sở dữ liệu đó. Quá trình này sẽ tiếp tục cho đến khi tất cả các tập mục hữu ích cao được tìm thấy bằng tìm kiếm theo chiều sâu. Sau đó, trong giai đoạn 2, cơ sở dữ liệu được quét để tính toán tiện ích chính xác của tất cả các ứng cử viên được tìm thấy trong giai đoạn 1. Sau đó, các tập mục có tiện ích nhỏ hơn minutil sẽ bị loại bỏ. Các tập mục còn lại được xuất ra dưới dạng các tập mục có tính tiện ích cao. Kết quả được thể hiện ở Bảng 5.

Ưu điểm chính của thuật toán tăng trưởng mẫu là chúng chỉ khám phá các tập mục thực sự xuất hiện ít nhất một lần trong cơ sở dữ liệu đầu vào, trái ngược với các thuật toán dựa trên Apriori, có thể tạo ra các mẫu không xuất hiện trong cơ sở dữ liệu. Ngoài ra, khái niệm cơ sở dữ liệu dự kiến cũng hữu ích trong việc giảm chi phí quét cơ sở dữ liệu vì cơ sở dữ liệu dự kiến nhỏ hơn cơ sở dữ liệu ban đầu. Một câu hỏi phổ biến về khái niệm cơ sở dữ liệu dự kiến là: việc tạo ra tất cả các bản sao của cơ sở dữ liệu gốc này có tốn kém không? Câu trả lời là không nếu sử dụng tối ưu hóa được gọi là phép chiếu giả, bao gồm việc triển khai cơ sở dữ liệu được chiếu dưới dạng một tập hợp các con trỏ trên cơ sở dữ liệu gốc thay vì dưới dạng bản sao [69, 81]. Ví dụ: Hình 3 hiển thị cơ sở dữ liệu giả dự kiến của {a, c}, tương đương với cơ sở dữ liệu dự kiến của Bảng 4, ngoại trừ việc nó được triển khai bằng cách sử dụng ba con trỏ trên cơ sở dữ liệu gốc,

để tránh tạo bản sao của cơ sở dữ liệu gốc. Lưu ý rằng nhiều cách tối ưu hóa khác cũng có thể được tích hợp vào các thuật toán tăng trưởng mẫu. Ví dụ: các thuật toán IHUP [5], UP-Growth [79], HUP-Growth [52] và MU-Growth [87] sử dụng cấu trúc cây tiền tố để biểu diễn cơ sở dữ liệu dự kiến và giảm mức sử dụng bộ nhớ. Các cấu trúc này mở rộng cấu trúc cây FP được sử dụng trong khai thác tập mục thường xuyên bằng thuật toán FPGrowth [39]. Sự khác biệt chính giữa các thuật toán này nằm ở việc sử dụng nhiều chiến lược khác nhau để giảm giới hạn trên của TWU đối với tiện ích. Trong số các thuật toán hai giai đoạn, UP-Growth là một trong những thuật toán nhanh nhất. Nó được chứng minh là nhanh hơn tới 1.000 lần so với Hai pha và IHUP. Các thuật toán hai pha gần đây hơn như PB và MU-Growth đã đưa ra nhiều cách tối ưu hóa và thiết kế khác nhau nhưng chỉ cung cấp một cải thiện tốc độ nhỏ so với Hai pha hoặc UP-Growth (MU-Growth được báo cáo là chỉ nhanh hơn UP tới 15 lần). -Sự phát triển).

Mặc dù thuật toán hai giai đoạn đã được nghiên cứu kỹ lưỡng và đưa ra nhiều ý tưởng quan trọng nhưng chúng vẫn chưa hiệu quả. Như đã giải thích, thuật toán hai pha khai thác các tập mục hữu ích cao theo hai giai đoạn. Trong giai đoạn 1, một tập hợp các ứng cử viên được tìm thấy. Sau đó, trong giai đoạn 2, tính hữu dụng của các ứng viên này được tính toán bằng cách quét cơ sở dữ liệu. Sau đó, các tập mục có lợi ích thấp được lọc và các tập mục có lợi ích cao được trả về cho người dùng. Cách tiếp cận này không hiệu quả vì tập các tập mục ứng cử viên được tìm thấy trong giai đoạn 1 có thể rất lớn và việc thực hiện giai đoạn thứ hai để đánh giá các ứng cử viên này rất tốn kém [79, 90]. Trong trường hợp xấu nhất, tất cả các tập mục ứng viên được so sánh với tất cả các giao dịch của cơ sở dữ liệu trong giai đoạn thứ hai. Do đó, hiệu suất của thuật toán hai giai đoạn bị ảnh hưởng nhiều bởi số lượng ứng cử viên được tạo ra để tìm các tập mục hữu ích thực tế cao. Để giảm số lượng ứng viên, nhiều chiến lược khác nhau đã được thiết kế để giảm giới hạn trên của TWU và do đó thu gọn nhiều ứng viên hơn [5, 52, 79, 87]. Nhưng để giải quyết vấn đề cơ bản của thuật toán hai pha, đó là tạo ra các ứng cử viên, thuật toán một pha đã được thiết kế và sẽ được mô tả tiếp theo.

