CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

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 30 tháng 01 năm 2016

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

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ Luận văn Thạc sĩ)

STT	Họ và tên	Chức danh Hội đồng
1		Chủ tịch
2		Phản biện 1
3		Phản biện 2
4		Ủy viên
5		Ủy viên, Thư ký

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

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

PHÒNG QLKH – ĐTSĐH **Độc lập – Tự do – Hạnh phúc**

TP. HCM, ngày 30 tháng 01 năm 2016

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên:	Phan Văn Bảo An	Giới tinh:	Nam
Ngày, tháng, năm sinh:	06/05/1983	Nơi sinh:	An Giang
Chuyên ngành:	Công nghệ thông tin	MSHV:	1441860001

I- Tên đề tài:

KHAI THÁC MẪU TRỌNG SỐ PHỔ BIẾN TỐI ĐẠI

TRONG CƠ SỞ DỮ LIỆU GIAO DỊCH

II- Nhiệm vụ và nội dung:

Đề tài nghiên cứu chỉ đơn giản là tập trung vào nghiên cứu các thuật toán khai thác các mẫu được đánh trọng số. Đề xuất ra thuật toán MWFIM của U.Yun và công sự, kết hợp sử dụng Diffset nhằm giảm thời gian khai thác và tiết kiệm bộ nhớ lưu trữ.

III- Ngày giao nhiệm vụ: 01/08/2015

IV- Ngày hoàn thành nhiệm vụ: 30/01/2016

V- Cán bộ hướng dẫn: *PGS.TS Lê Hoài Bắc*

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

(Họ tên và chữ ký) (Họ tên và chữ ký)

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 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 đã được cảm ơn và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

(Ký và ghi rõ họ tên)

Phan Văn Bảo An

LỜI CÁM ƠN

Trong cuộc sống của chúng ta không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác.

Trong suốt thời gian từ khi bắt đầu học tập ở trường Đại Học Công Nghệ Hutech đến nay, tôi đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô và bạn bè để tôi hoàn thành tốt khóa học. Tôi xin gửi đến quý Thầy Cô ở Phòng Quản Lý Khoa Học- Đào Tạo Sau Đại Học lời cám ơn chân thành và sâu nhất vì đã tổ chức giảng dạy cho chúng tôi được tiếp cận với các môn học mà theo tôi là rất hữu ích đối với sinh viên cao học ngành Ngành Công Nghệ Thông Tin cũng như tất cả các sinh viên thuộc các chuyên ngành nghề khác. Và đặc biệt, cho tôi cơ hội nghiên cứu Luận văn của mình xác với thực tế và tiếp cận thời đại.

Bên cạnh để có những kiến thức quí báu cho vận dụng vào việc nghiên cứu luận văn, tôi xin chân thành cảm ơn PGS.TS Lê Hoài Bắc một người Thầy đã đã tận tâm truyền đạt kiến thức, hướng dẫn tận tình để giúp tôi hoàn thành Luận văn của mình một cách tốt nhất.

Ngoài ra, sự thành công của luận văn tôi không thể không nhắc đến những người thân trong gia đình luôn luôn chia sẽ, động viên, giúp tôi có động lực vượt qua những thời điểm khó khăn nhất. Cuối cùng là cám ơn sự cố vũ nhiệt tình của bạn bè đã giúp tôi hoàn thành luận văn này.

(Họ và tên của tác giả Luận văn)

Phan Văn Bảo An

TÓM TẮT

Trong lĩnh vực khai thác dữ liệu, đã có nhiều nghiên cứu về khai thác mẫu phổ biến được ứng dụng thực tế rộng lớn trong các khai thác luật kết hợp, tương quan, mẫu tuần tự, ràng buộc mẫu phổ biến, mẫu đồ thị, mẫu mới nổi, nhiều công trình khai thác dữ liệu khác. Chúng tôi giới thiệu một thuật toán MWFIM[16] của U.Yun cho khai thác mẫu phổ biến tối đại từ một cơ sở dữ liệu giao dịch. Mẫu khai thác của U.Yun cắt tỉa mẫu không quan trọng và làm giảm kích thước của không gian tìm kiếm. Tuy nhiên, việc duy trì tính chất chống sự đơn điệu (anti-monotone) mà không mất mát thông tin cần được xem xét, do đó thuật toán của U.Yun cắt tỉa mẫu trọng số không phổ biến và sử dụng một tiền tố, cây có thứ tự trọng số giảm dần. Ngoài ra trong luận văn còn sử dụng kỹ thuật Diffsets nhằm khai thác nhanh độ hỗ trợ các items trong cơ sở dữ liệu giao dịch có mật độ trùng lắp cao nhằm giảm thời gian khai thác và tiết kiệm bộ nhớ.

ABSTRACT

In the field of data mining, there have been many studies on mining frequent patterns due to its broad applications in mining association rules, correlations, sequential patterns, constraint-based frequent patterns, graph patterns, emerging patterns, and many other data mining tasks. We propose a algorithm U.Yun 's MWFIM for mining maximal weighted frequent patterns from a transaction database. His mining paradigm prunes unimportant patterns and reduces the size of the search space. However, maintaining the antimonotone property without loss of information should be considered, and thus our algorithm prunes weighted infrequent patterns and uses a prefix-tree with weight-descending order. Besides, in dense database transaction, our algorithm used Diffset to reduce extraction time and save memory storage.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Ký hiệu và từ viết tắt	**Nghĩa tiếng Anh**	**Nghĩa tiếng Việt**
CSDL	Database	Cơ sở dữ liệu
DB	Database	Cơ sở dữ liệu
DBT	Database Transaction	Cơ sở dữ liệu giao dịch
Diffset	Different set	Tập khác Tidset
FP	Frequent Patern	Mẫu phổ biến
FP-TREE	Frequent Patern Tree	Thuật toán cây FP-TREE
FWI	Frequent weighted itemsets	Tập trọng số phổ biến
GD	Transaction	Giao dịch
I	The set of items	Tập hợp toàn bộ các thành phần trong cơ sở dữ liệu
IT	Itemset transaction	Tập giao dich
Itemset	The set of items	Tập thành phần
KDD	Knowledge Discovery in Database - Data Mining	Khám phá tri thức trong dữ liệu
KT	Datamining	Khai thác dự liệu
k-thành-phần	A set of k items	Tập k thành phần
L_i	Leyer i	Mức i
Min_sup	Threshold	Ngưỡng hỗ trợ
MWFIM	Maximal Weighted Frequent Itemset Mining	Thuật toán khai thác mẫu trọng số phổ biến tối đại
MWFP	Maximal weighted frequent patterns	Mẫu trọng số phổ biến tối đại
T	Transactions	Các giao dịch
Tidset	Transaction identity set	Tập giao dịch định danh

TW	Transaction weight	Giao dịch trọng số
WIT	Weight Itemset transaction	Tập giao dịch trọng số
WIT-TREE	Weight Itemset transaction tree	Thuật toán cây WIT-TREE
Ws	Weight support	Độ hỗ trợ trọng số

DANH MỤC CÁC HÌNH VÀ ĐỒ THỊ

DANH MỤC CÁC BẢNG

CHƯƠNG1: MỞ ĐẦU

1.1 Lý do chọn đề tài

Chúng ta đang sống trong thời đại thông tin. Nó giúp cho chúng ta dẫn đến thành công nhờ vào công nghệ hiện đại của máy tính. Chúng ta thu thập được một lượng thông tin rất lớn từ các cơ sở dữ liệu giao dịch như: Ngân hàng, bệnh viện, công ty, siêu thị,...

Do đó phân tích hành vi mua sắm của khách hàng là bài toán cơ bản trong lĩnh vực hoạt động cũng như nghiên cứu marketing. Trong đó, thông tin giao dịch thể hiện qua từng hóa đơn và các hóa đơn là dữ liệu quan trọng để các nhà buôn bán lẻ và sỉ rút ra được thông tin có giá trị về hành vi mua sắm của khách hàng để hoạch định chiến lược mua bán cũng như trong kế hoạch tích trữ tồn kho.

Tuy nhiên, để một người tự rút ra được những thông tin có giá trị từ nhiều dữ liệu hóa đơn thì thường dễ sai sót và tốn quá nhiều thời gian. Tác giả quan tâm đến việc hiện thực một hệ thống có tên gọi là Hệ phân tích hành vi khách hàng để hỗ trợ cho các nhà buôn bán lẻ tận dụng khai thác khối lượng dữ liệu giao dịch khổng lồ từ chính cửa hàng của họ.

Hệ thống gồm nhiều mô đun và nhiều giai đoạn để thực hiện, tất cả các mô đun và các giai đoạn thực hiện đều khó và phức tạp. Nhưng cốt lõi hơn hết là việc xử lý dữ liệu lớn hiệu quả. Vì vậy, tác giả ưu tiên tập trung nghiên cứu các thuật toán tìm các item có số lượng phổ biến tối đại từ đó cửa hàng tìm được các mặt hàng khách hàng quan tâm nhất để nhà quản lý có thể hoạch định kế hoạch mua bán cho mình.

1.2 Nội dung nghiên cứu

Đề tài tập trung vào nghiên cứu thuật toán khai thác mẫu trọng số phổ biến tối đại trên cơ sở dữ liệu giao dịch. Giới thiệu thuật toán MWFIM [16] khai thác mẫu trọng số tối đại phổ biến tối đại dựa trên cơ sở dữ liệu giao dịch có đánh trọng số. Nghiên cứu Diffset của Zaki nhằm ứng dụng vào khai thác mẫu tối đại tối đại.

1.3 Mục tiêu nghiên cứu

Mục tiêu tổng quát: Khảo sát các phương pháp làm thực nghiệm và phân tích thực nghiệm của các tác giả và đề xuất thuật toán mới mang tính tối ưu hơn thuật toán tác giả. Đánh giá thực nghiệm một số thuật toán khai thác mẫu trọng số phổ biến tối đại.

Mục tiêu cụ thể: Đưa ra các bước thực nghiệm cần thiết để đánh giá khách quan ưu điểm và khuyết điểm của các thuật toán MWFIM. Đánh giá thuật toán khai thác cải tiến mới. So sánh hiệu quả của tác giả và thuật toán cải tiến. Kiểm tra tính đúng đắn của mã nguồn các thuật toán khai thác mẫu trọng số phổ biến tối đại MWFIM so với mã giả của các thuật toán đưa ra trong các bài báo. Hiện thực lại các thực nghiệm cho từng thuật toán đã trình bày trong các bài báo đã công bố. Qua đó, đảm bảo môi trường thực nghiệm là hoàn toàn đáng tin cậy để so sánh và đánh giá với các kết quả mới sau này nếu có.

1.4 Đối tượng nghiên cứu

- Thuật toán khai mẫu trọng số phổ biến tối đại như: MWFIM của U.Yun.
- Khai thác độ hỗ trợ bằng kỹ thuật Diffsets.
- Dữ liệu mẫu như: Chess, mushroom, connect, MS1_itemset_mining.
- Nghiên cứu ngôn ngữ C#

1.5 Phạm vi nghiên cứu

Có nhiều khó khăn và hạn chế khách quan, nên giai đoạn này tác giả tìm hiểu các thuật toán khai thác mẫu trọng số phổ biến tối đại trên dữ liệu tĩnh (dữ liệu không có biến động), dữ liệu nghiên cứu được lấy từ nguồn dữ liệu nghiên cứu chuẩn (chưa thử nghiệm trên dữ liệu thực), việc đánh giá chỉ mới đánh giá dựa trên tốc độ xử lý dữ liệu của các thuật toán (chưa đánh giá tính có ích thực sự so với ý kiến thực của khách hàng). Việc xử lý dữ liệu theo hướng tập trung (chưa nghiên cứu hướng phân tán). Dữ liệu thực nghiệm được lấy từ nguồn đáng tin cậy: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

1.6 Phương pháp nghiên cứu

- Tiến hành thu thập và đọc các tài liệu có liên quan đến đề tài.
- Tìm hiểu các thuật toán hiện có để đánh giá các ưu, nhược điểm của từng thuật toán.
- Nghiêm cứu phương pháp khắc phục nhược điểm của thuật toán cũ

CHƯƠNG 2: TỔNG QUAN CÁC LĨNH VỰC NGHIÊN CỨU VÀ CƠ SỞ LÝ THUYẾT

2.1 Các khái niệm và định nghĩa

Khai thác dữ liệu là một công cụ giúp khai thác những thông tin hữu ích từ những kho dữ liệu được tích trữ trong suốt quá trình hoạt động của một công ty, tổ chức nào đó. Khai thác dữ liệu được dùng để mô tả quá trình tìm kiếm, chắt lọc và khai phá tri thức trong cơ sở dữ liệu hay chỉ việc tìm kiếm một tập hợp nhỏ có giá trị từ một số lượng lớn các dữ liệu thô. Quá trình này bao gồm tập hợp nhiều kỹ thuật được sử dụng trong tiến trình khám phá tri thức để tự động khai thác và chỉ ra sự khác biệt giữa các mối quan hệ và các mẫu chưa biết bên trong dữ liệu.

Khai thác luật kết hợp [2] là một phần quan trọng trong quá trình khám phá tri thức trong dữ liệu (KDD). Khai thác luật kết hợp được sử dụng để xác định mối quan hệ giữa các sản phẩm trong cơ sở dữ liệu GD và điều này dẫn đến việc nó chỉ quan tâm đến việc khách hàng có mua hay không mua sản phẩm nào đó. Thực tế, mỗi một sản phẩm có thể có giá trị khác nhau. Tương tự mỗi *item* trong cơ sở dữ liệu GD cũng có trọng số khác nhau tùy thuộc từng cơ sở dữ liệu cụ thể. Vì vậy việc khai thác trên loại dữ liệu này mang tính thực tiễn cao.

Năm 1998, Ramkumar, Ranka và Tsur [4] cũng như Cai, Fu, Cheng và Kwong [3] đã đề xuất một mô hình để mô tả các khái niệm về việc khai thác luật kết hợp có trọng số và dựa trên giải thuật Apriori để tìm ra các tập phổ biến được đánh trọng. Từ đó nhiều kỹ thuật khai thác luật kết hợp có trọng số được đề xuất như: Wang, Yang, Yu [6] và Tao, Murtagh, Farid [5].

2.1.1 Tổng quan về khai thác luật kết hợp

Trong lĩnh vực khai thác dữ liệu, mục đích của luật kết hợp (Association Rule - AR) [1] là tìm ra các mối quan hệ giữa các đối tượng trong khối lượng lớn dữ liệu. Nội dung cơ bản của luật kết hợp được tóm tắt như dưới đây.

Cho cơ sở dữ liệu gồm các GD T là tập các GD $T = \{t_1, t_2, \ldots, t_n\}$. Cho $I = \{i_1, i_2, \ldots, i_m\}$ là một tập các *item*. Mỗi tập con trong I được gọi một *itemset*, số lượng các phần tử trong một *itemset* được gọi là kích thước của một *itemset*. Mục đích của luật kết hợp là tìm ra sự kết hợp (association) hay tương quan (correlation) giữa các items.

Cho *X, Y* là các *itemset*, trong đó *X* và *Y* là hai tập không giao nhau khác rỗng. Một luật kết hợp được ký hiệu là $X \longrightarrow Y$, thể hiện mối ràng buộc của tập *Y* với tập *X* theo nghĩa là sự xuất hiện của tập *X* sẽ kéo theo sự xuất hiện của tập *Y* trong các GD, có thể hiểu rằng những người mua các item trong tập *X* cũng thường mua các item trong tập *Y*. Ví dụ, nếu *X* = {Táo, Chuối} và *Y* = {Anh Đào, Sầu Riêng} và ta có luật kết hợp $X \longrightarrow Y$, thì chúng ta có thể nói rằng những người mua Táo và Chuối thì cũng thường mua Anh Đào và Sầu Riêng.

Tập *X* được gọi là xuất hiện trong GD *t* nếu như nó là tập con của *t*. Độ hỗ trợ và độ tin cậy là hai tham số dùng để đo lường luật kết hợp. Thuật toán phổ biến nhất tìm các luật kết hợp là Apriori sử dụng các luật kết hợp nhị phân.

Định nghĩa 2.1

Độ hỗ trợ *(Min_sup)* của luật kết hợp $X \longrightarrow Y$ là tần suất GD chứa tất cả các *item* trong cả hai tập *X* và *Y*. Ví dụ: độ hỗ trợ của luật $X \longrightarrow Y$ là 40%, có nghĩa là 40% các GD *X* và *Y* được mua cùng nhau.

Công thức:

$$\text{Sup}\,(X \longrightarrow Y) = P\,(X \cup Y) = \frac{n(X \cup Y)}{N}$$

Trong đó $n(X \cup Y)$ là số GD có chứa cả X lẫn Y và N là tổng số GD trong CSDL.

Định nghĩa 2.2: Độ tin cậy (Conf) là xác suất xảy ra khi Y đã biết X.

Ví dụ: độ tin cậy của {Táo}⟶{Chuối} là 80% có nghĩa là 80% khách hàng mua {Táo} cũng mua {Chuối}.

Công thức:

$$\text{Conf}\ (X \longrightarrow Y) = P\ (X \mid Y) = \frac{n(X \cup Y)}{n(X)}$$

Để thu được các luật kết hợp, ta thường áp dụng 2 tiêu chí: Độ hỗ trợ tối thiểu min_sup và độ tin cậy tối thiểu min_conf là hai giá trị ngưỡng tối thiểu cho trước.

Luật kết hợp X ⟶ Y được coi là một mẫu có giá trị nếu xảy ra đồng thời

Sug (X ⟶ Y) ≥ min_sup và Con$f$(X ⟶ Y) ≥ min_con$f$.

Một tập X có độ hỗ trợ vượt quá ngưỡng min_sup được gọi là một tập phổ biến.

Định nghĩa 2.3: Lớp tương đương

Cho X⊆I, ta định nghĩa hàm p(X,k)= X[1,k] gồm k phần tử đầu của X và quan

hệ tương đương dựa vào tiền tố sau:

$$\forall X, Y \subseteq I, X \equiv \theta_k, Y \Leftrightarrow p(X, k) = p(Y, k)$$

Tập hợp tất cả itemset có cùng tiền tố là X gọi là lớp tương đương, được ký hiệu là [X].

