BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ QUỐC PHÒNG

HỌC VIỆN KỸ THUẬT QUÂN SỰ

VŨ THỊ QUYÊN

KHAI PHÁ TRI THỨC TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN

Chuyên ngành: Hệ thống thông tin

LUẬN VĂN THẠC SĨ KỸ THUẬT

Hà Nội - Năm 2013

BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ QUỐC PHÒNG

HỌC VIỆN KỸ THUẬT QUÂN SỰ

VŨ THỊ QUYÊN

KHAI PHÁ TRI THỨC TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN

Chuyên ngành: Hệ thống thông tin

Mã số: 60 48 05

LUẬN VĂN THẠC SĨ KỸ THUẬT

Hà Nội - Năm 2013

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

HỌC VIỆN KỸ THUẬT QUÂN SỰ

Cán bộ hướng dẫn chính: PGS. TS. Nguyễn Bá Tường

Cán bộ chấm phản biện 1: .....

Cán bộ chấm phản biện 2: .....

Luận văn thạc sĩ được bảo vệ tại:

HỘI ĐỒNG CHẤM BẢO VỆ LUẬN VĂN THẠC SĨ

HỌC VIỆN KỸ THUẬT QUÂN SỰ

Ngày tháng năm 2013

Tôi xin cam đoan:

Những kết quả nghiên cứu được trình bày trong luận văn là hoàn toàn trung thực, của tôi, không vi phạm bất cứ điều gì trong luật sở hữu trí tuệ và pháp luật Việt Nam. Nếu sai, tôi hoàn toàn chịu trách nhiệm trước pháp luật.

TÁC GIẢ LUẬN VĂN

Vũ Thị Quyên

MỤC LỤC

Chương 3
NGHIÊN CỨU CÁC THUẬT TOÁN KHAI PHÁ LUẬT KẾT HỢP TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN

KẾT LUẬN

Tóm tắt luận văn

- Họ tên học viên: **Vũ Thị Quyên**
- Lớp: Hệ thống thông tin Khóa: K23
- Cán bộ hướng dẫn: PGS. TS. Nguyễn Bá Tường
- Tên đề tài: Khai phá tri thức trong cơ sở dữ liệu phân tán
- Tóm tắt: Nghiên cứu tổng quan về khai phá dữ liệu, nghiên cứu một số thuật toán khai phá luật kết hợp trong đó đi sâu vào nghiên cứu khai phá luật kết hợp trong cơ sở dữ liệu phân tán. Nghiên cứu, cài đặt một hệ thống khai phá luật kết hợp trong cơ sở dữ liệu phân tán với hiệu năng cao.

DANH MỤC CÁC KÝ HIỆU VIẾT TẮT

KÍ HIỆU	Ý NGHĨA
AR	Tập luật kết hợp (Association Rules)
CGFI	Tập ứng cử phổ biến toàn cục (Candidate global frequence itemsets)
CSDL	Cơ sở dữ liệu
DB	Cơ sở dữ liệu (Database)
DB_i	Cơ sở dữ liệu ở site i
DMAR	Khai phá luật kết hợp phân tán (Distributed Minning Association Rule)
FDM	Thuật toán khai phá luật kết hợp phân tán nhanh (Fast Distributed Algorithm for Mining Association Rule)
FP-Tree	Cây phổ biến mẫu (Frequent Pattern Tree)
GFI	Tập phổ biến toàn cục (Gobal frequence itemsets)
KDD	Khám phá tri thức từ cơ sở dữ liệu (Knowledge Discovery in Databases)
KPDL	Khai phá dữ liệu
minconf	Độ tin cậy tối thiểu
minsup	Độ hỗ trợ tối thiểu
LFI	Tập phổ biến cục bộ (Lobal frequence itemsets)
LFI^i	Tập phổ biến cục bộ trên site i
Sup	Độ hỗ trợ (support)

DANH MỤC BẢNG

DANH MỤC HÌNH VẼ

MỞ ĐẦU

Trong vài thập niên gần đây, khai phá dữ liệu (KPDL) đã trở thành một trong những hướng nghiên cứu chính trong lĩnh vực khoa học máy tính và công nghệ tri thức. Trong quá trình phát triển đó với hàng loạt nghiên cứu, đề xuất được thử nghiệm và ứng dụng thành công vào đời sống đã chứng tỏ rằng KPDL là một lĩnh vực nghiên cứu ổn định có nền tảng lý thuyết vững chắc.