Thuật toán 2: Thuật toán tăng trưởng mô hình hai pha

đầu vào : D: cơ sở dữ liệu giao dịch định lượng, minutil: ngưỡng hữu dụng tối thiểu đầu ra : tập hợp các tập mục hữu ích cao

```
1 P = ∅ ;                    // P: tất cả các tập mục có ích cao ứng cử viên
```

2 Quét cơ sở dữ liệu tính toán TU, tiện ích giao dịch của giao dịch tại D; 3 Tăng trưởng đệ quy (TUs, D, ∅, minutil, P); // GIAI ĐOẠN 1 4 Quét cơ sở dữ liệu để tính toán tiện ích của từng tập mục trong P; // GIAI ĐOẠN 2 5 trả về từng tập mục X ∈ P sao cho u(X) ≥ minutil;

3.2 Thuật toán một pha

Bước đột phá lớn thứ hai trong khai thác tập mục có tính tiện ích cao là việc thiết kế các thuật toán không tạo ra các ứng viên. Các thuật toán một pha này ngay lập tức

Thuật toán 3: Thủ tục tăng trưởng đệ quy

đầu vào : TUs: tiện ích giao dịch trong cơ sở dữ liệu gốc, D: cơ sở dữ liệu giao dịch định lượng, X: tập mục hiện tại, minutil: ngưỡng tiện ích tối thiểu, P: tập lưu trữ các tập mục hữu ích cao ứng viên

đầu ra : tập các tập mục hữu ích cao 1 Quét cơ sở dữ liệu D để tính TWU của từng mục trong I sử dụng TU; 2 W = {i |i ∈ I ∧ TWU({i }) ≥ minutil }; // W : ứng viên có tính ứng dụng cao

```
1-tập mục trong D 3
cho mỗi mục z ∈ W do
4       Thêm X ∪ {z } vào P;
5D′ = Phép chiếu(D, z) ;              // Tạo cơ sở dữ liệu dự kiến của X ∪ {z }
6Tăng trưởng đệ quy (TUs, D′, {z }, minutil, P);                    // gọi đệ quy tới
mở rộng X ∪ {z } 7
kết thúc
```

Bảng 9: Cơ sở dữ liệu dự kiến của {a, c}

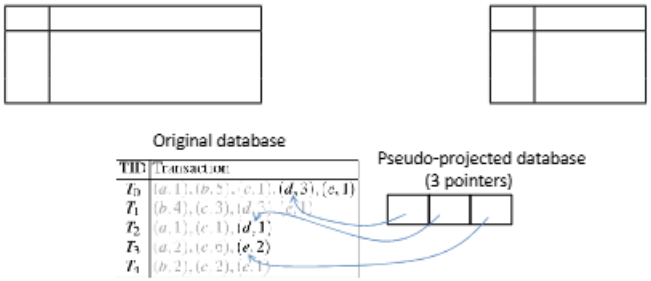
Giao dịch TID T0 (b, 5), (c, 1), (d, 3), (e, 1)	Giao dịch TID T0 (d, 3), (e, 1)
T2 (c, 1), (d, 1)	T2 (d, 1)
T3 (c, 6), (e, 2)	T3 (e, 2)

Cơ sở dữ liệu gốc

Cơ sở dữ liệu giả (3 con trỏ)

Hình 3: Cơ sở dữ liệu giả của {a, c}

tính toán tiện ích của từng mẫu được xem xét trong không gian tìm kiếm. Do đó, một tập mục có thể được xác định ngay lập tức là tập mục có ích lợi thấp hoặc có ích cao và các ứng cử viên không cần phải được lưu trữ trong bộ nhớ. Khái niệm về thuật toán một pha lần đầu tiên được xuất bản trong HUI-Miner [58, 71], và sau đó là trong thuật toán d2HUP [60]. Sau đó, các thuật toán một pha được cải tiến và hiệu quả hơn đã được thiết kế như FHM [31], mHUIMiner [70], ULB-Miner [17], HUI-Miner* [71] và EFIM [94]. Bên cạnh tính mới của việc khám phá các tập mục có tiện ích cao trong một pha, các thuật toán một pha cũng đã đưa ra các giới hạn trên mới về tiện ích của các tập mục dựa trên tiện ích chính xác của từng tập mục và do đó có thể cắt bớt phần lớn hơn của không gian tìm kiếm so với thước đo TWU. Các giới hạn trên này bao gồm tiện ích còn lại [58, 60] và các biện pháp mới hơn như tiện ích cục bộ và tiện ích cây con [94]. Các phần tiếp theo sẽ cung cấp cái nhìn tổng quan về thuật toán một pha.