Kết nối Galois

Cho quan hệ hai ngôi ð ⊆ I × Tchứa CSDL cần khai thác, trong đó I là tập các danh mục còn T là tập các giao tác. Đặt X ⊆ Ivà Y ⊆ T. Ta định nghĩa hai ánh xạ giữa P(I) và P(T) như sau:

$$t: I \longmapsto T, t(X) = \{y \in T | \forall\ x \in X, xðy\}$$

$$i : T \longmapsto I, i(Y) = \{x \in I | \forall\ y \in Y, xðy\ \}$$

Ánh xạ t(X) là tập hợp tất cả các GD có chứa X trong CSDL GD,

ánh xạ i(Y) là tập hợp tất cả các item chứa trong tất cả GD Y

Cho X, X1, X2 ∈P(I) và Y,Y1,Y2 ∈ P(T). Dựa theo Galois [9] ta có các tính chất sau:

(i). X1 ⊂ X2 ⇒ t(X1) ⊇ t(X2)

(ii). Y1 ⊂ Y2 ⇒ i(Y1) ⊇ i(Y2)

(iii). X ⊆ i(t(X))rà Y ⊆ t (i(Y))

2.1.2 Phương pháp Apriori

Khai thác tập phổ biến được đề xuất bởi Agrawal và các đồng sự năm 1993 [3] là một phương thức dành cho doanh nghiệp để phân tích giỏ hàng, nhằm mục đích tìm ra những quy luật trong việc mua sắm của khách hàng, siêu thị, v.v…

Thuật toán Apriori là thuật toán sinh ứng viên được đề xuất bởi Agrawal và Srikant vào năm 1994 [2]. Tư tưởng chính của thuật toán Apriori là:

Tìm ra tất cả các tập phổ biến có thể có trong cơ sở dữ liệu: k-*itemset* (tập danh mục gồm k phần tử) được dùng để tìm (k+1)-*itemset*

Đầu tiên tìm 1-itemset (ký hiệu L1). L1 được dùng để tìm L2 (2-itemset). L2 được dùng để tìm L3 (3-itemset) và tiếp tục cho đến khi không có k-itemset được tìm thấy.

Từ phổ biến sinh ra các luật kết hợp mạnh (các luật kết hợp thỏa mãn *min_sup* và *min_conf*)

Thuật toán Apriori dùng cách tiếp cận lặp được biết đến như tìm kiếm theo *mức*, với các tập có kích thước là *k* gọi là *k-itemset* được dùng để thăm dò các tập có kích thước *k+1* gọi là *(k+1)-itemset.*

Tính chất 2.1: *Mọi tập con của tập phổ biến đều phổ biến*, nghĩa là ∀ X ⊆ Y, có nghĩa là nếu Sup(Y) ≥ min_sup thì Sup(X) ≥ min_sup.

Tính chất 2.2: *Mọi tập cha của tập không phổ biến đều không phổ biến*, nghĩa là ∀ Y ⊇X, nếu Sup(X) < min_sup thì Sup(Y) < min_sup

* Các bước của giải thuật Apriori:

Bước 1: Tính độ hỗ trợ cho mỗi *item* có kích thước là 1, sau đó lọc ra các *item* thỏa mãn yêu cầu min_sup và đặt nó là tập *L1: 1-itemset* là tập kết quả tìm được. Chọn *L1* là tập hạt giống.

Bước 2: Bắt đầu từ tập hạt giống *1-itemset* là tập phổ biến có kích thước là 1 đã tìm được ở trên, phát sinh ra các tập phổ biến có kích thước là 2 gọi là các tập ứng viên (*C*) và tính độ hỗ trợ cho mỗi tập (*C*) này, từ đó chọn ra các tập phổ biến thỏa yêu cầu và đặt nó là tập L_2: *2-itemset* được dùng làm tập hạt giống cho bước kế tiếp.

Bước 3: Lặp lại bước 2, từ việc tiến hành chọn tập hạt giống có kích thước *l* là *l-itemset* để tìm ra các tập ứng viên có kích thước là *(l+1)-itemset*, quá trình này sẽ kết thúc khi không còn tìm được tập phổ biến nào thỏa yêu cầu *min_sup*.

Giải thuật Apriori:

Đầu vào: Tập các GD D, ngưỡng hỗ trợ tối thiểu cmin_sup

Đầu ra: L các tập phổ biến có trong D.

Phương thức:

```
Apriori()
{
    Gọi Ck Tập các ứng viên có kích thước k
    Lk Các tập phổ biến có kích thước k
```

```
        L1 = { các tập phổ biến có kích thước là 1 thỏa cmin_sup};
        for ( k = 1; Lk!= Ø; k++ )
        {
                Ck + 1= Apriori_gen(Fk) // Các ứng viên được tạo ra từ Fk
                for each t in D
                {
                        Ct = {c ∈ Ck +1|c ⊆ t}
                        for c ∈ Ct
                        {
                                c.count++
                        }
                        Fk +1 = {c ∈ Ck +1|c . count ≥ cmin_sup
                        }
                }
        return ⋃k Fk
}
```

Thuật toán Apriori sử dụng độ hỗ trợ tối thiểu dưới dạng số đếm (cmin_sup - min_sup count) để loại bỏ các ứng viên. Giá trị cmin_sup do người dùng đưa ra.

Hàm Apriori_gen có nhiệm vụ sinh ra các tập itemset có kích thước k + 1 từ tập hạt giống có kích thước là k trong tập Lk. Thủ tục này được thực thi bằng cách nối (join) các tập item có chung các tiền tố (prefix) và sau đó áp dụng tính chất 1.1 để loại bỏ các tập không thỏa mãn:

Bước nối: sinh ra các tập L_{k+1} là ứng viên của tập phổ biến có kích thước k+1

bằng cách kết hợp tập phổ biến Pk và Qk có kích thước k và trùng nhau ở k-1 tập đầu tiên. Ví dụ ta có: $L_{k+1} = P_k + Q_k = \{i_1, i_2, \ldots, i_{k-1}, i_k\}$ với Với $P_k = \{i_1, i_2, \ldots, i_{k-1}, i_k\}$ và $Q_k = \{i_1, i_2, \ldots, i_{k-1}, i_k\}$ trong đó $i_1 \leq i_2 \leq \ldots \leq i_{k-1} \leq i_k \leq i_k$.

Bước tỉa: Giữ lại tất cả các ứng viên L_{k+1} thỏa thỏa mãn tính chất Apriori (tính chất 1.1) tức là mọi tập con có kích thước k của nó đều là tập phổ biến ($\forall X \subseteq L_{k+1}$ và $|X| = k$ thì $X \in F_k$). Ta có cơ sở dữ liệu D gồm 4 GD với các tập item sau:

Bảng 2.1 Cơ sở dữ liệu các GD

Transaction	Item
1	**A, B, D, E**
2	**B, C, E**
3	**A, B, D, E**
4	**A, B, C, E**
5	**A, B, C, D, E**
6	**B, C, D**

Áp dụng giải thuật Apriori, cho CSDL GD D với ngưỡng *min_sup* = 0.4

Bước 1: Duyệt CSDL *D*để xác định độ hỗ trợ cho các tập phổ biến có kích thước là 1. Các tập *item* nào có độ hỗ trợ nhỏ hơn 40% sẽ bị loại bỏ. Sau lần duyệt đầu tiên thì tất cả các tập phổ biến đều thỏa ngưỡng *min_sup*, ta có:

$L_1 = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}\}$

Bảng 2.2 Apriori 1-itemset thỏa min_sup

Transactions	Items
1	A, B, D, E
2	B, C, E
3	A, B, D, E
4	A, B, C, E
5	A, B, C, D, E
6	B, C, D

1 - Itemset	Support
{A}	4
{B}	6
{C}	4
{D}	4
{E}	5

1 - Itemset	Support
{A}	4
{B}	6
{C}	4
{D}	4
{E}	5

Bước 2: Tạo ra các tập phổ biến có kích thước là 2 dựa trên các tập

phổ biến có kích thước là 1 thỏa điều kiện *min_sup* vừa tìm được, duyệt cơ sở dữ liệu *D* để xác định độ hỗ trợ cho từng tập phổ biến và loại bỏ tất cả các tập phổ biến có độ hỗ trợ nhỏ hơn *min_sup*. Sau khi duyệt ta có:

Bảng 2.3 Apriori 2-itemset thỏa min_sup

1 - Itemset	Support
{A}	4
{B}	6
{C}	4
{D}	4
{E}	5

→

2 - Itemset	Support
{A, B}	4
{A, C}	2
{A, D}	3
{A, E}	4
{B, C}	4
{B, D}	4
{B, E}	5
{C, D}	2
{C, E}	3
{D, E}	3

→

2 - Itemset	Support
{A, B}	4
{A, D}	3
{A, E}	4
{B, C}	4
{B, D}	4
{B, E}	5
{C, E}	3
{D, E}	3

Bước 3: Tạo ra các tập phổ biến có kích thước là 3 dựa trên các tập phổ biến có kích thước là 2 thỏa min_sup vừa tìm được, duyệt cơ sở dữ liệu D để xác định độ hỗ trợ cho từng tập phổ biến và loại bỏ tất cả các tập phổ biến có độ hỗ trợ nhỏ hơn min_sup.

Bảng 2.4 Apriori 3-itemset thỏa min_sup

2 - Itemset	Support
{A, B}	4
{A, D}	3
{A, E}	4
{B, C}	4
{B, D}	4
{B, E}	5
{C, E}	3
{D, E}	3

→

3 – Itemset	Support
{A, B, C}	2
{A, B, D}	3
{A, B, E}	4
{A, D, E}	3
{B, C, D}	2
{B, C, E}	3
{B, D, E}	3

→

3 – Itemset	Support
{A, B, D}	3
{A, B, E}	4
{A, D, E}	3
{B, C, E}	3
{B, D, E}	3

Bước 4: Tạo ra các tập phổ biến có kích thước là 4 dựa trên các tập phổ biến có kích thước là 3 vừa tìm được, duyệt cơ sở dữ liệu D để xác định độ hỗ trợ cho

từng tập phổ biến và loại bỏ tất cả các tập phổ biến có độ hỗ trợ nhỏ hơn min_sup.

Bảng 2.5 Apriori 4-itemset thỏa min_sup

3 – Itemset	Support
{A, B, D}	3
{A, B, E}	4
{A, D, E}	3
{B, C, E}	3
{B, D, E}	3

→

4 – Itemset	Support
{A, B, D, E}	3

Bước 5: Tổng hợp tất cả các tập phổ biến tìm được thỏa điều kiện ngưỡng hỗ trợ min_sup = 40%

2.1.3 Phương pháp IT-tree

Phương pháp IT-tree được Zaki đề xuất vào năm 1997 [8]. IT-Tree có cách tiếp cận đơn giản là dựa trên phần giao nhau của tập các giao tác để tính độ phổ biến và khái niệm mới lớp tương đương nhằm chia không gian xử lý ban đầu thành tập các không gian nhỏ độc lập giúp cho việc tìm kiếm nhanh hơn. Một điểm mới nữa của phương pháp IT-tree là dựa trên phần khác nhau trên Tidset của các tập dữ liệu nhằm làm giảm kích thước bộ nhớ yêu cầu và giúp cho việc tính độ phổ biến nhanh hơn.

Ứng dụng IT-Tree trong giai đoạn khai thác tập phổ biến:

Nhận xét: chỉ những itemset nào có tập giao tác khác rỗng thì mới có thể xuất hiện trong giao tác, lúc đó mới tính độ hỗ trợ và so sánh với min_sup. Còn lại những itemset có tập giao tác bằng rỗng thì không xuất hiện trong cơ sở dữ liệu đó. Ví dụ một tập các món hàng mà không được bán lần nào hết thì tập món hàng đó không xuất hiện trong hóa đơn bán hàng. Do vậy ta có thể tỉa bớt những tập đó trong quá trình sinh tập k-itemset từ tập (k-1)-itemset để giảm bớt không gian xử lí.

Để ứng dụng IT-Tree ta cần thay thế hàm Apriori_gen trong thuật toán Apriori

thành hàm IT_Tree, như trong hình 2.2, để sinh ra một k-itemset từ hai (k-1)-itemset ta cần phải xét đến hai yếu tố:

Hai (k-1)-itemset phải cùng tiền tố

Giao hai tập giao tác của hai (k-1)-itemset, nếu lực lượng của phần giao lớn hơn hay bằng min_sup thì k-itemset được thêm vào tập C_k.

Giải thuật Eclat

Bước 1: Đầu tiên duyệt cơ sở dữ liệu, khởi tạo lớp tương đương rỗng với tiền tố {} (hay [∅]) chứa tất cả các tập phổ biến có kích thước là 1 thỏa điều kiện ≥ min_sup.

Bước 2: Gọi hàm ENUMERATE_FREQUENT với đầu vào là lớp tương đương với tiền tố {}. Thủ tục này sẽ xét mỗi nút li∈ [P] với lj∈ [P] đứng sau nó, với mỗi một cặp (l_i,l_j), thủ tục này sẽ tính $Y=t(l_i \cup l_j) = t(l_i) \cap t(l_j)$, nếu$|Y| \geq$ min_sup nghĩa là độ hỗ trợ của li ∪ lj thỏa min_sup thì thêm nút X ×Y vào lớp tương đương [P_i].

Bước 3: Tiếp tục lặp lại bước 2 cho đến khi không tìm được lớp tương đương nào thỏa điều kiện min_sup.

Giải thuật Eclat

Đầu vào: cơ sở dữ liệu D với ngưỡng phổ biến cmin_sup

Đầu ra: IT-tree chứa tất cả các tập phổ biến của cơ sở dữ liệu

Phương **thức:**

Eclat( )	
1.	[∅] = i ∈ I ∧ o(i) ≥ cmin_sup}
2.	ENUMERATE_FREQUENT([∅])
	ENUMERATE_FREQUENT([P])

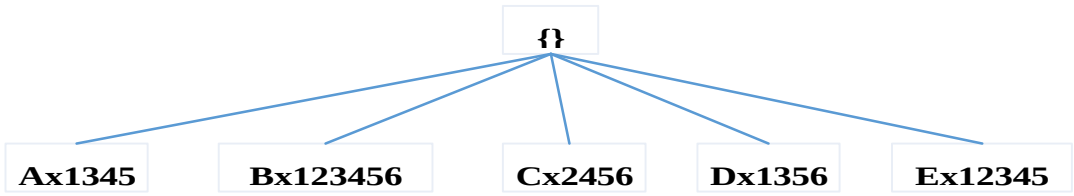
3.	For all $l_i \in [P]$do
4.	$[P_i]=\{ \emptyset\}$
5.	For all $l_j \in [P]$, with j > i do
6.	$X=l_i \cup l_j$
7.	$Y=t(l_i) \cap t(l_j)$
8.	If $\|Y\| \geq$ cmin_sup then
9.	$[P_i]= [P_i] \cup \{X \times Y\}$
10.	ENUMERATE_FREQUENT($[P_i]$)

Đầu tiên, thuật toán khởi tạo lớp tương đương rỗng ([∅]) chứa toàn bộ các nút có kích thước là 1 gọi 1-itemsets, chúng đều thỏa điều kiện ngưỡng min_sup.

Tất cả các nút ở mức 1 sẽ trở thành một lớp tương đương với tiền tố là [∅] (dòng 1). Sau đó hàm ENUMERATE_FREQUENT với biến đầu vào là lớp tương đương rỗng sẽ được thực thi. Thủ tục ENUMERATE_FREQUENT sẽ xét mỗi nút li∈ [P] (dòng 3) với nút lj∈ [P] (dòng 5) đứng sau nó, với mỗi cặp (li,lj), thủ tục này sẽ tính $Y= t(l_i \cup l_j)= t(l_i) \cap t(l_j)$ (dòng 7), nếu $|Y| \geq$ cmin_sup nghĩa là độ hỗ trợ theo số đếm của $l_i \cup l_j$ thỏa cmin_sup thì thêm nút $X \times Y$ vào lớp tương đương $[P_i]$ (được khởi tạo rỗng ở dòng 4). Sau đó, gọi đệ qui thủ tục ENUMERATE_FREQUENT để sinh ra các lớp tương đương con cho đến khi không còn lớp tương đương nào được tạo ra (dòng 10).

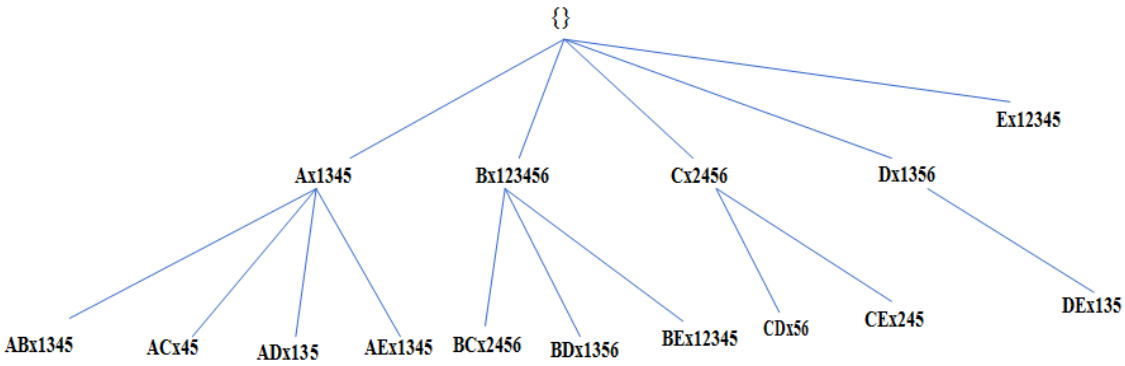
Sử dụng cơ sở dữ liệu D (Bảng 2.1.2.1) để tiến hành khai thác các tập phổ biến có độ hỗ trợ thỏa min_sup = 0.4.

Bước 1: Khởi tạo lớp [∅] chứa tất cả các tập phổ biến có kích thước là 1



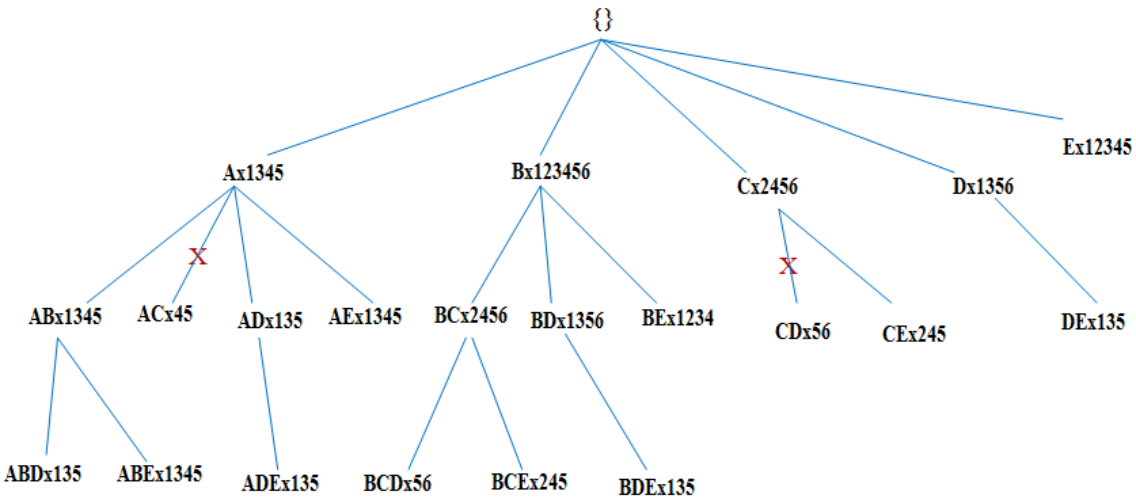
Hình 2.1 Khởi tạo lớp tương đương rỗng trên cây IT-tree

Bước 2: Với mỗi một nút ở mức 1, ta tiến hành khởi tạo các lớp tương đương của các tập phổ biến có kích thước là 2, loại bỏ các tập không thỏa điều kiện *min_sup*.