KPDL bao hàm rất nhiều hướng tiếp cận. Các kỹ thuật trong lĩnh vực này phần lớn là được thừa kế từ lĩnh vực cơ sở dữ liệu (CSDL), học máy (machine learning), trí tuệ nhân tạo (artificial intelligence), ly thuyết thông tin (informaion theory), xác suất thống kê (probability & statistics), và tính toán hiệu năng cao (high performance computing). Các bài toán chủ yếu trong KPDL là phân lớp/dự đoán (classification/prediction), phân cụm (clustering), khai phá luật kết hợp (association rules mining), khai phá chuỗi (sequence mining),…KPDL đã và đang được ứng dụng thành công vào thương mại, tài chính, thị trường chứng khoán, sinh học, y học,…

Khai phá luật kết hợp là một nội dung quan trọng trong KPDL, được khởi xướng từ năm 1993. Cho đến thời điểm này, đã có rất nhiều thuật toán khai phá luật kết hợp được các tác giả đưa ra: Apriori [10, 11], FP-Growth [13],… Thuật toán Apriori đến nay vẫn là một thuật toán khai phá luật kết hợp nổi tiếng. AprioriTID [11, 12] là một mở rộng, cải tiến của phương pháp Apriori cơ bản. Thay vì dựa vào cơ sở dữ liệu thô, AprioriTID biểu diễn bên trong mỗi giao dịch bởi các ứng viên hiện hành. Còn với AprioriHybrid là sự kết hợp giữa Apriori và AprioriTID [11, 12]. Khác với Apriori, FP-Growth không cần phải tạo các ứng

cử viên, không cần lặp lại quá trình duyệt cơ sở dữ liệu ban đầu nhiều lần, nó chỉ sử dụng hai lần quét và dữ liệu được nén gọn lại: cấu trúc FP-Tree [13].

Khi dữ liệu được lưu trữ trên một cơ sở dữ liệu phân tán, thì một thuật toán khai phá dữ liệu phân tán lại là cần thiết để khai phá các luật kết hợp. Khai phá các luật kết hợp trong môi trường phân tán là một vấn đề phải được giải quyết bằng việc sử dụng một thuật toán phân tán mà không cần phải trao đổi dữ liệu thô giữa các bên tham gia. Khai phá luật kết hợp phân tán (DARM: Distributed Association Rule Minning) thì đã được giải quyết bởi nhiều nghiên cứu và cũng đã có rất nhiều thuật toán phân tán đã được đề xuất [14, 15].

Mặc dù đã thu được một số kết quả nhưng hướng nghiên cứu khai phá luật kết hợp trong môi trường phân tán vẫn là một hướng nghiên cứu mới mẻ, thực tế, thú vị và thu hút được nhiều tác giả nghiên cứu. Với mục đích cải tiến độ phức tạp tính toán, cải tiến truyền thông, đưa các thuật toán vào ứng dụng trong thực tế và cải tiến cài đặt để các thuật toán hoạt động với hiệu quả cao hơn.

Bởi vậy, mục đích của luận án này là nghiên cứu tổng quan về khai phá dữ liệu, nghiên cứu một số thuật toán khai phá luật kết hợp trong đó đi sâu vào nghiên cứu, phân tích và đánh giá thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán. Nghiên cứu, cài đặt một hệ thống khai phá luật kết hợp trong cơ sở dữ liệu phân tán với hiệu năng cao.

Nội dung luận văn được trình bày trong 66 trang tài liệu và được chia thành 3 chương:

Chương 1: Tổng quan về khai phá dữ liệu: Giới thiệu tổng quan về quá trình khai phá dữ liệu, kho dữ liệu và khai phá dữ liệu; kiến trúc của một hệ thống khai phá dữ liệu; Nhiệm vụ chính và các phương pháp khai phá dữ liệu.

Chương 2: Khai phá luật kết hợp trong cơ sở dữ liệu: Chương này trình bày tổng quan về luật kết hợp; giới thiệu một số thuật toán khai phá luật kết hợp: tuần tự, song song và phân tán.

Chương 3: Nghiên cứu các thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán: Trong chương này đi sâu vào nghiên cứu chi tiết một số thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán.

Chương 1
TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

1.1. Các khái niệm cơ bản

Dữ liệu (Data): có thể xem là chuỗi các bit, là số, ký tự…mà chúng ta tập hợp hàng ngày trong công việc.