3.2.1 Thuật toán FHM

Một trong những loại thuật toán khai thác tập mục tiện ích cao phổ biến nhất là các thuật toán dựa trên cấu trúc danh sách tiện ích. Cấu trúc này được giới thiệu trong thuật toán HUI-Miner [58] bằng cách tổng quát hóa cấu trúc danh sách tid [91] được sử dụng trong khai thác tập mục thường xuyên. Sau đó, các thuật toán dựa trên danh sách tiện ích nhanh hơn đã được đề xuất như FHM [31], mHUIMiner [70] và ULB-Miner [17], và các phần mở rộng đã được đề xuất cho một số biến thể của bài toán khai thác tập mục tiện ích cao. Lý do cho sự phổ biến của các thuật toán dựa trên danh sách tiện ích là vì chúng nhanh và dễ thực hiện. Tiểu mục này mô tả thuật toán FHM [31] như một thuật toán dựa trên danh sách tiện ích đại diện, được chứng minh là nhanh hơn tới bảy lần so với HUI-Miner và được nhiều nhà nghiên cứu sử dụng và mở rộng.

FHM là thuật toán một pha thực hiện tìm kiếm theo chiều sâu để khám phá không gian tìm kiếm của các tập mục. Trong quá trình tìm kiếm, thuật toán FHM tạo ra một danh sách tiện ích cho mỗi tập mục được truy cập trong không gian tìm kiếm. Danh sách tiện ích của một tập mục lưu trữ thông tin về tiện ích của tập mục đó trong các giao dịch nơi nó xuất hiện và thông tin về tiện ích của các mục còn lại trong các giao dịch này. Danh sách tiện ích cho phép tính toán nhanh chóng tiện ích của một tập mục và giới hạn trên của tiện ích của các siêu tập hợp của nó mà không cần quét cơ sở dữ liệu. Hơn nữa, danh sách tiện ích của k-tập mục k > 1) có thể được tạo nhanh chóng bằng cách nối danh sách tiện ích của các mẫu ngắn hơn. Cấu trúc danh sách tiện ích được định nghĩa như sau.

Định nghĩa 8 (Danh sách tiện ích). Giả sử có một tập mục X và một cơ sở dữ liệu định lượng D. Không mất tính tổng quát, giả sử rằng tổng thứ tự được xác định trên tập các mục I xuất hiện trong cơ sở dữ liệu đó. Danh sách tiện ích ul(X) của X trong cơ sở dữ liệu định lượng D là một tập hợp các bộ dữ liệu sao cho có một bộ dữ liệu (tid, iutil, rutil) cho mỗi giao dịch Ttid chứa X. Phần tử iutil của bộ dữ liệu là tiện ích của X trong Ttid. tức là u(X, Ttid). Phần tử rutil của một bộ dữ liệu được định nghĩa là Ôi ∈Tt id ∧ix∀x ∈X u(i, Ttid).

Ví dụ: giả sử đó là thứ tự bảng chữ cái. Danh sách tiện ích của {a}, {d} và {a, d}

được hiển thị trong Hình 4. Hãy xem xét danh sách tiện ích của {a}. Nó chứa ba hàng (bộ dữ liệu) tương ứng với các giao dịch T0, T2 và T3 vì {a} xuất hiện trong ba giao dịch này. Cột thứ hai của danh sách tiện ích (giá trị iutil) của {a} chỉ ra rằng tiện ích của {a} trong T0, T2 và T3 lần lượt là 5, 5 và 10. Cột thứ ba của danh sách tiện ích của {a} chỉ ra rằng giá trị rutil của {a} cho các giao dịch T0, T2 và T3 lần lượt là 20, 3 và 10.