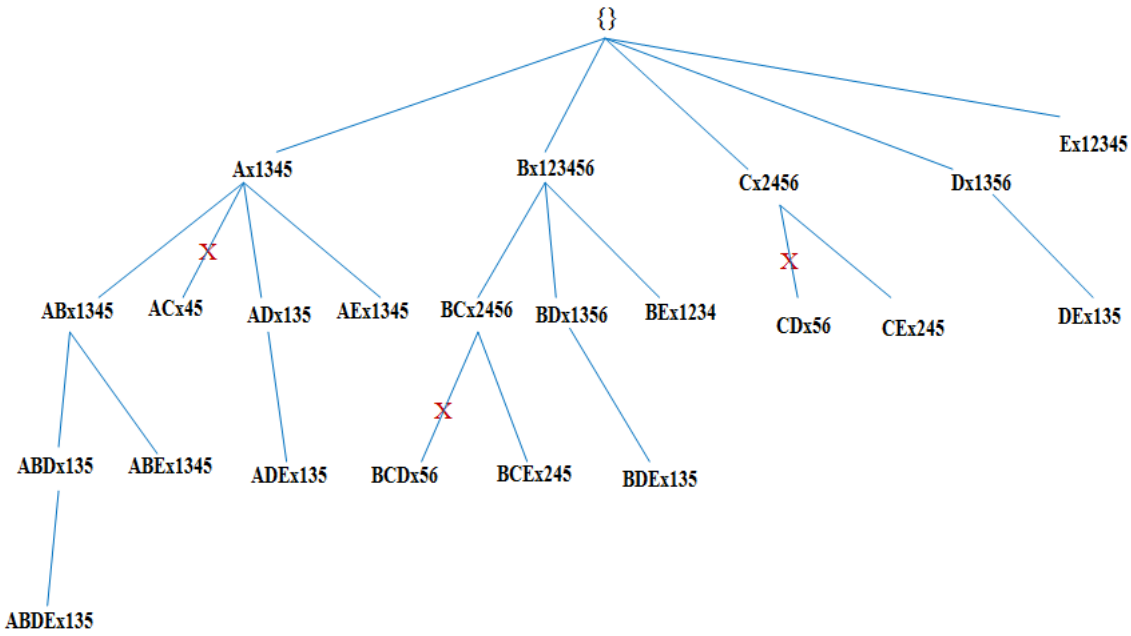
Hình 2.2 Cây IT-tree với lớp tương đương ở mức 2

Bước 3: tiến hành lặp lại bước 2, khởi tạo các lớp tương đương cho các tập phổ biến có kích thước là 3, với tiền tố là từ các tập ở lớp tương đương mức 2, loại bỏ các tập không thỏa điều kiện *min_sup*.



Hình 2.3 Cây IT-tree với lớp tương đương ở mức

Bước 4: tạo ra các tập phổ biến có kích thước là 4, từ các tập ở mức 3, tiến hành loại bỏ các tập không thỏa điều kiện *min_sup*



Hình 2.4 Cây IT-tree với lớp tương đương ở mức 4

Giải thuật dừng lại.

Bước 5: Không thể phát sinh ra được các ứng viên nào thỏa điều kiện *cmin_sup*. Kết thúc giải thuật ta thu được tập F gồm các tập phổ biến thoả điều kiện minsup F = {{A}, {B}, {C}, {D}, {E}, {AB}, {AD}, {AE}, {BC }, {BD}, {BE}, {CE}} , {DE}, {ABD}, {ABE }, {ADE}, {BCE}, {BDE}, {ABDE}

2.1.4 Phương pháp FP-tree

Nhược điểm của thuật toán Apriori là phát sinh ra quá nhiều ứng viên và việc duyệt cơ sở dữ liệu nhiều lần dẫn tới phát sinh chi phí lớn khi kích thước của các tập phổ biến tăng lên. Giải thuật khai thác tập phổ biến dựa trên cấu trúc cây FP-tree được Han và các đồng sự giới thiệu vào năm 2000 [7] đã khắc phục được nhược điểm này của thuật toán Apriori:

Sử dụng cấu trúc dữ liệu nén dữ liệu từ tập dữ liệu vào cây FP-tree:

Giảm chi phí cho toàn tập dữ liệu dùng trong quá trình khai thác do những phần tử không phổ biến (không thỏa mãn điều kiện minsup) đã bị loại bỏ ngay từ đầu.

Tránh tạo tập dự tuyển mỗi lần kiểm tra một tập phần tử

Cấu trúc này cho phép thực hiện việc tìm kiếm theo chiều sâu và áp dụng mô hình chia để trị khá hiệu quả do quá trình khai thác được chia thành các tác vụ nhỏ:

Xây dựng cây FP-tree

Khai thác tập phổ biến với FP-tree

Cây FP được xây dựng theo các bước sau:

Bước 1: Duyệt CSDL, lấy ra tập các item phổ biến F và tính độ phổ biến của chúng.

Sắp xếp các item trong tập F theo thứ tự giảm dần của độ phổ biến, ta được tập kết quả là L.

Bước 2: Tạo nút gốc cho cây T, tên của nút gốc sẽ là Null. Sau đó duyệt CSDL lần thứ hai.

Ứng với mỗi giao tác trong CSDL thực hiện 2 công việc sau:

Chọn các item phổ biến trong các giao tác và sắp xếp chúng theo thứ tự giảm dần độ phổ biến trong tập L

Gọi hàm Insert_Tree(P,Root) để đưa các item vào trong cây T

Thủ tục con Insert_Tree được định nghĩa như sau:

```
Insert_Tree(P, R)
{
Đặt P=[p|P – p], với p là phần tử đầu và P – p là phần còn lại của tập hợp;
```

```
If R có một con N sao cho N.item-name = pthen N.count ++;
else {
Tạo nút mới N; N.count = 1; N.item-name = p;
N. parent = R;
// Tạo node-link chỉ đến item, H là bảng Header N.node-link = H[p].head;
H[p].head = N;
}
// Tăng biến count của p trong bảng header thêm 1
H[p].count ++;
If (P – p) != null then Call Insert_Tree(P – p, N) ;
}
```

Để khai thác các các tập phổ biến từ FP-tree, ta sử dụng thủ tục FP-Growth. Phương pháp này là phương pháp duyệt cây theo chiều sâu và dựa vào mô hình chia để trị.

FP-Growth(*Tree*, α)

{

F = \$;

If *Tree* chỉ chứa một đường dẫn đơn P **then**

{

For each tổ hợp β của các nút trong P **do**

{

Phát sinh mẫu $p = \beta \cup \alpha$;

support_count(p) = độ hỗ trợ nhỏ nhất của các nút trong *β;*

F = F ∪ p;

}

}

else{

For each a_i trong bảng header của Tree

{

Phát sinh mẫu *β = a_i∪α*; *support_count(β)=a_i.support_count;*

F = F ∪ β;

Xây dựng cơ sở có điều kiện của *β;*

Xây dựng FP-tree có điều kiện Treeβ của β;

}

Dựa vào cơ sở dữ liệu *D1* ta tiến hành khai thác tìm ra các tập phổ biến, với ngưỡng hỗ trợ *minsup = 0.6*

Bảng 2.6 Cơ sở dữ liệu các GD D1

Transaction	Item
1	**f, a, c, d, g, i, m,p**
2	**a, b, c, f, l, m, o**
3	**b, f, h, j, o**
4	**b, c, k, s, p**
5	**a, f, c, e, l, p, m, n**

Bước 1: Duyệt cơ sở dữ liệu tìm ra tất cả các tập phổ biến có kích thước là 1.

Bảng 2.7 Độ hỗ trợ của các item trong cơ sở dữ liệu D1

Item	a	B	c	d	e	f	g	i	j	l	k	m	n	o	p	s
Sup	3	3	4	1	1	4	1	1	1	2	1	3	1	2	3	1

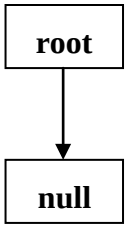
Sắp xếp chúng theo danh sách với trật tự giảm dần theo tần số xuất hiện. Loại bỏ các tập phổ biến có độ hỗ trợ không thỏa *minsup* ta có danh sách. Ta có một danh sách các item phổ biến: *L*= {f (4), c (4), a (3), b (3), m (3), p (3)}

Bước 2: Xây dựng cây FP qua từng bước. Các *item* trong mỗi GD được xử lý theo trật tự trong *L*.

Bảng 2.8 Các item phổ biến thỏa mãn Min_sup trong CSDL D1

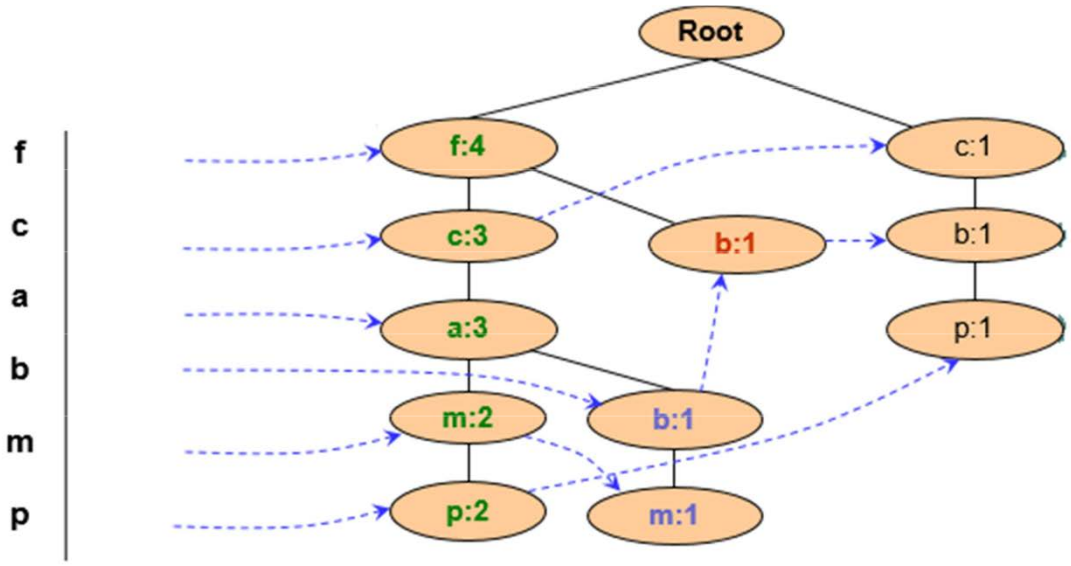
TID	*Các item được mua*	*Các item phổ biến thỏa mãn Min_sup (đã sắp theo thứ tự)*
1	f, a, c, d, g, i, m,p	f, c, a, m, p
2	a, b, c, f, l, m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

Khởi tạo FP-tree với nút gốc là *root* gán nhãn "*null*"



Hình 2.5 Khởi tạo nút root trên cây FP

Xét GD và thêm vào cây ta được cây FP-tree như sau:



Hình 2.6 Cây FP hoàn chỉnh

2.2 Tổng quan về khai thác luật kết hợp trên CSDL được đánh trọng số

2.2.1 Định nghĩa và tính chất của tập được đánh trọng số

Cơ sở dữ liệu GD của tập được đánh trọng số bao gồm: một tập hợp các GD T = $\{t_1,t_2,\ldots,t_m\}$; một tập các item I = $\{i_1,i_2,\ldots,i_n\}$ và một tập hợp các trọng số W = $\{w_1,w_2,\ldots,w_n\}$ tương ứng với mỗi một item có trong I.

Bảng 2.9 CSDL các giao dịch D

Transaction	Item
1	A, B, D, E
2	B, C, E
3	A, B, D, E
4	A, B, C, E
5	A, B, C, D, E
6	B, C, D

Bảng 2.10 Trọng số các giao dịch CSDL D

Item	Weight
A	**0.6**
B	**0.1**
C	**0.3**
D	**0.9**
E	**0.2**

Định nghĩa 2.1: Trọng số GD của một GD (transaction weight – t_w)t_k được gọi là tw và được nghĩa là tỉ số của tổng các trọng số của các item được mua chia cho số item:

$$tw(t_k) = \frac{\sum_{ij \in tk} wj}{|tk|}$$

Khai thác tập được đánh trọng phổ biến quan tâm đến trọng số (weighted hay benefit) của các item và chưa quan tâm đến số lượng mua. Dựa vào CSDL trong Bảng 2.1.2.1 và Bảng 2.1.4.1 và định nghĩa 2.1 ta tính được trọng số GD của GD T_1 như sau:

Ta có T1 = {A,B,D,E} tương ứng với WA= 0.6, WB= 0.1, WD= 0.9, WE = 0.2 suy ra ta có tw(t_1)được tính như sau:

$$tw(t_1) = \frac{Wa+Wb+Wc+We}{4} = \frac{0.7+0.1+0.9+0.2}{4} = 0.4$$

Tương tự với các GD khác ta có:

Bảng 2.11 Trọng số GD của các GD có trong D

Transation	tw
1	**0.45**
2	**0.2**

3	**0.45**
4	**0.3**
5	**0.42**
6	**0.43**
Sum	**2.25**

Định nghĩa 2.2: Trọng số hỗ trợ được ký hiệu là *ws* của một tập phổ biến được định nghĩa như sau:

$$ws(A) = \frac{tw(t1) + tw(t3) + tw(t4) + tw(t5)}{4} = \frac{0.45 + 0.45 + 0.3 + 0.42}{4} = 0.72$$

Bảng 2.12 Bảng trọng số hỗ hợ cho tập phổ biến 1 phần tử

X	Weighted support (ws)
A	**0.72**
B	**1**
C	**0.6**
D	**0.78**
E	**0.81**

Định lý 2.1: *"Mọi tập con khác rỗng của tập phổ biến cũng là tập phổ biến và mọi tập chứa tập không phổ biến đều là tập không phổ biến"* có nghĩa là nếu cho $X \subset Y$ khi đó $ws(X) \geq ws(Y)$.

2.2.2 Thuật toán khai thác dựa trên WIT-tree[9]

Cấu trúc WIT-tree

Để khai thác các luật kết hợp có trọng số, đầu tiên chúng ta phải tìm tất cả các tập được đánh trọng số thỏa điều kiện ngưỡng trọng số tối thiểu

min_ws. Việc khai thác các tập được đánh trọng số được xem là quá trình quan trọng nhất trong việc khai thác các luật kết hợp có trọng số. Ramkumar và các đồng sự [4] đã trình bày giải thuật khai thác các tập được đánh trọng số dựa trên mô hình thuật toán Apriori.

Nhược điểm chính của các giải thuật dựa trên thuật toán Apriori là việc phải duyệt cơ sở dữ liệu nhiều lần để tìm ra các tập phổ biến, dẫn đến việc sẽ phát sinh chi phí lớn.

FWI khai thác trên CSDL GD có quan tâm đến trọng số của các mục.

Năm 2003, Tao và các đồng sự đề xuất phương pháp khai thác WAR (Weighted Association Rule) [11]. Thuật toán được đề nghị sử dụng một biến thể của thuật toán Apriori cho khai thác các FWI. Năm 2013, Vo và các đồng sự đề xuất một phương pháp khai thác nhanh FWI sử dụng WIT-tree và phát triển các tính chất trên WIT- tree để tính nhanh ws của các itemset.

Giải thuật khai thác các tập được đánh trọng số dựa trên cấu trúc WIT-tree (Weighted *Item*set-Tidset), cấu trúc này là phần mở rộng dựa trên cấu trúc cây IT- tree đã được trình bày ở mục 1.2.3. Mỗi một node trên WIT-tree bao gồm 3 thành phần:

- X: đại diện cho một tập phổ biến
- t(X):Tidset tập hợp các GD có chứa X
- *ws:* trọng số hỗ trợ của *X*

Mỗi một nút được ký hiệu như là một bộ ba <X, t(X), ws>.

Giá trị của mỗi một nút được tính dựa theo công thức trọng số hỗ trợ (định nghĩa 2.2). Việc tính toán các trọng số hỗ trợ dựa trên các *Tidset*. Các liên kết kết nối các nút ở mức thứ *k* (gọi là *X*) với các các mức thứ *k+1* (gọi là *Y*).

Nút gốc *"root"* của cây WIT-tree chứa tất cả các nút có kích thước là 1 gọi là *1-itemset*. Tất cả các nút ở mức 1 sẽ trở thành lớp tương đương với tiền tố là {}

(hay [∅]). Mỗi một nút trong mức 1 sẽ trở thành một lớp tương đương mới, sử dụng các *item* này như là một tiền tố. Với mỗi một nút có chung tiền tố, nó sẽ kết hợp với các nút phía sau nó để tạo ra các lớp tương đương mới. Quá trình này sẽ được thực hiện đệ qui để tìm ra các lớp tương đương ở các mức cao hơn.

Giải thuật WIT-FWI Mô tả giải thuật:

Bước 1: Cho tập L_r chứa tất các các tập được đánh trọng có kích thước là 1 và trọng số hỗ trợ của chúng thỏa điều kiện ngưỡng trọng số hỗ trợ tối thiểu *min_ws* (dòng 1).

Bước 2: Các nút chứa trong L_r được sắp xếp theo thứ tự tăng dần dựa trên trọng số hỗ trợ (dòng 2).

Bước 3: Khởi tạo tập *FWI* và gán nhãn *"null"* (dòng 3).

Bước 4: Gọi hàm **FWI-EXTEND** để khai thác các tập được đánh trọng (dòng 4). Hàm **FWI-EXTEND**: sẽ xem xét mỗi một nút l_i có trong L_r với các nút phía sau nó để tạo ra một tập những nút mới là L_i (dòng 5 và 7). Để tạo ra L_i: đầu tiên, cho $X = l_i.itemset \cup l_j.itemset$ và tính toán $Y = t(X) = t(l_i) \cap t(l_j)$ (dòng 8). Nếu $ws(X)$ (được tính toán thông qua $t(X)$), (dòng 9) thỏa điều kiện *min_ws* (dòng 10). Nút mới tạo ra sẽ được thêm vào trong tập L_i (dòng 11). Sau khi tạo thành L_i, hàm **FWI- EXTEND** sẽ được gọi đệ qui với biến đầu vào là L_i (dòng 13) nếu số lượng các nút có trong tập L_i lớn hơn 1. Hàm **COMPUTE-WS*(Y)*** được sử dụng để tính trọng số hỗ trợ của tập X dựa trên giá trị trong Bảng 2.2 với $Y = t(X)$ (dòng 12).

Đầu vào: cơ sở dữ liệu D và min_ws (ngưỡng trọng số hỗ trợ tối thiểu)

Đầu ra: tập FWI chứa các tập phổ biến được đánh trọng thỏa ngưỡng min_ws

Phương thức:

WIT-FWI ()

{

1. L_r = tất cả các item mà ws thỏa min_ws

2. Sắp các nút trong L_r tăng dần theo ws

3. Khởi tạo tập FWI = ∅

4. Gọi hàm FWI-EXTEND với tham số là Lr

FWI-EXTEND(L_r)

5. Cho mỗi nút l_i trong L_r

6. Chèn (l_i.itemset, l_i.ws) vào trong FWI

7. Tạo một tập Li bằng cách nối livới tất cả l_j theo sau trong L_r:

8. Tập X = l_i.itemset l_j.itemset và Y = t(l_i) t(l_j)

9. ws(X) = COMPUTE-WS(Y)

10. Nếu ws(X) thỏa min_ws khi đó

11. Chèn một nút vào trong L_i

12. Nếu số lượng nút trong L_i 2 khi đó

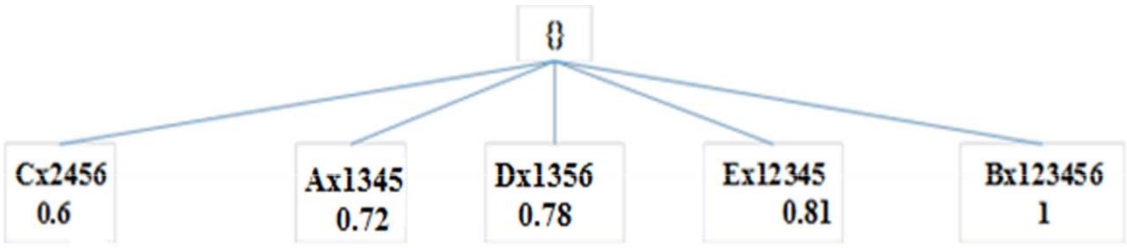
13. Gọi đệ qui hàm FWI-EXTEND với biến Li

Sau đây ta sẽ tiến hành khai thác các tập được đánh trọng phổ biến dựa trên Bảng 2.2.1.3 và Bảng 2.2.1.4 với ngưỡng trọng số hỗ trợ tối thiểu *min_ws* = 0.4

Bước 1: Ta tiến hành tính toán trọng số hỗ trợ của các tập có kích thước là 1 và khởi tạo tập L_r. Ta có *ws(A) = 0.72, ws(B)= 1, ws(C) = 0.6,ws(D) = 0.78, ws(E) = 0.81*.

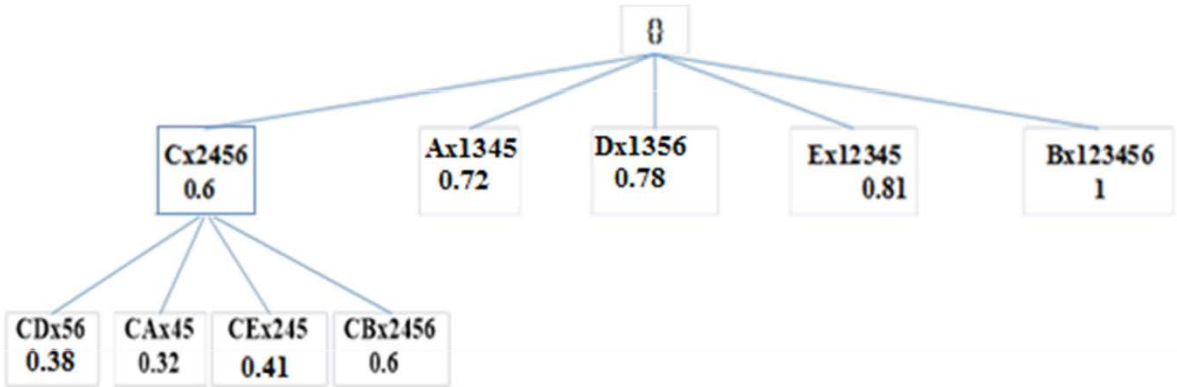
Tất cả các giá trị của các tập này đều thỏa điều kiện min_ws. Suy ra ta có tập: L_r= Sau đó ta tiến hành sắp xếp Lr theo thứ tự tăng dần của trọng số hỗ trợ, chúng ta có:

Ta tiến hành khởi tạo lớp {} chứa các tập có kích thước là 1.

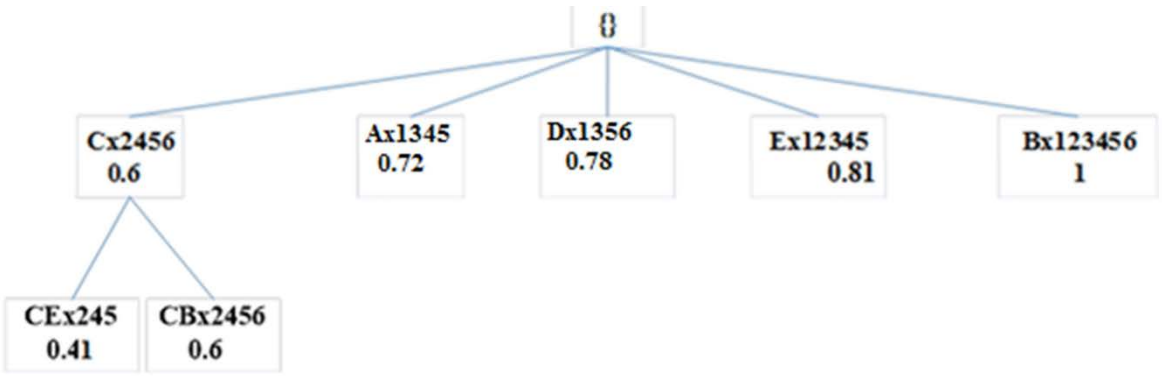


Hình 2.6 Khởi tạo lớp tương đương rỗng cho WIT-tree

Bước 2: Tiến hành tạo ra lớp tương đương mới dựa trên lớp tương đương cũ. Ví dụ ta có *C* sẽ nối với *D*, ta có một tập được trọng mới *CD* với *t(CD)* = 56 và ws(CD) = 0.38, vì ws(CD) không thỏa min_ws, nên không được phép thêm vào tập LC. Ta tiếp tục tiến hành ghép CA, với t(CA) = 45 và ws(CA) = 0.32 cũng không thỏa min_ws. Tiếp tục ghép C với E ta có CE với t(CE) = 245, ws(CE) = 0.41 thỏa min_ws. Ta thêm nút CE vào trong tập L_C suy ra L_C = . Kết quả cuối cùng sau khi đã tiến hành ghép nút với các nút còn lại sau nó ta có tập L_C =

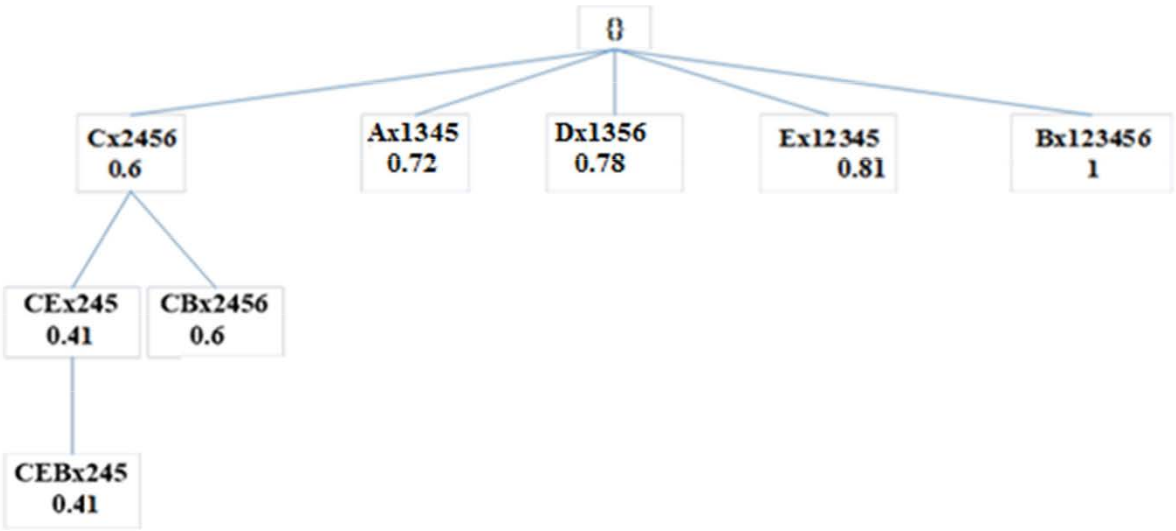


Hình 2.7 Cây WIT-tree với tập Lc



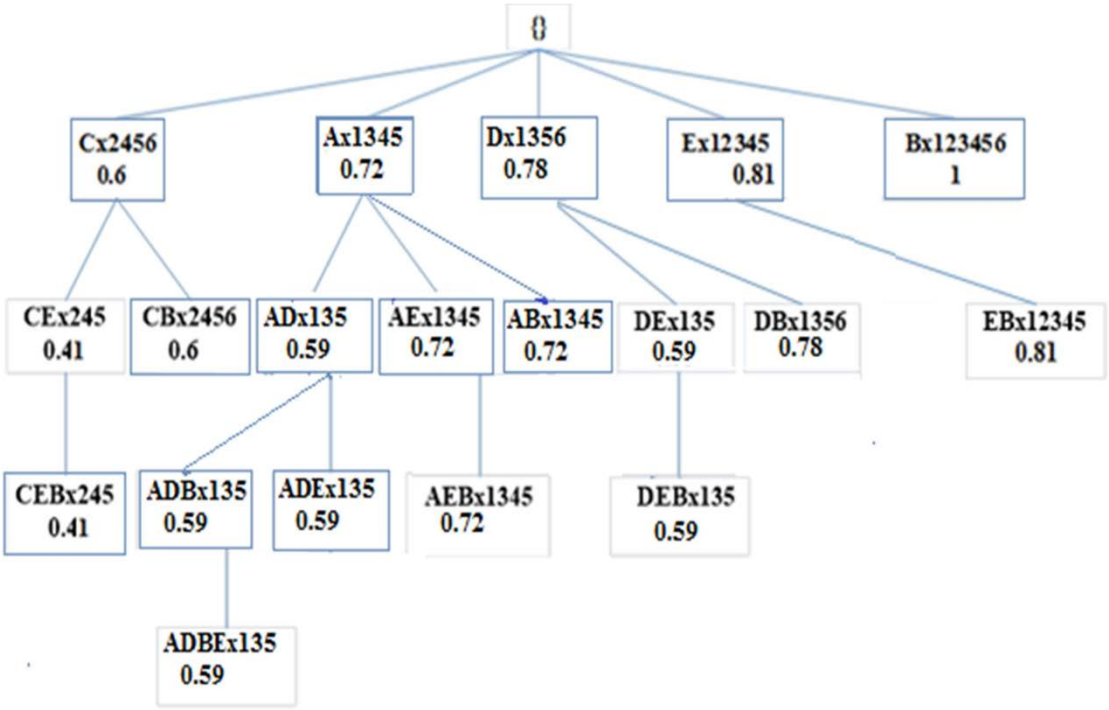
Hình 2.8 Cây WIT-tree sau khi tiến hành tỉa các tập không thỏa min_ws

Bước 3: Sau khi tạo ra tập L_C, bởi vì số lượng các nút trong L_C nhiều hơn 1, nên ta tiến hành gọi đệ qui hàm **FWI–EXTEND**để tiến hành tạo ra các nút con của tập L_C. Ta ghép *CE* với *CB* ta có *t(CEB)* = 245, *ws(CEB*) = 0.41, ta thêm tập *CEB* vào trong tập L_{CE} =



Hình 2.9 Cây WIT-tree với tập L_{CE}

Bước 4: Ta tiến hành thực hiện tiếp đối với các nút còn lại, để tìm ra tất cả các tập *FWI* thỏa điều kiện *min_ws*.



Hình 2.10 Cây WIT-tree hoàn chỉnh với min_ws = 0.4

Kết quả ta tìm được tập *FWI={{C}, {CE},{CB},{CEB},{A},{AD}, {ADB},*

{ADBE}, {ADE}, {AE}, {AEB},{AB},{D},{DA},{DE}, {DEB},{DB}, {E},{EB},{B}}

2.3 Khai thác mẫu phổ biến tối đại MFP

Khái niệm: Một mẫu X được gọi là mẫu phổ biến tối đại (MFP) nếu X phổ biến và không tồn tại tập nào bao nó phổ biến.

Bảng 2.13 Mình họa tập phổ biến tối đại

Tid	Items
10	**A,B,C,D,E**
20	**B,C,D,E,**

30	**A,C,D,F**

- {B, C, D, E}, {A, C, D} -tập phổ biến tối đại
- {B, C, D} - không phải tập phổ biến tối đại

Với Min_sup=2

Độ hỗ trợ của tất cả các mẫu phổ biến cần thiết cho việc rút ra cho luật kết hợp. Tất nhiên, nó thường không áp dụng để tạo nên toàn bộ mẫu phổ biến hoặc mẫu bao đóng trong khi có nhiều mẫu rất dài trong dữ liệu. Bộ của mẫu phổ biến tối đại là nhỏ nhất có thể có trong cách diễn tả của Data mà vẫn có thể sử dụng để trích xuất một bộ mẫu phổ biến.

Sau khi FI được tạo ra, các thông tin độ hỗ trợ có thể dễ dàng tính toán lại từ các cơ sở dữ liệu GD. Burdick đề xuất một thuật toán khai thác MFP, MAFIA (các thuật toán mẫu phổ biến tối đại). Thuật toán này sử dụng một đại diện bitmap thẳng đứng, nơi mà các số của một mẫu dựa trên các cột trong bitmap. Ngược lại, thuật toán FPmax sử dụng một định dạng ngang. Nó cũng là một thuật toán MFP khai thác sâu đầu tiên, nhưng nó sử dụng một cấu trúc FP-Tree cách tiếp cận các tăng trưởng của mẫu. Để tìm ra mẫu phổ biến tối đại từ đó ta sẽ tìm ra tất cả các mẫu phổ biến từ mẫu phổ biến tối đại được tìm thấy.

CHƯƠNG 3: KHAI THÁC MẪU PHỔ BIẾN TRỌNG SỐ TỐI ĐẠI TRONG CSDL GIAO DỊCH

3.1 Tổng quát khai thác tập phổ biến trọng số tối đại

Tập trung chính vào khái thác mẫu phổ biến trọng số (WFP) là quan tâm đến tính chất anti-monotone.

Tính chất anti-monotone. Nếu i_{q1}, i_{q2}, i_{q3}, ..., i_{qk} là k-items phổ biến thì mọi tập con của nó cũng là phổ biến.

Từ tính chất này thông thường bị phá vỡ khi trọng số khác nhau ứng dụng đến các items và các mẫu. Nghĩa là, Mặc dù mẫu X có trọng số là không phổ biến, những siêu mẫu của X là phổ biến. Khai thác luật kết hợp với item có trọng số được định nghĩa là độ hỡ trợ trọng số, mà nó được tính bởi phép nhân độ hỗ trợ của một mẫu với trọng số trung bình của một mẫu. Trong khai thác luật kết hợp có trọng số (WARM), vấn đề phá vỡ tính chất anti-monotone được giải quyết bằng cách sử dụng một độ hỗ trợ trọng số và đề ra một tính chất bao đóng giảm trọng (weighted downward closure property).

Tính chất bao đóng giảm trọng: Nếu một mẫu X là không phổ biến thì cha của nó cũng không thể phổ biến và trọng số hỗ trợ WS(X) ≥ WS(cha X) luôn đúng.

Tất nhiên, độ hỗ trợ trọng số của mẫu {A,B} trong WARM không xem xét biện pháp hỗ trợ này. Gần đây, có trọng số phổ biến các thuật toán khai thác mẫu [17,18] dựa trên cách tiếp cận mẫu tăng trưởng đã được phát triển. Trọng số dựa trên khai thác mẫu làm giảm kích thước của không gian tìm kiếm, mà còn tìm các mẫu quan trọng hơn. Trọng số của một item là một số thực không âm giao để phản ánh tầm quan trọng của mỗi mục trong TDB. Cho một tập hợp của các mục, I = {i_1, i_2, i_3, ..., i_n}, trọng số của một mẫu được định nghĩa chính thức như sau:

$$\text{Weight(P)} = \frac{\sum_{i=1}^{length(P)} Weight(Pi)}{Length(P)}$$

Giá trị đạt được khi độ hỗ trợ của một mẫu phép nhân với trọng số của mẫu đó được gọi là độ hỗ trợ trọng số của mẫu đó. Nghĩa là, cho mẫu P, độ hỗ trợ trọng số của P được định nghĩa như sau:

$$WSupport(P)=Weight(P).Support(P).$$

Một mẫu gọi là mẫu phổ biến có trọng số nếu độ hỗ trợ trọng số của mẫu không nhỏ hơn độ hỗ trợ của ngưỡng.

3.1.1 Mẫu trọng số phổ biến tối đại

Đầu tiên, chúng ta đề xuất khai thác mẫu phổ biến tối đại với ràng buộc trọng số. Chúng ta định nghĩa một mẫu tham gia mới xem xét như cả hai mẫu. Thứ tự tham gia của chúng là khai thác mẫu phổ biến trọng số tối đại (MWFP). Trong MWFP khai thác, mẫu phổ biến trọng số được tìm thấy đầu tiên, mẫu phổ biến tối đại được khai thác từ mẫu phổ biến trọng số MFP.

Định nghĩa 3.1.1. Một mẫu được là trọng số phổ biến tối đại nếu không có mẫu cha nào có trọng số phổ biến chứa nó

Trong phương pháp tiếp cận này, MWFP khai thác khám phá những ứng viên mẫu phổ biến có trọng số dùng để MaxW (trọng số cao nhất trong CSDL) trước nhất, các mẫu phổ biến tối đại được phát hiện sau. Còn lại tính chất anti-monotone của MWFP, MaxW được sử dụng để phát hiện liệu một mẫu xấp xỉ mẫu phổ biến trọng số trước khi kiểm tra tập cha (Liệu rằng mẫu này có những tập cha có trọng số là phổ biến) .

Cuối cùng, một số của phổ biến tối đại nhưng trọng số không là các mẫu phổ biến được cắt tỉa, các mẫu phổ biến tối đại mới được khai thác như ứng viên MWFPs. Các mẫu trọng số tối đại được kiểm tra lần nữa cho thu giảm dữ liệu MWFP.