Danh sách tiện ích của {a }

thời gian iutil rutile		
T0	5	20
T2	5	3
T3	10	12

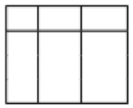
Danh sách tiện ích của {d }

thời gian iutil rutile		
T0	6	3
T1	6	3
T2	2	0

Danh sách tiện ích của {a, d }

thời gian iutil rutil		
T0	11	3
T2	7	0

Hình 4: Danh sách tiện ích của {a}, {d} và {a, d}



Thuật toán FHM quét cơ sở dữ liệu một lần để tạo danh sách tiện ích gồm 1 tập mục (các mục đơn lẻ). Sau đó, danh sách tiện ích của các tập mục lớn hơn được xây dựng bằng cách nối danh sách tiện ích của các tập mục nhỏ hơn. Hoạt động nối cho các mục đơn lẻ được thực hiện như sau. Xét hai mục x, y sao cho xy và tiện ích của chúng là danh sách ul({x}) và ul({y}). Danh sách tiện ích của {x, y} có được bằng cách tạo một bộ dữ liệu (ex.tid, ex.iutil + ey.iutil, ey.rutil) cho mỗi cặp bộ dữ liệu ex ∈ ul({x}) và ey ∈ ul ({y}) sao cho ex.tid = ey.tid. Thao tác nối hai tập mục P ∪ {x} và P ∪ {y} sao cho xy được thực hiện như sau. Đặt ul(P), ul({x}) và ul({y}) là danh sách tiện ích của P, {x} và {y}. Danh sách tiện ích của P ∪ {x, y} có được bằng cách tạo một bộ dữ liệu (ex.tid, ex.iutil + ey.iutil − ep.iutil, ey.rutil) cho mỗi bộ bộ dữ liệu ex ∈ ul({x }), ey ∈ ul({y}), ep ∈ ul(P) sao cho ex.tid = ey.tid = ep.tid. Ví dụ: có thể lấy danh sách tiện ích của {a, d} bằng cách nối danh sách tiện ích của {a} và {d} (được mô tả trong Hình 4) mà không cần quét cơ sở dữ liệu.

Cấu trúc danh sách tiện ích của một tập mục rất hữu ích vì nó cho phép lấy trực tiếp tiện ích của một tập mục mà không cần quét cơ sở dữ liệu.

Thuộc tính 5 (Tính toán tiện ích của một tập mục bằng cách sử dụng danh sách tiện ích của nó). Giả sử có một tập mục X. Tổng các giá trị iutil trong danh sách tiện ích ul(X) của nó bằng với tiện ích của X [58]. Nói cách khác, u(X) = Õe∈ul(X) e.iutil.

Ví dụ: tiện ích của tập mục {a, d} bằng tổng các giá trị trong cột iutil của danh sách tiện ích của nó (được mô tả trong Hình 4). Do đó, bằng cách nhìn vào danh sách tiện ích của {a, d}, người ta thấy rằng tiện ích của nó là u({a, d}) = 11 + 7 = 18.

Danh sách tiện ích của một tập mục cũng được sử dụng để cắt bớt không gian tìm kiếm dựa trên định nghĩa và thuộc tính sau.

Định nghĩa 9 (Tiện ích còn lại giới hạn trên). Cho X là một tập mục. Giả sử các phần mở rộng của X là các tập mục có thể thu được bằng cách thêm một mục y vào X sao cho yi, ∀i ∈ X. Tiện ích còn lại giới hạn trên của X là tổng giá trị iutil và rutil trong tiện ích của nó -list ul(X). Về mặt hình thức, giới hạn trên này được định nghĩa là reu(X) = Õe∈ul(X) (e.iutil + e.rutil). Giá trị reu(X) là giới hạn trên của tiện ích của X và tất cả

các phần mở rộng của nó [58]. Nói cách khác, mối quan hệ u(Y) ≤ reu(X) đúng với bất kỳ tập mục Y nào là phần mở rộng của X.

Ví dụ: hãy xem xét tính toán giới hạn trên tiện ích còn lại của tập mục {a, d} bằng cách sử dụng danh sách tiện ích của nó (được mô tả trong Hình 4). Giới hạn trên là tổng các giá trị trong cột iutil và rutil trong danh sách tiện ích của nó, đó là reu({a, d}) = 11+7 = 18. Do đó, người ta biết rằng tập mục {a, d } và tất cả các phần mở rộng của nó như {a, d, e} không thể có tiện ích lớn hơn 18. Nếu chúng ta giả sử rằng minutil = 25, như trong ví dụ đang chạy, thì các tập mục này có thể được cắt bớt khỏi không gian tìm kiếm, vì chúng sẽ là tập mục có ích lợi thấp. Điều này được chính thức hóa bởi tính chất sau.

Thuộc tính 6 (Cắt bớt không gian tìm kiếm bằng danh sách tiện ích). Cho X là một tập mục. Nếu tổng giá trị iutil và rutil trong ul(X) nhỏ hơn minutil (tức là reu(X) < minutil), thì X và phần mở rộng của nó là tập mục có tiện ích thấp [58].