Sự tích hợp của một mẫu phổ biến trọng số với một mẫu phổ biến tối đại là hiển nhiên. Các ràng buộc trọng số phải được cân nhắc trước khi áp dụng vào kiểm

tra tập cha. Ví dụ: Giả sử các ngưỡng tối thiểu là 2, các GD trong CSLD đó trong bảng 1a và trọng số Range(0.4-0.8) được sử dụng như là trọng số thông thường trong bảng 3.3

Bảng 3.1 Cho CSDL giao dich MWFP

TID	*TRANSACTION*
100	**a b c d f g**
200	**a b c d f**
300	**a b d e h i**
400	**a b d e g**
500	**a b c d f g**
600	**a b e f g h**

Bảng 3.2 Các trọng số và độ hỗ trợ của item

ITEM	*WEIGHT*	*SUPPORT*
a	0.7	4
b	0.6	4
c	0.8	3
d	0.65	5
e	0.45	3
f	0.5	4
g	0.4	4

h	0.5	2
i	0.45	1

3.1.2 Ví dụ

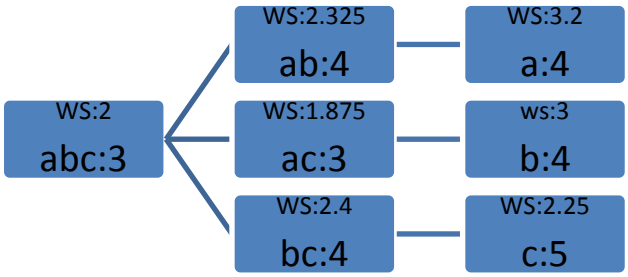
Tập của ứng viên các mẫu phổ biến trọng số sử dụng thuật toán Apriori là {{a}, {b}, {c}, {d}, {e}, {f}, {g}, {a,b}, {a,c}, {a,d}, {a, f}, {a, g}, {b,c}, {b,d}, {b, f}, {c, d}, {c, f}, {d, f}, {d, g}, {a,b,c}, {a,b,d}, {a,d, f}, {a,c,d}, {a,c, f}, {a, d, g}, {b,c,d}, {b,c, f}, {c,d, f,}, {a,b,c,d}, {a,b,c, f}, {a,c,d, f}, {a, b,d, f}, {b,c,d, f}, {a,b,c,d, f}}, MaxW bằng 0.8. Sau đó, các mẫu {a, b,c,d, f} và {a,d, g} là tối đại vì nó không có tập cha nào phù hợp. Tất nhiên, không phải là một mẫu phổ biến trọng số thực tế nào độ hỗ trợ trọng số tối đa của nó (MaxW support) là nhỏ hơn một độ hỗ trợ tối thiểu, do đó nó phải được cắt tỉa và kiểm tra mẫu trọng số phổ biến. Do đó, các mẫu phổ biến tối đại mới {{a,b,c,d}, {a,b,c, f}, {a,c,d, f}, {a, b,d, f}, {b,c,d, f}, {a, d}, {a, g}} được khai thác lần nữa. Trong số các mẫu, chỉ có mẫu {a, b,c,d} là mẫu phổ biến trọng số đúng, nó đó là MWFP. Ngược lại, {{a,b,c, f}, {a,b,d, f}, {a,c,d, f}, {b,c,d, f}, {a, d}, {a, g}} được cắt tỉa và tạo ra ít hơn các tập con một cấp.

Vì vậy, các mẫu ứng viên mới {{a,b, f}, {a,c, f}, {a, d, f}, {b,c, f}, {b,d, f}, {c,d, f}, {g}} được tạo ra. Mẫu {a,c, f} là một MWFP, nhưng mẫu khác thì không phải.Cuối cùng ta nhận được kết quả MWFPs {{a,b,c,d},và {a,c, f}}. Như trong ví dụ trên, để duy trì tính chất anti-monotone trong MWFP khai thác, chúng ta sử dụng trọng số tối đại (MaxW) khi một trọng số của item là không phổ biến phải được cắt tỉa. Vì lý do này, một số mẫu ứng viên có thể có trọng số là không phổ biến là chính xác.

Do đó, chúng ta phải kiểm tra liệu các ứng viên có trọng số là phổ biến một cách chính xác. Vấn đề xảy ra khi một mẫu ứng viên có trọng số không chính xác

phổ biến, nó là nếu chúng ta xóa ứng viên mẫu trọng số phổ biến thì sau đó thông tin nhất định các mẫu khác sẽ bị mất. Sự kiện nếu ứng viên có trọng số không phổ biến, bất kỳ mẫu phổ biến trọng số nào thuộc về các tập con của ứng viên có thể tồn tại, vì thế nó có thể tìm thấy một MWFP từ tập con. Trong kết luận, nếu tạo ra một mẫu ứng viên là phổ biến với trọng số chính xác, sau đó chúng ta kiểm tra mẫu cha phổ biến trọng số liệu có thích hợp trong khai thác MWFP hay không. Chỉ nếu nó không phù hợp mẫu cha phổ biến trọng số, thì mẫu ứng viên là một MWFP và có thể chèn vào kết quả tập của khai thác MWFP. Ngược lại, Nếu ứng viên phổ biến trọng số không chính xác thì chúng ta phải nhắc lại việc kiểm tra liệu tất cả mức 1 giảm tập con của ứng viên mà phổ biến trong số. Điều đó có nghĩa là nếu mẫu ứng viên của một nút lá là phổ biến trọng số khi thuật toán khai thác MWFP nghiên cứu kỹ trong thứ tự đầu tiênsâu cho việc tạo ra các mẫu ứng viên MWFP, sau đó các nút cha của nó phải được duyệt và được kiểm tra các nút cha. Việc nghiên cứu chiến lược khai thác MWFP của chúng ta liên quan đến xử lý bổ sung này, tìm thấy toàn bộ MWFPs của dữ liệu GD mà không làm mất thông tin.

ITEM	*WEIGHT*
a	0.8
b	0.75
c	0.45

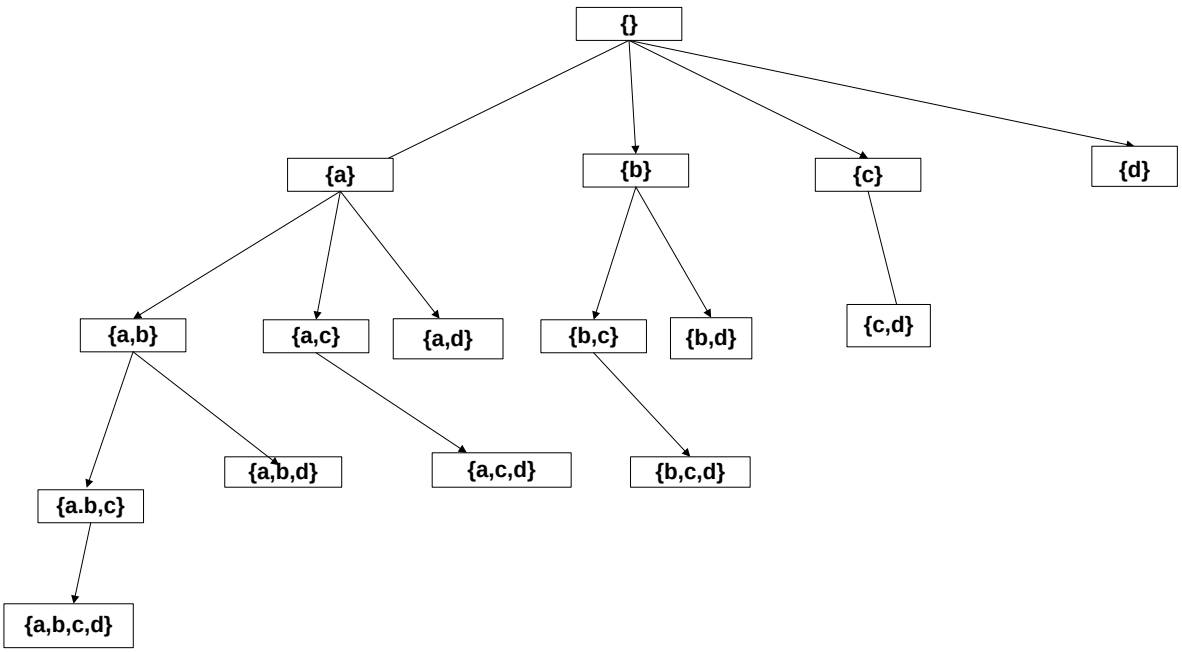


Hình 3.1 Các node và trọng số hỗ trợ

Cho ví dụ: như hình 12, với độ hỗ trợ cực tiểu là 2 và một danh sách của 3 item trọng số (a,b,c), trong nó {a:0.8,b:0.75,c:0.45}, mẫu {a,b,c} là phổ biến trọng số một cách chính xác độ hỗ trợ trọng số của nó là 2. Hơn nữa, nó không có mẫu cha phổ biến trọng số riêng, do đó nó là một MWFP. Tất nhiên, một tập con của nó không đúng độ hỗ trợ trọng số. Một mẫu, {a,c}, độ hỗ trợ trọng số của nó, 1.875, là nhỏ hơn độ hỗ trợ của độ hỗ trợ cực tiểu. Vì vậy, Một trong những đặc điểm của MFP, một mẫu phổ biến hoàn toàn thiết lập mà có thể được tạo ra từ một MFP có thể không được thừa hưởng một MWFP. Tuy nhiên, khai thác MWFP vẫn có ứng dụng rộng rãi như khai thác mẫu tiêu cực và tích cực cho phân loại.

3.2 Phương pháp khai thác MWFP

Trong phần này, chúng ta trình bày một khái niệm khung của mạng tập con, cùng với chiến lược nghiên cứu cho khai thác MWFP. Giả sử có một thứ tự giảm dần trong tổng trọng số ≤WD của các item, i trong CSDL. Nếu một item trước một item j xảy ra trong sự sắp xếp, chúng ta biểu thị sự xảy ra này là $i \leq WD_J$.



weight(a) ≥ weight(b) ≥ weight(c) ≥ weight(d)

Hình 3.2 Một ví dụ về một cây tiền tố 4 phần tử

Sự sắp xếp này có thể liệt kê sơ đồ mạng của tập con item hoặc một phần vượt ảnh hưởng tập S của item I. Chúng ta định nghĩa phần thứ tự $\leq$ trong S1,S2$\in$S sao cho S1 $\leq$ S2 if S1 $\subseteq$ S2. Hình 2 thể hiện một ví dụ hoàn thành sơ đồ mạng của tập con cho 4 item. Phần tử đầu của mạng là rỗng (ký hiệu là {} hoặc root), mỗi cấp dưới k khai báo tất cả k mẫu.

Những mức k thì được sắp xếp trong thứ tự trọng số giảm dần ở mỗi mức, tất cả các nút con được liên kết với tập con dễ dàng nhất trên mức trước. Sự định danh của mẫu là phần đầu nút, trong phần mở rộng của nút được đặt tên đuôi (tail).

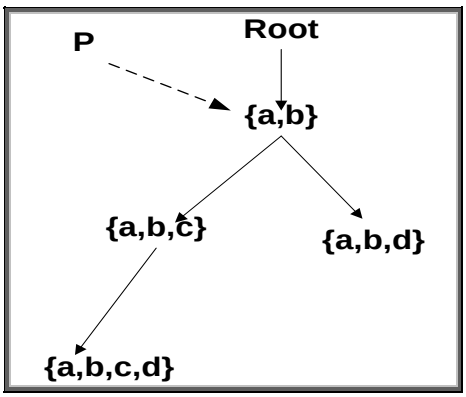
Cho ví dụ: Xem xét nút P ở hình 3.3 Phần đầu của P là {a,b}, phần đuôi của nó là {c,d}. Không gian tìm kiếm được sử dụng trong khai thác các mẫu phổ biến trọng số tối đại là một cây tiền tố chỉ có duy nhất những item phổ biến trọng số. Cụ thể, Các item phổ biến trọng số xấp xỉ được hình thành còn lại tính chất anti-monotone. Sử dụng tiền tố trọng số, chúng ta có thể tìm những item như vậy mà không bất cứ thông tin nào.

Cây tiền tố là mạng tập con, như thể hiện hình 2. Phần đuôi chứa tất cả những item trọng số mà xấp xỉ phổ biến, các trọng số của nó không lớn hơn mọi item trọng số của phần đầu. Cắt tỉa những item không phổ biến,không phải tất cả đều chính xác độ hỗ trợ trọng số với các item này là cần thiết.Thay vào đó chúng ta nhân với trọng số của phần đầu của item bởi độ hỗ trợ của nó.

Sử dụng bổ đề 1, độ hỗ trợ trọng số xấp xỉ này, W, luôn lớn hơn độ hỗ trợ trọng số chính xác. Ngoài ra, W là độ hỗ trợ trọng số tối đại của phần đầu tập con, do đó tính chất anti-monotone luôn được duy trì

Để khai thác các mẫu phổ biến tối đại từ cây tiền tố, chúng ta duyệt cây thứ tự sâu đầu tiên. Tại mỗi nút P, mỗi phần tử cuối của nút được tạo ra và được đếm là một phần mở rộng của mức 1. Nếu độ hỗ trợ của {đầu của P} $\cup$ {1phần mở rộng} là nhỏ hơn độ hỗ trợ cực tiểu ngưỡng, mọi siêu mẫu trong cây con của root tại {đầu của P} $\cup$ {1phần mở rộng} sẽ là trọng số không phổ biến.

Items	Weight
a	0.8
b	0.7
c	0.6
d	0.5



Hình 3.3 Một ví dụ về một cây tiền tố với thứ tự giảm dần trọng số

Bổ đề 1. Trong cây tiền tố, cây có thứ tự trọng số giảm dần, trọng số của mẫu P chỉ chứa các item đầu luôn luôn là bằng hoặc lớn hơn bất kỳ trọng số của bất kỳ mẫu cha chứa P.

$$\frac{w1+w2+w3+\cdots wn+wk}{n+1}, \frac{w1+w2+w3+\cdots+wn}{n} \geq \frac{w1+w2+w3+\cdots wn+wk}{n+1}$$

→ $(n+1)\ (w_1+w_2+w_3...+w_n) \leq n(w_1+w_2+w_3+...+w_n+w_k)$

→ $w_1+w_2+w_3...+w_n \geq n(w_k)$

và $w_1+w_2+w_3...+w_n$ lớn hơn hoặc bằng $n(w_k)$

Chứng minh: Cho tất cả item $a_1,a_2,a_3,...,a_m$, a_k được sắp xếp giảm dần theo trọng số như sau:

$w_1 \geq w_2 \geq w_3 \geq ... \geq w_n \geq w_k > 0, \frac{W1}{1} \geq \frac{W1+w2}{2} \rightarrow 2w_1 \geq (w_1+w_2) \rightarrow w_1 \geq w_2$

luôn đúng, Do đó trong lượng của mẫu {a1} luôn bằng hoặc lớn hơn mẫu $\{a_1,a_2\}$. Ngoài ra, trọng số mẫu {a1,a2} chỉ luôn luôn bằng hoặc lớp hơn $\{a_1,a_2,a_3\}$ khi:

$\frac{w1+w2}{2} \geq \frac{w1+w2+w3}{3}$ → $3(w_1+w_2) \geq 2(w_1+w_2+w_3) \rightarrow w_1+w_2 \geq 2w_3$

Trọng số mẫu $\{a_1,a_2,a_3,...,a_k\}$ là $(w_1 + w_2 + w_3 + ... + w_n)/n$ và luôn bằng hoặc lớn hơn mẫu {a1,a2,a3,...,an, ak},

$$\frac{w1+w2+w3+\cdots wn+wk}{n+1}, \frac{w1+w2+w3+\cdots+wn}{n} \geq \frac{w1+w2+w3+\cdots wn+wk}{n+1}$$

→ $(n+1)\ (w_1+w_2+w_3...+w_n) \leq n(w_1+w_2+w_3+...+w_n+w_k)$

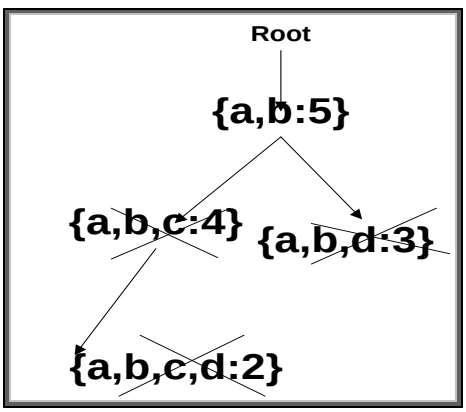
→ $w_1+w_2+w_3...+w_n \geq n(w_k)$

và luôn bằng hoặc lớn hơn $n(w_k)$

Hình. 3 cho thấy mục có thứ tự trọng số giảm dần và cây tiền tố của họ. Trọng số của tập {a, b} luôn luôn là bằng hoặc lớn hơn của nó siêu mẫu, {a, b, c} và {a, b, d} của Bổ đề 1. Như hình. 3, trọng số của tập {a, b} là 0,75, của {ab, c} là (0,8 + 0,7 + 0,6) / 3 = 0,7. Trọng số của {a, b, d} là (0,8 + 0,7 + 0,5) / 3 ≈ 0,667, cũng là bằng hoặc ít hơn so với các tập nút cha của nó, {a, b}.

Bổ đề 2: Trong một trọng số giảm dần cây tiền tố, nếu một mẫu nút trọng số là không phổ biến, sau đó tất cả các nút con của nó trọng số như không phổ biến. Như vậy, trong một duyệt sâu đầu tiên, nếu một mẫu được chính xác trọng số như không phổ biến, sau đó duyệt cây con của nó được dừng lại và các nút con còn lại không được xem xét.

Items	Weight
a	0.8
b	0.7
c	0.6
d	0.5



Hình 3.4 Một ví dụ mẫu trọng số phổ biến và cây tiền tố của nó

Chứng minh: Mỗi nút mẫu luôn là siêu mẫu nút cha của nó trong cây tiền tố. Dựa trên bổ đề 1, mỗi nút con luôn bằng hoặc ít hơn nút cha. Ngoài ra độ hỗ trợ của một nút cha là luôn luôn bằng hoặc lớn hơn các siêu mẫu của nó kể từ khi,nếu các mẫu tăng mức, số độ hỗ trợ giảm. Do đó, tất cả các nút con trong cây tiền tố thứ tự trọng số giảm thì phải nhỏ hơn hoặc bằng độ hỗ trợ trọng số của độ hỗ trợ của nút cha.

Như thể hiện ở Hình 3.4, độ hỗ trợ của {a,b} là 5, do đó độ hỗ trợ trọng số của mẫu {a,b} là 0.75 x 5=3.75. Nếu độ hỗ trợ ngưỡng (min_sup) là 3, mẫu {a,b} là phổ biến trọng số. Mặc dù, độ hỗ trợ trọng số của {a,b,c}, 0.7x4=2.8, không lớn hơn min_sup, do đó {a,b,c} là trọng số không phổ biến. Không cần thiết để kiểm tra liệu {a,b,c,d} là trọng số phổ biến hay không từ một nút con của {a,b,c}.

Bổ đề 3. Trong một trọng số giảm dần cây tiền tố, nếu phần đầu đứng đầu của một nút là trọng số là không phổ biến, sau đó nút N phải là một nút lá.

Chứng minh. Mẫu của N là một tập con của các nút con N (ký hiệu là C). Sự kiện nếu mẫu của N là trọng số không phổ biến, một số siêu mẫu có thễ phổ biến. Tuy nhiên, nếu tất cả item được sắp xếp thứ tự giảm dần, độ hỗ trợ trọng số của C luôn nhỏ hơn hoặc bằng độ hỗ trợ trọng số của phần đầu N dựa theo bổ đề 1. Do đó, nếu mẫu của N là trọng số không phổ biến, C phải trọng số không phổ biến

Vì vậy, mọi nút con của N luôn luôn trọng số không phổ biến. Hơn nữa, một cây con mọc từ N không thể có mọi mẫu là trọng số phổ biến. Vì lý do này, nút P trong cách duyệt sâu đầu tiên của cây, chúng ta có được một ứng viên cho tập kết quả của mẫu trọng số phổ biến tối đại. Trong khi đó, một tập cha trọng số phổ biến của P có thể đã được phát hiện

Vì vậy, chúng ta cần kiểm tra liệu một siêu mẫu của ứng viên P là được chứa trong tập kết quả. Chỉ có các mẫu trọng số phổ biến mà các tập cha là không trọng số phổ biến có thể được thêm vào tập kết quả. Mẫu phổ biến lớn nhất có thể được chứa các cây con có gốc là P là H $\cup$ T của P. Như hiển thị hình 3, là phần đầu của P

là {a,b} và phần đuôi là {c,d}, H ∪ T của P là {a,b,c,d}. Nếu {a,b,c,d} được phát hiện trọng số phổ biến, nó là cần thiết để mà duyệt mọi tập con của H ∪ T. Do đó, chúng ta có thể cắt tỉa toàn cây con có gốc là nút P. Các item của phần cuối là luôn sắp xếp giảm dần thứ tự trọng số của nó, thứ tự item là luôn luôn sửa lại. Vì vậy, không cần thiết để thứ tự lại các item cho mọi cây con, chúng ta có thể bỏ qua thời gian sắp xếp.