Quy trình chính của FHM (Thuật toán 4) lấy cơ sở dữ liệu giao dịch định lượng và ngưỡng tối thiểu làm đầu vào. Đầu tiên FHM quét cơ sở dữ liệu để tính toán TWU của từng mục. Sau đó, thuật toán xác định tập I∗ của tất cả các mục có TWU không nhỏ hơn minutil (các mục khác bị bỏ qua vì chúng không thể là một phần của tập mục có tiện ích cao theo Thuộc tính 4). Sau đó, giá trị TWU của các mục được sử dụng để thiết lập thứ tự tổng thể trên các mục, là thứ tự tăng dần của các giá trị TWU (như được đề xuất trong [58]). Việc quét cơ sở dữ liệu sau đó được thực hiện. Trong quá trình quét cơ sở dữ liệu này, các mục trong giao dịch được sắp xếp lại theo tổng thứ tự, danh sách tiện ích của từng mục i ∈ I∗ được xây dựng và cấu trúc có tên EUCS (Cấu trúc xuất hiện tiện ích ước tính) được xây dựng [31]. Cấu trúc sau này được định nghĩa là một tập hợp các bộ ba có dạng (a, b, c) ∈ I∗ × I∗ × R. Bộ ba (a,b,c) chỉ ra rằng TWU({a, b}) = c . EUCS có thể được triển khai dưới dạng ma trận tam giác lưu trữ các bộ ba này cho tất cả các cặp vật phẩm. Ví dụ: EUCS cho ví dụ đang chạy được hiển thị trong Hình 5. EUCS rất hữu ích vì nó lưu trữ TWU của tất cả các cặp mục, thông tin này sẽ được sử dụng sau này để cắt bớt không gian tìm kiếm. Ví dụ: ô trên cùng bên trái cho biết TWU({a, b}) = 25. Việc xây dựng EUCS diễn ra rất nhanh (được thực hiện bằng một lần quét cơ sở dữ liệu) và chiếm một lượng bộ nhớ nhỏ, được giới hạn bởi |I∗ | × |Tôi∗ |. Người đọc có thể tham khảo bài viết về FHM [31] để biết thêm chi tiết về việc xây dựng cấu trúc này và tối ưu hóa việc triển khai. Sau khi xây dựng EUCS, việc khám phá tìm kiếm theo chiều sâu của các tập mục bắt đầu bằng cách gọi thủ tục đệ quy FH MSearch với tập mục trống ∅, tập hợp các mục đơn I∗, minutil và cấu trúc EUCS. TWU

Mục	Một	b	c	d	e	f
b	30					
c	65	61				
d	38	50	58			
e	57	61	88	50		
f	30	30	30	30	30	
g	27	11	38	0	38	0

Hình 5: Cấu trúc cùng xảy ra tiện ích ước tính

```
===== NỘI DUNG CỦA ESCS ===== Mục:1 -- 3
(TWU 65) 5 (TWU 57) Mục:2 -- 1 (TWU 30)
3 (TWU 61)                                          5 (TWU 61)

Mục:3 -- Mục:4 -- 1 (TWU
38)                              2 (TWU 50)      3 (TWU 58)      5 (TWU 50)
                                 l 88) Mục:6
                                                 3 (TWU 30)      4 (TWU 30)      5 (TWU 30)
                                                 3 (TWU 38)      5 (TWU 38)
```

Thuật toán 4: Thuật toán FHM

đầu vào : D: cơ sở dữ liệu giao dịch, minutil: đầu ra ngưỡng do người dùng chỉ định: tập hợp các tập mục tiện ích cao

1 Quét D để tính TWU của các mục riêng lẻ; 2 I∗ ← mỗi mục i sao cho TWU(i) ≥ minutil; 3 Gọi là tổng thứ tự các giá trị TWU tăng dần trên I∗; 4 Quét D để xây dựng danh sách tiện ích của từng mục i ∈ I∗ và xây dựng EUCS; 5 Xuất ra mỗi mục i ∈ I∗ sao cho SUM({i }.utilitylist.iutils) ≥ minutil; 6 FHMSearch (∅, I∗, minutil, EUCS);

```
c    3    3
d    2    2    3
e    2    3    4    2
f    1    1    1    1    1
g    1    1    2    0    2    0
```

```
===== NỘI DUNG ESCS ===== Mục: 1 -
- 3 (Hỗ trợ 3)                   5 (Hỗ trợ 2)
Mục:2 -- 1 (Hỗ trợ 1)           3 (Hỗ trợ 3)        5 (Hỗ trợ 3)
Mục:3 -- Mục:4 -- 1 (Hỗ trợ 2)
                                 2 (Hỗ trợ 2)        3 (Hỗ trợ 3)        5 (Hỗ trợ 2)
Mục:5 -- 3 (Hỗ trợ 4) Mục:6 -
- 1 (Hỗ trợ 1)                   2 (Hỗ trợ 1)        3 (Hỗ trợ 1)        4 (Hỗ trợ 1)        5
(Hỗ trợ 1) Mục:7 -- 1 (Hỗ trợ
1)                               2 (Hỗ trợ 1)        3 (Hỗ trợ 2)        5 (Hỗ trợ 2)
```