Để phát hiện các tập kết quả chính xác của MWFPs, chúng ta phải kiểm tra tất cả MWFPs để chắc chắn rằng không có tập cha nào của mọi mẫu được phát hiện trước khi thêm mẫu đến MWFPs. Một kỹ thuật được tập trung cải tiến đã được giới thiệu cải thiện việc kiểm ta hiệu suất tập cha mà không truy xuất quá nhiều tất cả tập MWFPs. Ý tương cơ bản là như sau. Nếu toàn bộ tập MWFP là quá lớn tại bất kỳ nút nào, chỉ một phần của tập MWFP là có thể là tập cha của các mẫu tại nút cho trước.

Do đó, chúng ta sử dụng một cục bộ tập MWFP, mà nó là tập cha của toàn bộ tập MWFP và liên quan các nút, để kiểm tra hiệu quả của các tập cha. Trong khai thác MWFP của chúng ta, tập MWFP cục bộ cho thư mục gốc được khởi tạo là tập null. Giả sử chúng ta đang kiểm tra nút k và duyệt qua Kn, nơi Kn= K ∪ {i} (i là mỗi item của phần đuôi N). Tập MWFP cục bộ cho Kn chứa tất cả các mẫu trong tập MWFP cục bộ chứa trong MWFPs của Kn được chèn đến tập MWFP toàn cục.

Do đó, tập ứng cử viên không còn cần thiết để tiến hành kiểm tra siêu trong tập MWFP toàn cục. Thay vào đó, tập MWFP cục bộ bao gồm tất cả các tập cha của nút hiện tại. Như vậy, nếu tập MWFP cục bộ của một nút ứng cử viên là rỗng, sau đó tập MWFP toàn cục không có các tập cha. Ngược lại, nếu tập MWFP cục bộ không phải là rỗng, sau đó một tập cha sẽ được tìm thấy trong tập MWFP toàn cục. Khuôn khổ khai thác MWFP của tác giả để trọng số giảm dần là như sau. Đầu tiên, một mẫu ứng cử viên có thể được kiểm tra để xác định xem liệu độ hỗ trợ trọng số thực sự của nó là nhỏ hơn hoặc bằng với ngưỡng hỗ trợ tối thiểu (min_sup) hay không. Nếu tập này được thực trọng số như phổ biến, tập cha của nó (phần mở

rộng) có khả năng được mẫu phổ biến thực sự trọng số. Do đó, sự kết hợp mẫu và mỗi item phổ biến xấp xỉ trọng số của mẫu đuôi làm cho tiện ích mở rộng 1 cấp của nó. Cuối cùng, thực mẫu phổ biến có trọng số của một nút lá phải được kiểm tra để xác định xem nó có một tập cha trọng số phổ biến.

3.3 Nghiên cứu liên quan

Trong CSDL dày đặc, mật độ item xuất hiện tầng suất cao, Zaki và Gouda [12] đề xuất Diffset như là một phương pháp để tính toán trọng số hỗ trợ của tập phổ biến một cách nhanh chóng và tiết kiệm bộ nhớ để lưu trữ Tidset. Vì vậy, luận văn nghiên cứu các thuật toán khai thác tập đánh trọng, áp dụng **Diffset,** đề nghị thuật toán khai khác mẫu trọng số phổ biến tối đại sử dụng Diffset nhằm giảm thời gian khai thác và tiết kiệm bộ nhớ lưu trữ.

Dựa vào định nghĩa trên ta có thể sử dụng Diffset thay vì sử dụng Tidset cho việc tính toán trọng số hỗ trợ của tập itemset trong quá trình khai thác FWI.

3.4 Giới thiệu Diffset

Diffset tính sự khác biệt giữa hai Tidset (tập khác nhau giữa Tidset(X) so với Tidset(Y)) trong lớp tương đương nhau.

Trong một cơ sở dữ liệu dày đặc, kích thước của Diffset là nhỏ hơn so với Tidset. Vì vậy, sử dụng Diffset sẽ tiêu tốn ít dung lượng bộ nhớ, không gian lưu trữ giảm đáng kể và do đó cho phép các máy tính nhanh giá trị hỗ trợ trọng số.

Cho d(PXY) là sự khác nhau giữa tập PX và PY.

Ta có: d(PXY) = t(PX) - t(PY)[12] (3.1)

Trong đó PX và PY thuộc lớp tương đương [P].

Giả sử rằng chúng ta đã có d(PX) và d(PY) và cần tính d(PXY):

Theo các kết quả trong [12], chúng ta có thể có được nó một cách dễ dàng

bằng cách tính toán sự khác biệt giữa tập d(PY) và d(PX):

d(PXY) = d(PY) - d(PX) [12] (3.2)

Dựa vào công thức 3.1, 3.2 ta có thể tính toán độ hỗ trợ trọng số của PXY bằng cách tính d(PXY) như sau:

$$ws(PXY) = ws(PX) - \frac{\sum_{t \in d(PXY)} tw(t)}{\sum_{t \in T} tw(t)}$$

Định lý 3.3

Nếu d(PXY) = Ø thì ws(PXY) = ws(PX)

Vì d(PXY) = Ø => $ws(PXY) = ws(PX) - \frac{\sum_{tk \in t(PXY)} tw(tK)}{\sum_{tk \in T} tw(tK)} = ws(PX)$

Để tiết kiệm bộ nhớ cho việc lưu trữ Diffset và thời gian cho việc tính toán Diffset, sắp xếp tập itemsets in tập lớp tương đương tăng dần theo trọng số hỗ trợ.

3.5 Thuật toán dựa trên Diffset

3.5.1 Thuật toán WIT-FWI-DIFF dựa trên Diffset

Thuật toán WIT-FWI-DIFF sử dụng Diffset để tính toán giá trị *ws*. Vì mức đầu tiên chỉ lưu trữ Tidset, nếu L_r thuộc mức đầu tiên, nghĩa là l_i và l_j thuộc mức 1, chúng ta sẽ sử dụng $Y=d(X) = d(l_i \cup l_j) = t(l_i) - t(l_j)$ (theo công thức (3.1)).

Diffset được tính từ mức thứ 2, chúng ta có thể sử dụng công thức (3.2) để tính $Y = d(X) = d(l_i \cup l_j) = d(l_j) - d(l_i)$.

Ngoài ra thuật toán còn sử dụng định lý 3.3 nhằm tính toán nhanh chóng ws(X).

Đầu vào: Cơ sở dữ liệu các GD D, ngưỡng hỗ trợ trọng số tối thiểu *min_ws*

Đầu ra: FWI chứa tất cả các tập phổ biến itemsets từ CSDL D, thỏa mãn *min_ws*

Phương thức:

WIT-FWI-DIFF()

{

1. L_r = tất cả các *item* mà *ws* thỏa *min_ws*
2. Sắp các nút trong L_r tăng dần theo ws
3. Khởi tạo tập FWI = ∅
4. Gọi hàm FWI-EXTEND-DIFF với tham số là L_r FWI-EXTEND-DIFF(L_r)
5. Cho mỗi nút l_i trong L_r
6. Thêm (l_i.itemset, l_i.ws) vào trong FWI
7. Tạo một tập L_i bằng cách nối li với tất cả $_{lj}$ theo sau trong L_r:
8. Thiết lập tập X = l_i.itemset ∪ l_j.itemset
9. Nếu L_r là mức đầu tiên thì Y = t(l_i)- t(l_j) // theo công thức 3.1
10. Ngược lại Y= d(l_j) - d(l_i) // theo công thức 3.2
11. Nếu Y= ∅ thì ws(X)= ws(l_i) // theo định lý 3.3
12. Ngược lại ws(X) = COMPUTE-WS-DIFF(Y) // theo định lý 3.3
13. Nếu ws(X) thỏa min_ws thì khi đó
14. Thêm một nút (X, Y, ws (X)) vào trong L_i
15. Nếu số lượng nút trong $L_i \geq 2$ khi đó
16. Gọi đệ qui hàm FWI-EXTEND-DIFF với biến L_i

}

Bảng 5: Thuật toán WIT-FWI-DIFF cho khai thác trọng số các itemset

Sử dụng các dữ liệu ví dụ được trình bày trong bảng cơ sở dữ liệu GD D

2.1 và bảng trọng số các item trong CSDL D 2.2 như sau:

Bảng 3.3 Trọng số GD của từng GD

Transations	Tw
1	0.45
2	0.2
3	0.45
4	0.3
5	0.42
6	0.43
Sum	2.25

Bảng 3.4 Trọng số hỗ hợ cho tập phổ biến 1 item

X	Weighted support (ws)
A	0.72
B	1
C	0.6
D	0.78
E	0.81

Ta minh họa các thuật toán WIT-FWI-DIFF với min_ws = 0.4 như sau.

Mức 1 của WIT-cây có các nút con chứa duy nhất một mặt item. Chúng được sắp xếp thứ tự tăng dần của |tidset|. Mục đích của việc này là để tính toán Diffset nhanh hơn. Ví dụ:

- Xét các nút B và D, nếu họ không được sắp xếp, chúng ta phải tính toán

- Xét A nối với C

d(AC) = t(A) - t(C) = 1345 - 2456 = 13

=> ws(AC) = ws(A)- $\frac{\sum_{t\in d(AC)} tw(t)}{\sum_{t\in T} tw(t)}$ =0.72- $\frac{0.45+0.45}{2.25}$= 0.32 < min_ws

- Xét A nối với D

d(AD) = t(A) - t(D) = 1345 - 1356 = 4

=> ws(AC) = ws(A)- $\frac{\sum_{t\in d(AD)} tw(t)}{\sum_{t\in T} tw(t)}$ =0.72- $\frac{0.3}{2.25}$= 0.59> min_ws

- Xét A nối với E:

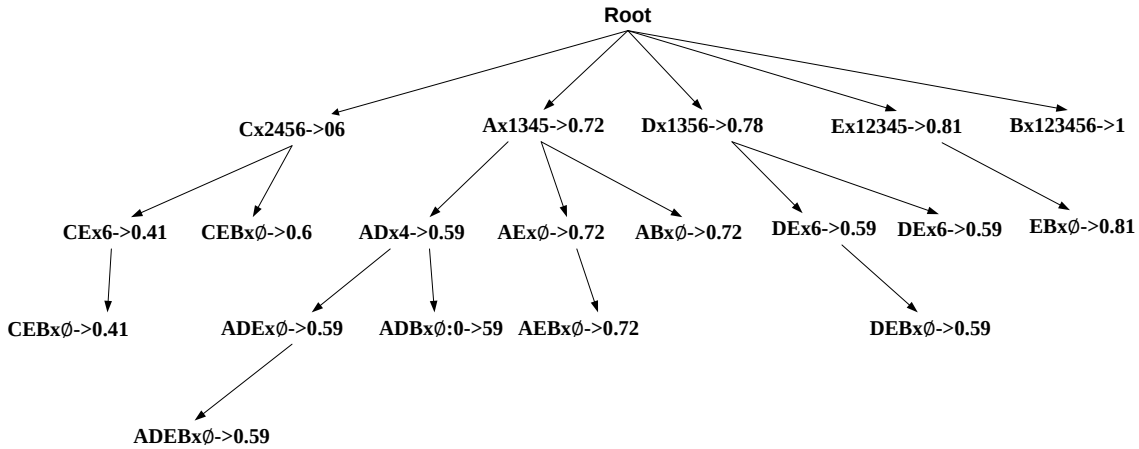
d(AE) = t(A) - t(E) = 1345-12345 = ∅

=> ws(AD) = ws(A) = 0.72

- Xét A nối với B:

d(AB) = t(A) - t(B) = 1345 - 123456 = ∅

= > ws(AB) = ws(A) = 0.72



Hình 3.5 Kết quả của thuật toán WIT-FWI-DIFF từ CSDL trong Bảng 2.9 với trọng số hỗ trợ tối thiểu Min_sup=0.4

3.5.2 Khai thác MWFIM_DIFF dựa trên Diffset

3.5.3.1 Thuật toán MWFIM_DIFF dựa trên Diffset

ALGORITHM [MWFIM_DIFF]: Maximal Weighted Frequent Itemset Mining Diffset

Input: (1) A Transaction Database: TDB;

(2) Weights of the items within weight range:

MinW–MaxW;

(3) Minimum support threshold: min_sup

Output: The complete set of maximal weighted frequent

patterns

Begin

1. Let MWFP be the set of maximal weighted frequent patterns. Initialize

MWFP ← {};

2. Scan TDB once to find the global weighted frequent items as follows:

support * MaxW ≥ min_sup

3. Sort the items in weight-descending order;

4. Scan the TDB again and build vertical bitmaps Diffset to store

weighted frequent candidate items of transactions in the TDB.

5. Call MWFIM(root, MWFP, false);

Procedure MWFIM (Current node C, MWFP, Boolean isHUT)

1: isAdded = false;

2: allWF = true;

```
3: HeadWS = C.Head.Weight * C.Head.Support;
4: If (HeadWS ≥ min_sup)
5: For each item i in all the remaining items
//remaining items that have lower orders than the items of
C.Head
6: If (C.Head.Weight * C.Head ∪ {i} ≥ min_sup), then
insert i into C.Tail;
7: Else allWF = false;
8: for each item i in C.Tail
9: If (i is the first item in C.Tail) isHUT = true;
10: Else isHUT = false;
11: extended_C = C∪ {i};
12: isAdded_local = MWFIM(extended_C, MWFP, isHUT);
13: If (isAdded_local) isAdded_MWFP = true;
14: If (isHUT and AllWF = true) return isAdded;
15: if (C.Tail == {})
16: If (not ExistSubset(C.Head, MWFP)) {
// C.Head is not subset of a weighted frequent pattern
17: Insert C.Head into MWFP;
18: isAdded = true; }
19: Else isAdded = false;
20: return isAdded;
```

3.5.2.2 Ví dụ một thuật toán MWFIM

Trong thuật toán ta áp dụng các công thức:

1) d(PXY) = t(PX)-t((PY)

2) d(PXY) = d(PY) - d(X)

3) σ(PXY) = σ(X) - |d(PXY)| = σ(X) - |d(PY) - d(PX)|

Trong khai thác luật kết hợp với trọng số (MINWAL) [14] trọng số được tính như sau:

$$\text{Weight(P)} = \frac{\sum_{i=1}^{length(P)} Weight(Pi)}{length(P)}$$

Trong số hỗ trợ:

$$\text{WS(P)} = \text{Weight(P)} * \sigma(\text{P})$$

Ví dụ ta có một CSDL TDB như hình 12, các items I = {a, b, c, d, e, f, g, h, i}, các GD T ={100,200,300,400,500,100}, các items được đánh trọng số. Cho một ngưỡng min_sup có giá trị do người dùng định nghĩa ví dụ bằng 2. Tìm các mẫu phổ biến tối đại thỏa mãn ngưỡng trong CSDL TDB vừa cho.

Bảng 3.5 CSDL TDB GD và trọng số items

TID	*TRANSACTION*
100	**a b c d e f g**
200	**a b c d f**
300	**a b d e h i**
400	**a b d e g**

500	**a b c d e f g**
600	**a b e f g h**

ITEM	*WEIGHT*	*SUPPORT*
a	**0.7**	**6**
b	**0.6**	**6**
c	**0.8**	**3**
d	**0.65**	**5**
e	**0.45**	**5**
f	**0.5**	**4**
g	**0.4**	**4**
h	**0.5**	**2**
i	**0.45**	**1**

Bước 1: Khởi tạo MWFP = {};

Bước 2: Duyệt toàn bộ CSDL TDB tìm ra các item thỏa mãn min_sup: item.support *item.weight ≥ min_sup. Kết quả thu được ở Bảng 3.7 ta thấy các item bị gạch chéo "g","h","i" không thỏa mãn min_sup nên không thêm vào danh sách phổ biến.

Bảng 3.6 Danh sách trọng số hỗ trợ của các item

ITEMS	*WEIGHT*	*SUPPORT*	WS
a	0.7	6	4.2
b	0.6	6	3.6

d	0.65	5	3.25
c	0.8	3	2.4
e	0.45	5	2.25
f	0.5	4	2
~~**g**~~	~~0.4~~	~~4~~	~~1.6~~
~~**h**~~	~~0.5~~	~~2~~	~~1~~
~~**i**~~	~~0.45~~	~~1~~	~~0.45~~

Bước 3: Sắp xếp các items theo thứ tự giảm dần trọng số. Danh sách sau khi được sắp xếp theo giảm trọng số như sau:

Bảng 3.7 Sắp xếp trọng số giảm dần các item

ITEM	*WEIGHT*
c	**0.8**
a	**0.7**
d	**0.65**
b	**0.6**
f	**0.5**
e	**0.45**

Bước 4: Duyệt CSDL TDB lần nữa xây dựng bảng thanh dọc bitvectors Diffset từ các ứng viên phổ biến Bảng 3.7

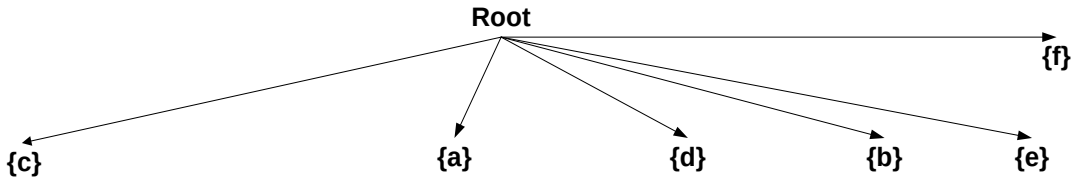
Bảng 3.8 Minh họa thanh dọc Diffset từ CSDL TDB

VERTICAL TIDSET					
c	**a**	**d**	**b**	**e**	**F**
1	1	1	1	1	1
2	2	2	2	2	4
5	3	3	3	4	5
	4	4	4	5	6
	5	5	5	6	
	6		6		

VERTICAL BITVECTORS DIFFSET					
c	**a**	**d**	**b**	**e**	**f**
0	0	0	0	0	0
0	0	0	0	0	1
1	0	0	0	1	1
1	0	0	0	0	0
0	0	0	0	0	0
1	0	1	0	0	0

Bước 5: Ta gọi hàm MWFIM_DIFF để thực hiện các bước thuật toán:

Đầu tiên ta khởi tạo nút gốc là root, sắp xếp thứ tự giảm trọng số mà trọng số lớn nhất là 0.8 là (c, a,d, b, e, f). Danh sách này lần lượt thêm vào phần đuôi của root hình 3.7.



Hình 3.6 Khởi tạo mức root và các item phổ biến

Ta thực hiện duyệt cây tiền tố theo theo thứ tự sâu đầu tiên từ trái sang phải để kiểm tra và mở rộng nút. Bắt đầu từ nút {c} có phần đuôi có nút là {a, d, b, e, f} trong đó các item được sắp xếp giảm dần nên phần đầu {c} là có trọng số lớn nhất

0.8. Để mở rộng mức 1 chúng ta kết hợp {c} ∪ {i} là từng item {a, d, b, e, f}. Ta kiểm tra mẫu {c,a} có trọng số hỗ trợ có thỏa mãn min_sup = 2 hay không?

- Tính ws(ca)

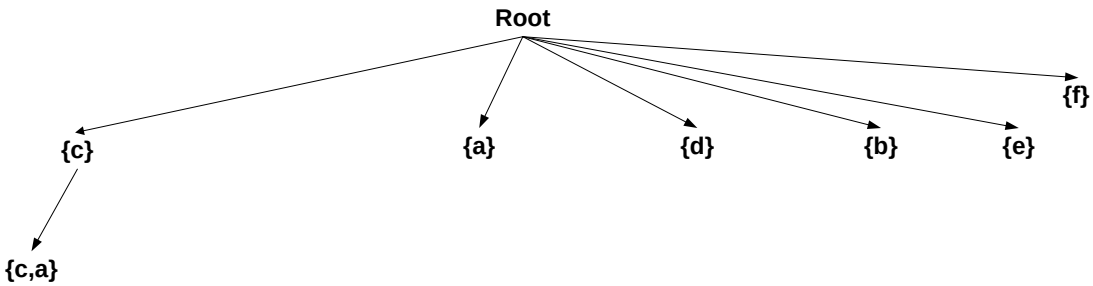
weight(ca) = [weight(c)+weight(a)]/2 = [(0.8+0.7)]/2 =0.75

ws(ca) = σ(ca) * weight(ca)

mà:

σ(ca) = σ(c) - |d(ca)| = σ(c) - |d(a) -d(c)| = 3 - |∅| = 3-0 =3

→ ws(ca) = σ(ca) * weight(ca) = 0.75 * 3 = 2.25 > min_sup



Hình 3.7 Mở rộng nút {c}

Vì {c,a} thỏa mãn min_sup nên {c,a} là mẫu phổ biến và thêm vào phần mở rộng của {c} Hình 3.8. Ta tiếp tục thực hiện duyệt theo thứ tự sâu đầu tiên từ trái sang phải. Ta kiểm tra mẫu {c,a,d} có trọng số hỗ trợ có thỏa mãn min_sup = 2 hay không?

- Tính ws(cad) như sau:

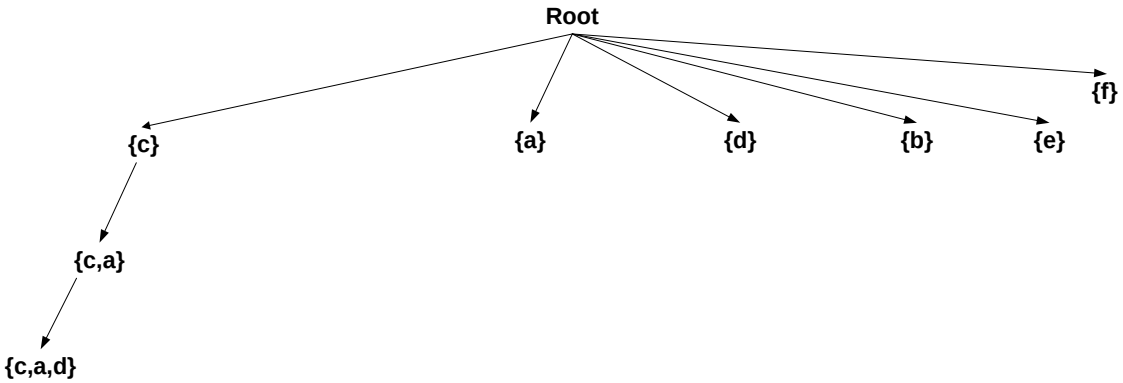
ws(cad) = σ(cad) * weight(cad)

weight(cad) = [weight(ca)+weight(d)]/2 =(0.75 + 0.65)/2 = 0.70

và: σ(cad) = σ(ca) - |d(cad)| = σ(ca) - |d(d) - d(ca)| = 3 - |∅| = 3- 0 =3 (σ(ca) = 3 từ mẫu {c,a})

→ ws(cad) = σ(cad) * weight(cad) = 3 * 0.70 = 2.1 > min_sup

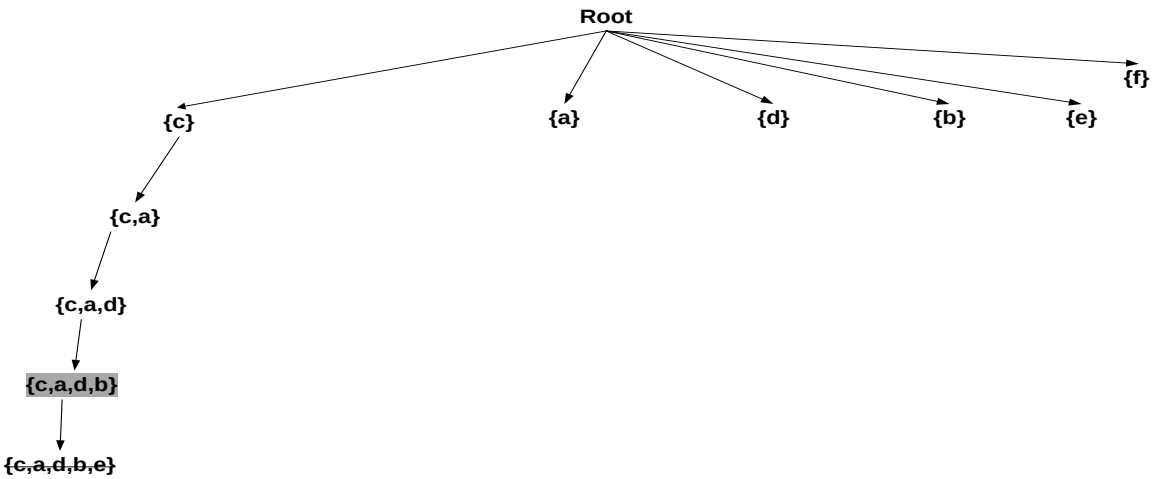
Vì {c,a,d} thỏa mãn min_sup nên {c,a,d} là mẫu phổ biến và thêm vào phần mở rộng của {c,a}. Ta tiếp tục thực hiện duyệt thứ tự sâu đầu tiên từ trái sang phải.



Hình 3.8 mở rộng nút {c,a}

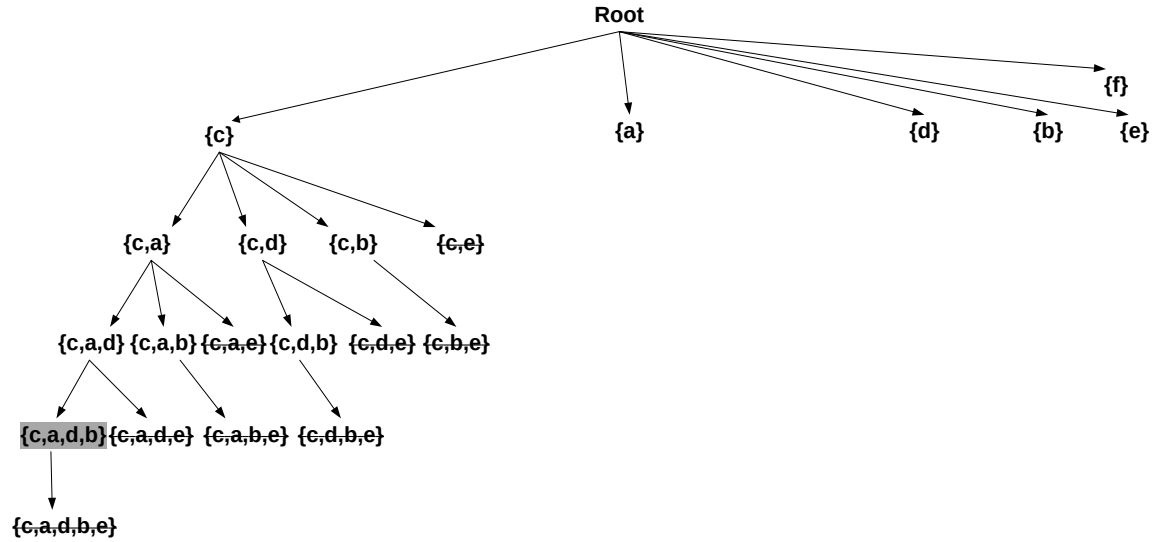
Thực hiện tương tự các bước trên ta tính được mẫu {c,a,d,b} có trọng số hỗ trợ bằng 2.0625 > min_sup nên nó được thêm vào phần mở rộng. Nhưng khi kiểm tra mẫu{c,a,d,b,e} có trọng số hỗ trợ bằng 19.2 rõ ràng không thỏa mãn min_sup vì nhỏ hơn 2 nên nó là mẫu không phổ biến và cũng chính là một nút lá của nhánh {c}. Do đó ta trở lại với nút {c,a,d,b} và kiểm tra xem nó có phải là một mẫu trọng số phổ biến tối đại nếu thỏa mãn điều kiện không có mẫu nào chứa nó nằm trong tập MWFP.

Thực hiện kiểm tra: Duyệt tất cả các mẫu trong MWFP xem có mẫu nào chứa {c,a,d,b} nếu có mẫu chứa {c,a,d,b} thì nó không là MWFP ngược lại thì nó là MWFP, ta nhận thấy rằng không có mẫu nào chứa nó nên {c,a,d,b} (lúc này MWFP là rỗng) được thêm vào tập MWFP ta có được cây tiền tố Hình 3.9



Hình 3.9 mở rộng nhánh {c}

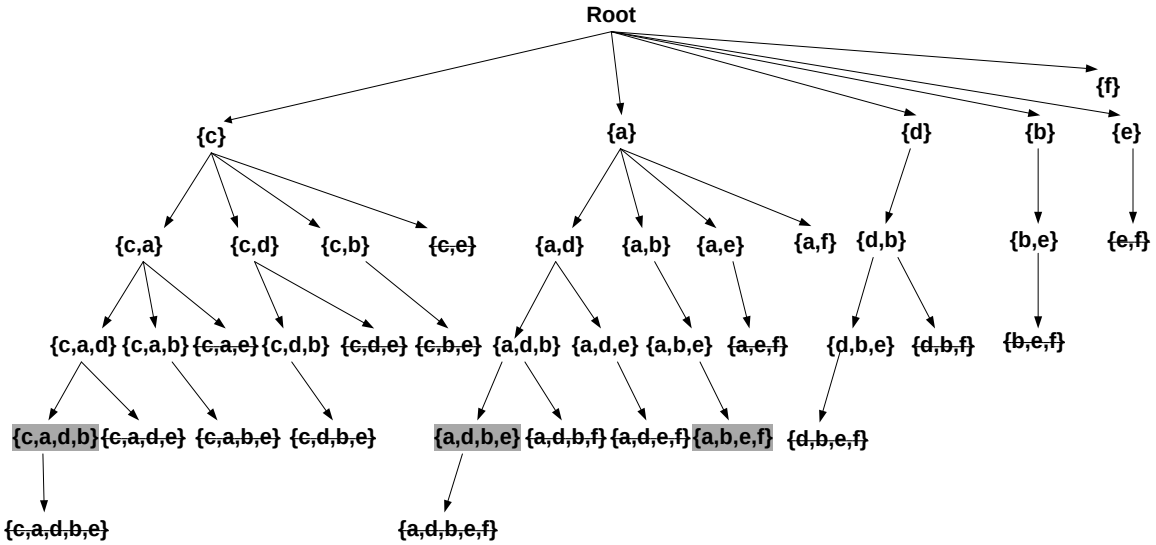
Thực hiện tương tự các bước trên ta có được hình 3.10 Các mẫu bị gạch tên là các mẫu không thỏa mãn min_sup nên nó không phổ biến và nó cũng là các nút là của nhánh {a}. Các mẫu không bị gạch như {c,ab},{c,d,b},{c,b} là các mẫu trọng số phổ biến vì thỏa mãn min_sup. Ngoài ra các mẫu này không là mẫu MWFP vì nó chứa trong mẫu {c,a,d,b} là một MWFP.



3.10 duyệt hết nhánh {c}

Chúng ta tiếp tục thực hiện tương tự các bước trên cho các nhánh còn lại ta được Hình 20. Ta thấy rằng sau khi thực hiện duyệt thứ tự sâu đầu tiên từ trái sang phải nhánh {a} ta có nút {a,d,b,e,f},{a,d,b,f},{a,d,e,f},{d,e,f},{a,f} là các nút không thỏa mãn min_sup bằng 2 nên nó không là mẫu phổ biến và cũng là các nút là của

nhánh {a}. Ta xem xét hai mẫu {a,d,b,e},{a,b,e,f} thỏa mãn điều kiện thêm vào tập MWFP vì nó không có mẫu nào chứa nó trong MWFP. Còn lại các mẫu phổ biến khác không thỏa mãn để thêm vào MWFP cuối cùng là được Hình 3.11



Hình 3.11 MWFP là {{c,a,d,b},{a,d,b,e},{a,b,e,f}} là các mẫu trọng số phổ biến tối đại

Vậy sau khi thuật toán kết thúc ta có tập MWFP là {{c,a,d,b},{a,d,b,e},{a,b,e,f}} là các mẫu trọng số phổ biến tối đại.

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Môi trường thực nghiệm

- Ngôn ngữ lập trình C# VS2013
- CPU Intel core i3, Ram 6G
- Microsoft Windows 7 64 – bit.

❖ **Đặc điểm cơ sở dữ liệu thực nghiệm**

Các kết quả thực nghiệm đã được thử nghiệm trên CSDL liệu được lấy từ trang web Frequent Itemset Mining Dataset Repository: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php. Các bộ dữ liệu chuẩn đã được sửa đổi bằng cách tạo ra một bảng để lưu trữ các giá trị trọng số của từng *item* (giá trị trong khoảng từ 1 đến 90) và giá trị min_sup từ cho mỗi một cơ sở dữ liệu.

Cơ sở để gán trọng số là: Trong CSDL bán hàng siêu thị trọng số thể hiện số lượng của mặt hàng. min_sup là do người dùng tự định nghĩa nên có thể thay đổi tùy ý. Thuật toán có thể ứng dụng vào CSDL GD thực tế.

Kết quả thực nghiệm khai thác mẫu trọng số phổ biến tối đại trên các CSDL chuẩn được hiển thị minh họa ở hình 4.1,4.2,4.3,4.4,4.5

Bảng 4.1 Cơ sở dữ liệu thực nghiệm có chỉnh sửa

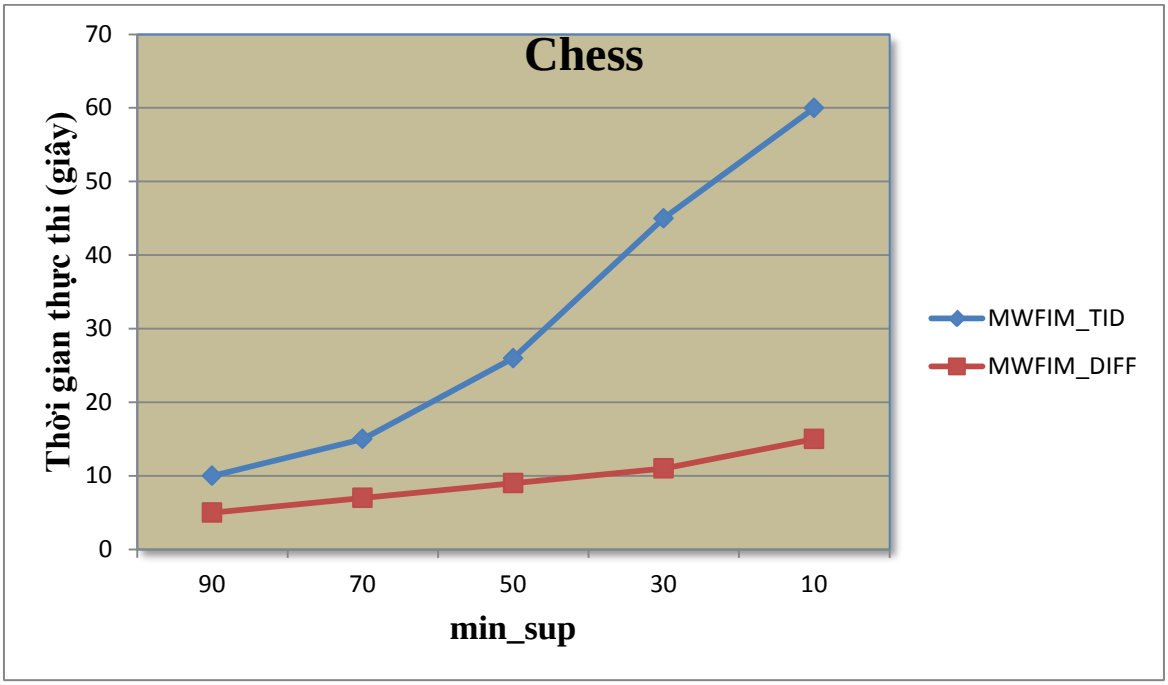
CSDL	#Trans	#Item	Size	Tình trạng
Chess	3196	75	334 KB	Đã được sửa đổi
Mushrooms	8416	119	589 KB	Đã được sửa đổi
BMS1_itemset_mining	59602	498	934 KB	Đã được sửa đổi
Connect	88162	129	8.82 MB	Đã được sửa đổi

4.2 Kết quả thực nghiệm

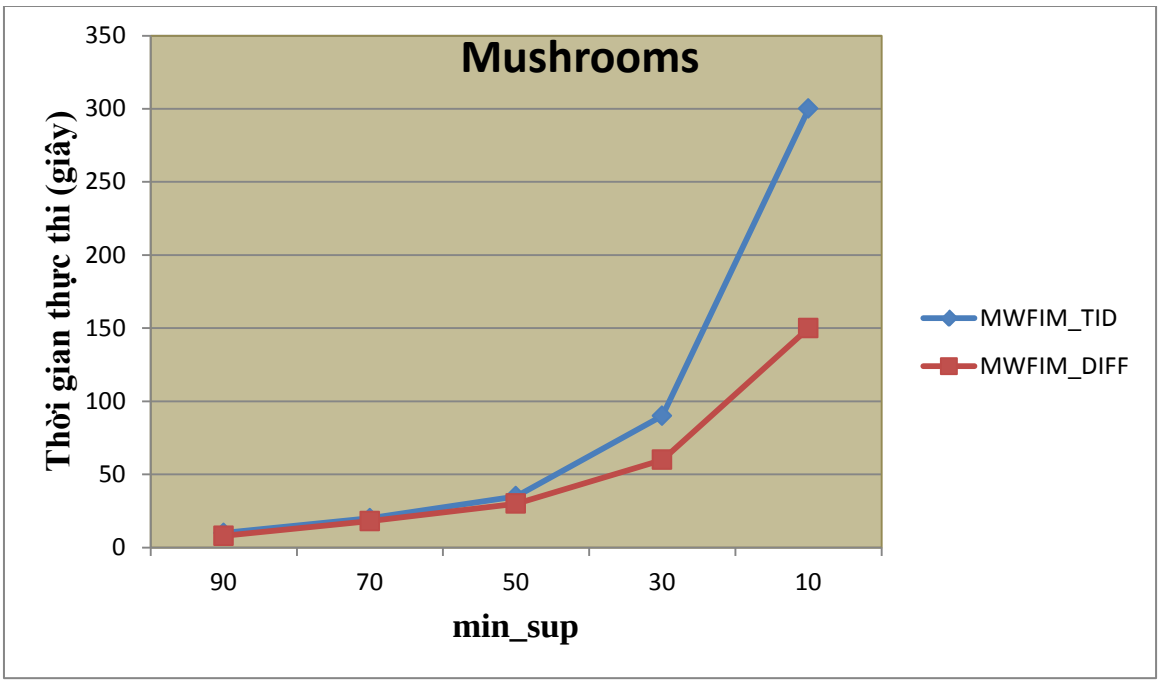
Thời gian thực thi để tìm kiếm các tập được đánh trọng số thay đổi tùy theo min_sup từ 10 đến 90, giá trị min_sup càng nhỏ thì thực hiện càng lâu và ngược lại. Số giao dịch càng nhiều thì bộ nhớ sử dụng càng tăng và ngược lại.