Item	a	b	c	d	e	f
b	30					
c	65	61				
d	38	50	58			
e	57	61	88	50		
f	30	30	30	30	30	
g	27	11	38	0	38	0

Thuật toán 5: Thủ tục FH MSearch

đầu vào : P: một tập mục, ExtensionsOfP: tập hợp các phần mở rộng của P, minutil: ngưỡng do người dùng chỉ định, EUCS: cấu trúc EUCS

đầu ra : tập các tập mục hữu ích cao 1
foreach tập mục Px ∈ ExtensionsOfP do

```
2       nếu SUM(Px.utilitylist.iutils)+SUM(Px.utilitylist.rutils) ≥ minutil thì
3           Phần mở rộngOfPx ← ∅;
4           foreach tập mục Py ∈ ExtensionsOfP sao cho yx làm
5               nếu ∃(x, y, c) ∈ EUCS sao cho c ≥ minutil thì
6                   Pxy ← Px ∪ Py;
7                   Pxy.utilitylist ← Cấu trúc (P, Px, Py);
8                   Tiện ích mở rộngOfPx ← Tiện ích mở rộngOfPx ∪ Pxy;
9                   nếu SUM(Pxy.utilitylist.iutils) ≥ minutil thì xuất Px;
                kết thúc
10
            kết thúc
11
12          FHMSearch (Px, ExtensionsOfPx, mỗi phút);
13      kết thúc
14 kết thúc
```

Thuật toán 6: Thủ tục xây dựng

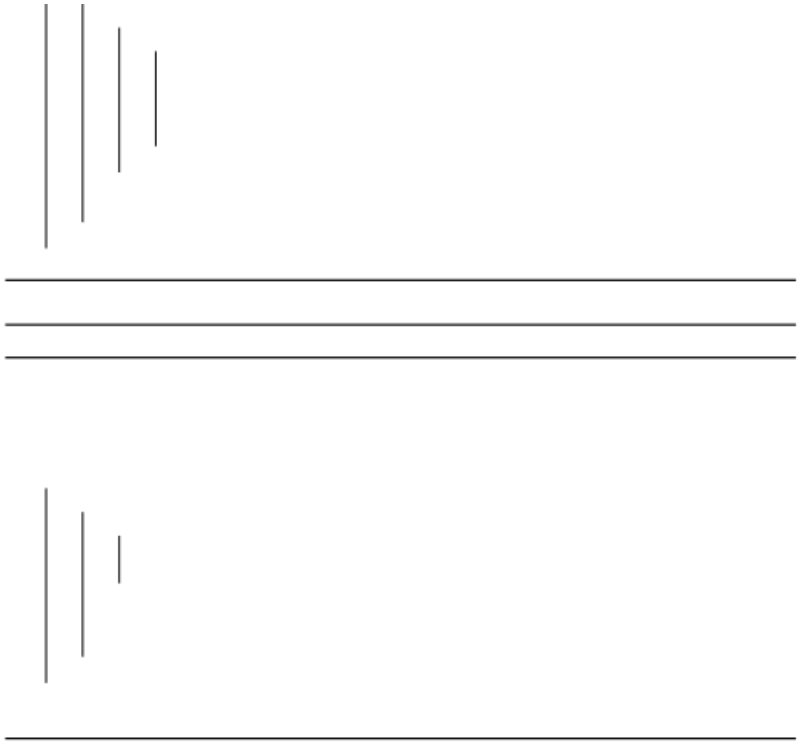
đầu vào : P: một tập mục, Px: phần mở rộng của P với một mục x, Py: phần mở rộng của P với một mục y Đầu ra: danh sách tiện ích của Pxy

1 Danh sách tiện íchO f Pxy ← ∅; 2
foreach tuple ex ∈ Px.utilitylist do

```
3       nếu ∃ey ∈ Py.utilitylist và ex.tid = exy.tid thì
4           nếu P.utilitylist , ∅ thì
5               Tìm phần tử e ∈ P.utilitylist sao cho e.tid = ex.tid.;
6               exy ← (ex.tid, ex.iutil + ey.iutil − e.iutil, ey.rutil);
            kết thúc
7
8           khác exy ← (ex.tid, ex.iutil + ey.iutil, ey.rutil);
9           Danh sách tiện íchO f Pxy ← Danh sách tiện íchO f Pxy ∪ {exy };
```

11 đầu 12 trả về
UtilityListPxy;

Thủ tục FH MSearch (Thuật toán 5) lấy đầu vào (1) tập mục P, (2) phần mở rộng của P có dạng Pz nghĩa là Pz có được trước đó bằng cách thêm một mục z vào P, (3) minutil và (4) EUCS. Quy trình tìm kiếm hoạt động như sau. Đối với mỗi phần mở rộng Px của P, nếu tổng các giá trị iutil của danh sách tiện ích của Px không nhỏ hơn minutil, thì Px là tập mục hữu ích cao và nó là đầu ra (xem Thuộc tính 4). Sau đó, nếu tổng giá trị iutil và rutil trong danh sách tiện ích của Px không nhỏ hơn minutil, điều đó có nghĩa là nên khám phá các phần mở rộng của Px. Điều này được thực hiện bằng cách hợp nhất Px với tất cả các phần mở rộng Py của P sao cho yx tạo thành các phần mở rộng có dạng Pxy chứa |Px| + 1 món đồ. Danh sách tiện ích của Pxy sau đó được xây dựng bằng cách gọi thủ tục Xây dựng để nối các danh sách tiện ích của P, Px và



Py. Sau đó, lệnh gọi đệ quy đến thủ tục Tìm kiếm bằng Pxy được thực hiện để tính toán tiện ích của nó và khám phá (các) phần mở rộng của nó. Vì thủ tục FH MSearch bắt đầu từ các mục đơn lẻ, nên nó khám phá đệ quy không gian tìm kiếm của các tập mục bằng cách nối thêm các mục đơn lẻ và nó chỉ cắt bớt không gian tìm kiếm dựa trên Thuộc tính 6. Có thể chứng minh rằng quy trình này là đúng và đầy đủ để khám phá tất cả các mục có giá trị cao. tập mục tiện ích.

Cấu trúc danh sách tiện ích được sử dụng bởi thuật toán FHM được cho là biểu diễn cơ sở dữ liệu theo chiều dọc. Biểu diễn cơ sở dữ liệu theo chiều dọc cho biết danh sách các giao dịch trong đó mỗi tập mục xuất hiện. Điều này khác với cơ sở dữ liệu theo chiều ngang truyền thống, trong đó mỗi mục nhập là một giao dịch cho biết các mục chứa trong đó.

Lợi ích của các thuật toán dựa trên danh sách tiện ích là chúng dễ thực hiện và hiệu quả. Nó đã chỉ ra rằng các thuật toán dựa trên danh sách tiện ích có thể nhanh hơn hai bậc so với các thuật toán hai pha [31, 58, 94]. Tuy nhiên, các thuật toán dựa trên danh sách tiện ích có những hạn chế quan trọng. Đầu tiên, các thuật toán này có thể khám phá một số tập mục không bao giờ xuất hiện trong cơ sở dữ liệu vì các tập mục được tạo bằng cách kết hợp các tập mục mà không cần đọc cơ sở dữ liệu. Do đó, các thuật toán này có thể lãng phí rất nhiều thời gian để xây dựng danh sách tiện ích của các tập mục không tồn tại. Thứ hai, các thuật toán này đôi khi tiêu tốn rất nhiều bộ nhớ, vì danh sách tiện ích phải được xây dựng cho mỗi tập mục được truy cập trong không gian tìm kiếm. Danh sách tiện ích của một tập mục có thể khá lớn. Trong trường hợp xấu nhất, nó chứa một bộ dữ liệu cho mỗi giao dịch của cơ sở dữ liệu. Thao tác nối cũng có thể đặc biệt tốn kém vì phải so sánh hai hoặc ba danh sách tiện ích để xây dựng danh sách tiện ích của mỗi k-itemset (k > 1).

Để giảm yêu cầu bộ nhớ của thuật toán dựa trên danh sách tiện ích, thuật toán ULB-Miner [17] gần đây đã được đề xuất bằng cách mở rộng HUI-Miner [58] và FHM [31]. ULB-Miner sử dụng bộ đệm để tái sử dụng bộ nhớ nhằm lưu trữ danh sách tiện ích. Chiến lược này đã được chứng minh là cải thiện cả thời gian chạy và mức sử dụng bộ nhớ. Một cải tiến khác của HUI-Miner là HUI-Miner* [71], dựa trên cấu trúc danh sách tiện ích* được cải tiến để tăng tốc HUI-Miner.

3.2.2 Thuật toán một pha tăng trưởng mẫu

Thuật toán một pha tăng trưởng mẫu giải quyết một số hạn chế của thuật toán dựa trên danh sách tiện ích. Họ khám phá không gian tìm kiếm bằng cách đọc cơ sở dữ liệu và do đó chỉ xem xét các tập mục tồn tại trong cơ sở dữ liệu. Thuật toán d2HUP [60] là thuật toán đầu tiên như vậy. Nó thực hiện tìm kiếm theo chiều sâu và biểu diễn cơ sở dữ liệu cũng như cơ sở dữ liệu được chiếu bằng cách sử dụng siêu cấu trúc, tương tự như thuật toán H-Mine [69] trong khai thác mẫu thường xuyên. Mặc dù thuật toán này được chứng minh là nhanh hơn một số thuật toán khác nhưng quá trình tạo và cập nhật siêu cấu trúc có thể khá tốn kém.