Cài đặt thực nghiệm cho thuật toán MWFIM sử dụng Tidsets và MWFIM_DIFF sử dung Diffsets ta nhận được kết quả thực nghiệm với cùng kết quả mẫu trọng số phổ biến tối đại nhưng khác nhau về thời gian thực thi.

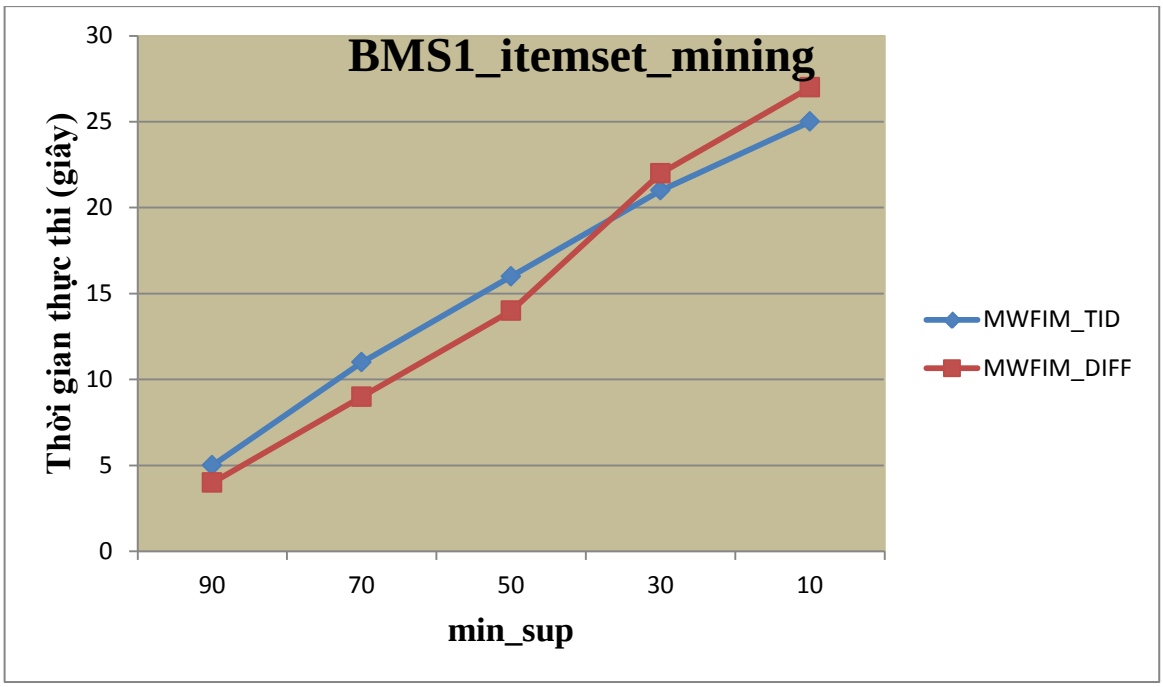
Từ những kết quả thử nghiệm ở trên, ta thấy được thời gian xử lý MWFIM_TID (sử dụng Tidset) tốn khá nhiều thời gian khi xử lý các CSDL có số sản phẩm quá lớn hoặc min_sup nhỏ. Tuy nhiên với thuật toán cải tiến MWFIM_DIFF (sử dụng Diffsets), hệ thống xử lý khá nhanh và ổn đối với các CSDL có tầng số GD dày đặc, có kích thước vừa và nhỏ đối với min_sup thích hợp CSDL.



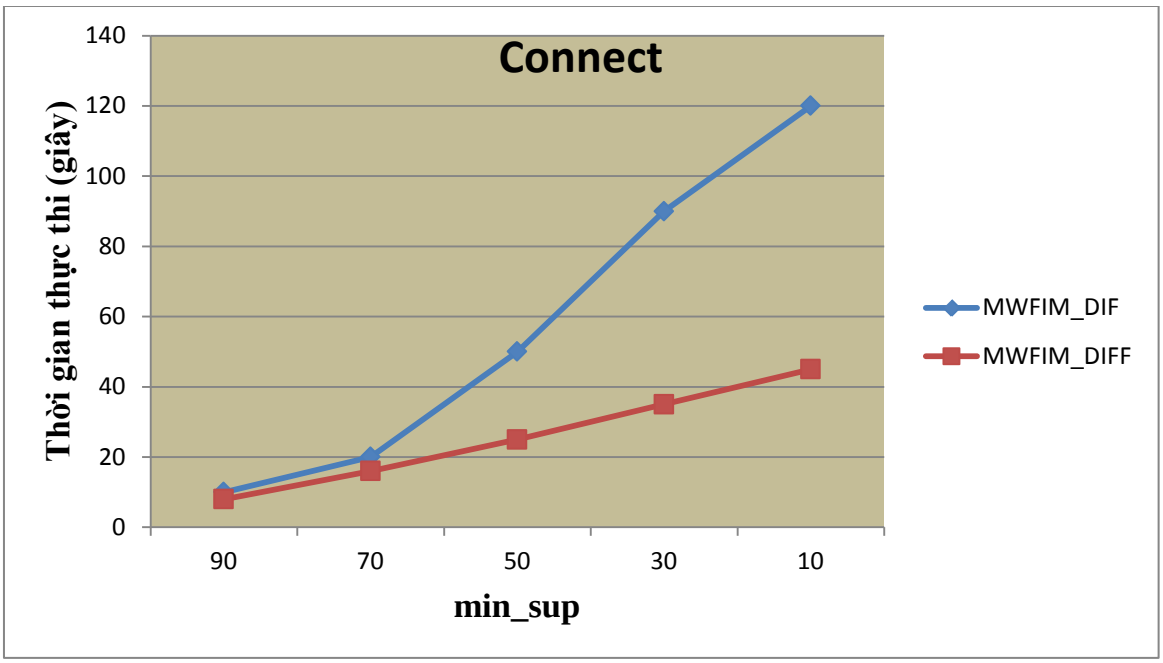
Hình 4.1 Biểu đồ thực nghiệm MWFP trên CSDL Chess



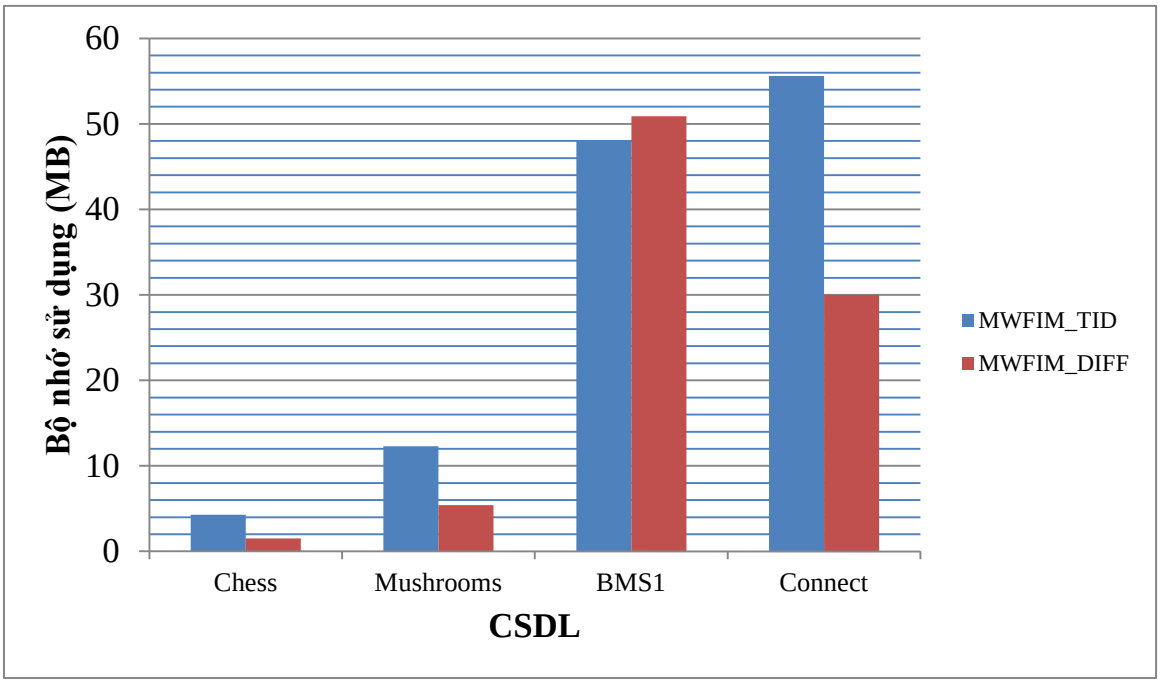
Hình 4.2 Biểu đồ thực nghiệm MWFP trên CSDL Mushrooms



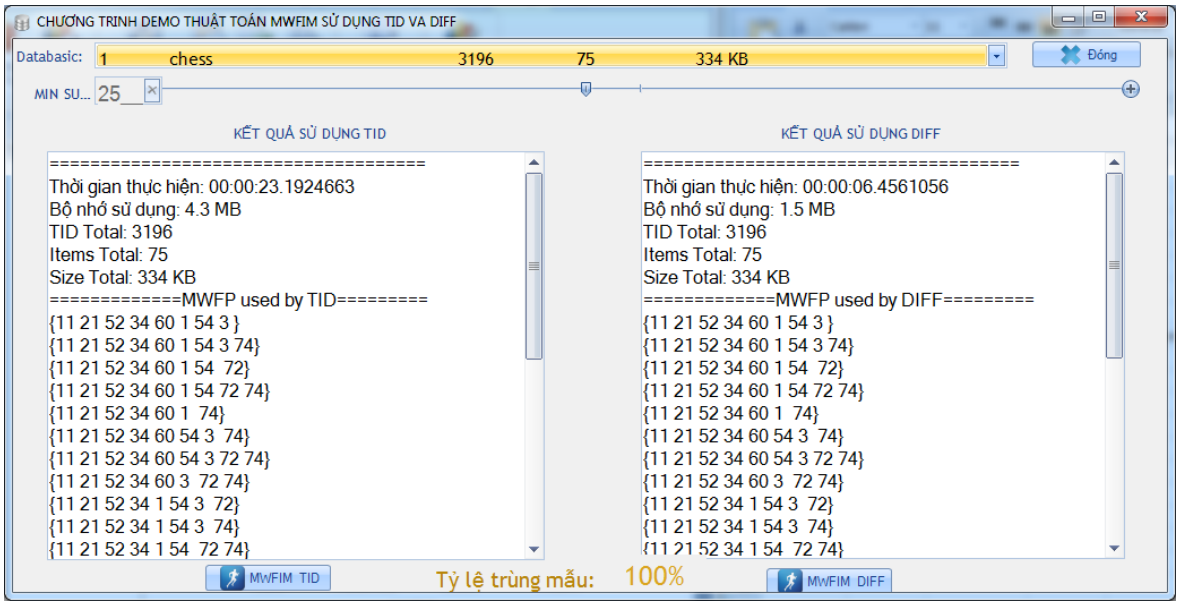
Hình 4.3 Biểu đồ thực nghiệm MWFP trên CSDL BMS1_itemset_mining



Hình 4.4 Biểu đồ thực nghiệm MWFP trên CSDL Connect



Hình 4.5 Biểu đồ thực nghiệm bộ nhớ sử dụng



Hình 4.6 Chương trình đề mô thuật toán

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Đề tài tập trung vào nghiên cứu các thuật toán khai thác các mẫu trọng số phổ biến được đánh trọng số dựa trên các thuật toán khai thác tập phổ biến trên CSDL nhị phân. Thông qua quá trình thực hiện đề tài đã thực hiện được các mục tiêu:

- Nghiên cứu cơ sở lý thuyết về các kỹ thuật khai thác các tập phổ biến như các phương pháp Apriori, FP-tree, IT-tree.

- Tìm hiểu về cơ sở dữ liệu GD có trọng số, trọng số hỗ trợ và các định nghĩa lý thuyết liên quan.

- Tìm hiểu về độ khác nhau của hai tập tương đương Diffset

- Nghiên cứu các thuật toán khai thác các tập phổ biến trên cơ sở dữ liệu GD có trọng số WIT-FWI, WIT-FWI-DIF.

- Cài đặt thực nghiệm để khảo sát kết quả của thuật toán đề xuất: tiến hành khai mẫu trọng số phổ biến tối đại trên các cơ sở dữ liệu chuẩn như: Chess, Mushrooms, BMS1_itemset_mining, Connect

Từ đó đề xuất ra thuật toán khai thác các mẫu trọng số phổ biến tối đại của U.Yun là các tập được đánh trọng số dựa trên CSDL GD có trọng số áp dụng Diffset để tiến hành tính nhanh các độ hỗ trợ. Dựa vào đó để khai thác nhanh các tập được đánh trọng số giúp cho việc khai thác mẫu trọng số phổ biến tối đại được xử lý nhanh hơn. Nhờ áp dụng Diffset, chúng tôi có thể tính toán trọng số hỗ trợ dựa trên sự khác nhau của các tập Tidset, nhằm tối ưu về thời gian xử lý cho khai thác các Mẫu trọng số phổ biến tối đại, cũng như giảm chi phí cho không gian lưu trữ khi khai thác CSDL lớn. Với những cải tiến này, thuật toán đề xuất có hiệu suất tốt hơn so với các thuật toán trước đó với tất cả kết quả. Từ đó ứng dụng các thuật toán này

vào trong thực tiễn.

5.2 Nhận xét ưu điểm và hạn chế

- **Ưu điểm:**

  Trong những cơ sở dữ liệu dày đặc, kích thước của Diffset là nhỏ hơn so với Tidset. Vì vậy, sử dụng Diffset sẽ tiêu tốn ít dung lượng bộ nhớ, không gian lưu trữ giảm đáng kể và do đó cho phép các máy tính nhanh độ hỗ trợ của các itemset.

  Thuật toán phù hợp với tất cả các loại CSDL, nhưng đặc biệt hiệu quả khi khai thác với những CSDL mà mật độ trùng lắp giữa các GD là lớn hoặc vừa được thu thập từ thông tin trạng thái của người chơi trong các game (chứa các nước đi của người chơi), hoặc Mushroom chứa các bản ghi mô tả đặc điểm của các loài nấm khác nhau.

- **Hạn chế:**

  Thuật toán đạt được hiệu quả với cơ sở dữ liệu dày đặc, mật độ trùng lắp giữa các GD lớn, nhưng với cơ sở dữ liệu nhỏ thì thời gian thực thi không có sự khác biệt so với sử dụng Tidset.

  Với những CSDL thưa như CSDL chứa các GD mua hàng ở các siêu thị lớn như BMS1_itemset_mining, thì thuật toán cho hiệu quả tương đương so với những thuật toán đã được đề nghị trước đây.

- **Hướng phát triển**

  - Tiếp tục nghiên cứu cách thức khai thác mẫu trọng số phổ biến tối đại tập được đánh trọng phổ biến hiệu quả hơn.
  - Tiến đến việc khai thác mẫu trọng số phổ biến tập đóng được đánh trọng phổ biến.
  - Nghiên cứu cách thức cập nhật tập kết quả khi CSDL thay đổi.

TÀI LIỆU THAM KHẢO

[1] Agrawal at al. (1993). Mining Association Rule between sets of items in large databases. ACM SIGMOD Record 22 (2) 207-216

[2] Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In: VLDB’94 (pp. 487-499)

[3] Cai, C. H., Fu, A. W., Cheng, C. H., & Kwong, W. W. (1998). Mining association rules with weighted items. In: Proceedingss of international database engineering and applications symposium (IDEAS 98) (pp. 68-77).

[4] Ramkumar, G. D., Ranka, S., & Tsur, S. (1998). Weighted association rules: Model and algorithm. In: SIGKDD’98 (pp. 661-666).

[5] Tao, F., Murtagh, F., & Farid, M. (2003). Weighted association rule mining using weighted support and signficance framework. In: SIGKDD’03 (pp. 661-666)

[6] Wang, W., Yang, J., & Yu, P. S. (2000). Efficient mining of weighted association rules. In: SIGKDD 2000 (pp. 270-274)

[7] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In: SIGMOD (pp. 1-12)

[8] Zaki et al. (1997). New algorithms for fast discovery of association rules.

[9] Vo, B., Coenen, F., Le, B (2013). A new method for mining frequent weighted itemsets based on WIT-trees. Expert systems with applications 40(4), 1256-1264.

[11] Zaki, M. J. (2004). Mining non-redundant association rules. Data Mining andKnowledge Discovery, 9(3), 223–248.

[12] Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. In:

SIGKDD’03 (pp.326–335).

[13] Nguyễn Lâm, 2014 Khai thác Top-rank-k tập được đánh trọng số (Luận văn cao học, Học viện kỹ thuật quân sự).

[14] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In: SIGMOD (pp. 1-12)

[15] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, T. Yiu, MAFIA: a maximal frequent itemset algorithm, IEEE Transactions on Knowledge and Data Engineering 17 (11) (2005) 1490–1504.

[16] U. Yun, Hyeonil Shin, Keun Ho Ryu, EunChul Yoon: An efficient mining algorithm for maximal weighted frequent patterns in transactional databases, Knowledge and Information Systems pages Vol 33, page 53–64 (2012)

[17] U. Yun, K. Ryu, Approximate weighted frequent pattern mining with/without noisy environments, Knowledge Based Systems 24 (1) (2011) 73–82.

[18] U. Yun, An efficient mining of weighted frequent patterns with length decreasing support constraints, Knowledge Based Systems 21 (8) (2008) 741–752.