Gần đây, thuật toán EFIM đã được đề xuất [94], lấy cảm hứng từ thuật toán LCM trong khai thác tập phổ biến. Nó được thiết kế để xử lý từng tập mục trong không gian tìm kiếm theo không gian và thời gian tuyến tính. EFIM thực hiện tìm kiếm theo chiều sâu bằng cách sử dụng biểu diễn cơ sở dữ liệu theo chiều ngang để giảm mức sử dụng bộ nhớ. Hơn nữa, nó còn giới thiệu hai giới hạn trên mới được gọi là tiện ích cục bộ và tiện ích cây con để giảm không gian tìm kiếm một cách hiệu quả. EFIM cũng giới thiệu một công nghệ đếm tiện ích dựa trên mảng mới-

duy nhất có tên là Đếm tiện ích nhanh để tính toán các giới hạn trên này theo thời gian và không gian tuyến tính bằng cách sử dụng cấu trúc mảng có thể tái sử dụng. Hơn nữa, để giảm chi phí quét cơ sở dữ liệu, EFIM tích hợp các kỹ thuật kết hợp giao dịch và chiếu cơ sở dữ liệu hiệu quả có tên là Phép chiếu cơ sở dữ liệu tiện ích cao (HDP) và Hợp nhất giao dịch tiện ích cao (HTM), cũng được thực hiện trong thời gian tuyến tính. Nó đã chỉ ra rằng EFIM nói chung nhanh hơn hai đến ba bậc so với các thuật toán d2HUP, HUI-Miner, FHM và UPGrowth, trong khi thường tiêu thụ bộ nhớ thấp hơn nhiều.

3.3 So sánh các thuật toán khai thác tập mục tiện ích cao

Phần này đã cung cấp cái nhìn tổng quan về một số thuật toán khai thác tập mục có tính tiện ích cao phổ biến. Bảng 10 cung cấp sự so sánh các đặc điểm của chúng về loại tìm kiếm (tìm kiếm theo chiều rộng hoặc tìm kiếm theo chiều sâu), số lượng giai đoạn (một hoặc hai), biểu diễn cơ sở dữ liệu (ngang hoặc dọc) và tập mục phổ biến giống nhau nhất thuật toán khai thác.

Bảng 10: Các thuật toán khai thác tập mục có tính tiện ích cao

Thuật toán	Loại tìm kiếm	Nb của giai đoạn	đại diện cơ sở dữ liệu	mở rộng
Hai Pha [59]	chiều rộng đầu tiên	Hai	Nằm ngang	Tiên nghiệm [2]
PB [47]	chiều rộng đầu tiên	Hai	Nằm ngang	Tiên nghiệm [2]
IHUP [5]	chiều sâu đầu tiên	Hai	Ngang (cây tiền tố)	Tăng trưởng FP [39]
UPGrowth(+) [79]	theo chiều sâu	Hai	Ngang (cây tiền tố)	Tăng trưởng FP [39]
HUP-Tăng trưởng [52]	theo chiều sâu	Hai	Ngang (cây tiền tố)	Tăng trưởng FP [39]
MU-Tăng Trưởng [87]	chiều sâu đầu tiên	Hai	Ngang (cây tiền tố)	Tăng trưởng FP [39]
D2HUP [60]	chiều sâu đầu tiên	Một	Dọc (siêu cấu trúc)	H-Mỏ [69]
Công cụ khai thác HUI [58]	chiều sâu đầu tiên	Một	Dọc (danh sách tiện ích)	Sự rạng rỡ [91]
FHM [31]	chiều sâu đầu tiên	Một	Dọc (danh sách tiện ích)	Sự rạng rỡ [91]
mHUIMiner [70]	chiều sâu đầu tiên	Một	Dọc (danh sách tiện ích)	Sự rạng rỡ [91]

Công cụ khai thác HUI* [71]	chiều sâu đầu tiên	Một	Dọc (danh sách tiện ích*)	Sự rạng rỡ [91]
Công cụ khai thác ULB [17]	chiều sâu đầu tiên	Một	Dọc (danh sách tiện ích được đệm)	Sự rạng rỡ [91]
EFIM [94]	chiều sâu đầu tiên	Một	Ngang (có sự hợp nhất)	LCM [81]

4 phần mở rộng của vấn đề

Mặc dù việc khai thác tập mục có tiện ích cao có nhiều ứng dụng nhưng nó cũng có một số hạn chế đối với một số ứng dụng. Phần này trình bày tổng quan về các phần mở rộng của bài toán khai phá tập mục có tính tiện ích cao được thiết kế để giải quyết một số hạn chế này. Hầu hết các thuật toán cho các phần mở rộng này đều dựa trên các thuật toán được mô tả ở phần trước.