Thông tin (Information): là tập hợp của những mảnh dữ liệu đã được chắt lọc dùng mô tả, giải thích đặc tính của một đối tượng nào đó.

Tri thức (Knowledge): là tập hợp những thông tin có liên hệ với nhau, có thể xem tri thức là sự kết tinh từ dữ liệu. Tri thức thể hiện tư duy của con người về một vấn đề.

Khám phá tri thức từ cơ sở dữ liệu (KDD): là quy trình bao gồm nhiều công đoạn như: xác định vấn đề, tập hợp và chọn lọc dữ liệu, khai thác dữ liệu, đánh giá kết quả, giải thích dữ liệu, áp dụng tri thức vào thực tế.

Tại sao phải khai phá dữ liệu?

Nhà bác học nổi tiếng Karan Sing đã từng nói rằng "*Chúng ta đang ngập chìm trong biển thông tin nhưng lại đang khát tri thức*".

Dữ liệu được thu thập hàng ngày là rất lớn: Từ các cơ sở dữ liệu khổng lồ, từ Internet. Theo các báo cáo của IBM, chỉ có 80% dữ liệu được khai thác, 20% còn lại ẩn trong các cơ sở dữ liệu là những tri thức quý giá.

Khai phá dữ liệu (Data mining): Là một bước trong quy trình khám phá tri thức, nhằm:

- Rút trích thông tin hữu ích, chưa biết, tiềm ẩn trong khối dữ liệu lớn

- Phân tích dữ liệu bán tự động
- Giải thích dữ liệu trên các tập dữ liệu lớn

1.2. Quá trình khai phá dữ liệu

Một quá trình KPDL bao gồm năm giai đoạn chính sau [6]:

(1) Tìm hiểu nghiệp vụ và dữ liệu

(2) Chuẩn bị dữ liệu

(3) Mô hình hóa dữ liệu

(4) Hậu xử lý và đánh giá mô hình

(5) Triển khai tri thức

Quá trình này có thể được lặp lại nhiều lần một hay nhiều giai đoạn dựa trên phản hồi từ kết quả của các giai đoạn sau. Tham gia chính trong quá trình KPDL là các nhà tư vấn và phát triển chuyên nghiệp trong lĩnh vực KPDL.

Tìm hiểu nghiệp vụ và dữ liệu: Nhà tư vấn nghiên cứu kiến thức về lĩnh vực sẽ áp dụng, bao gồm các tri thức cấu trúc về hệ thống và tri thức, các nguồn dữ liệu hiện hữu, ý nghĩa, vai trò và tầm quan trọng của các thực thể dữ liệu. Việc nghiên cứu này được thực hiện qua việc tiếp xúc giữa nhà tư vấn và người dùng. Khác với phương pháp giải quyết vấn đề truyền thống khi bài toán được xác định chính xác ở bước đầu tiên, nhà tư vấn tìm hiểu các yêu cầu sơ khởi của người dùng và đề nghị các bài toán tiềm năng có thể giải quyết với nguồn dữ liệu hiện hữu. Tập các bài toán tiềm năng được tinh chỉnh và làm hẹp lại trong các giai đoạn sau. Các nguồn và đặc tả dữ liệu có liên quan đến tập các bài toán tiềm năng cũng được xác định.

Chuẩn bị dữ liệu: Sử dụng các kỹ thuật tiền xử lý để biến đổi và cải thiện chất lượng dữ liệu để thích hợp với những yêu cầu của các giải thuật học. Phần lớn các giải thuật KPDL hiện nay chỉ làm việc trên một tập dữ liệu đơn và phẳng, do đó dữ liệu phải được trích xuất và biến đổi từ các dạng cơ sơ dữ liệu phân bố, quan hệ hay hướng đối tượng sang dạng cơ sở dữ liệu quan hệ đơn giản với một bảng dữ liệu. Các giải thuật tiền xử lý tiêu biểu bao gồm:

(a) Xử lý dữ liệu bị thiếu/mất: các dữ liệu bị thiếu sẽ được thay thế bởi các giá trị thích hợp.

(b) Khử sự trùng lặp: các đối tượng dữ liệu trùng lặp sẽ bị loại bỏ đi. Kỹ thuật này không được sử dụng cho các tác vụ có quan tâm đến phân bố dữ liệu.

(c) Giảm nhiễu: nhiễu và các đối tượng tách rời (outlier) khỏi phân bố chung sẽ bị loại đi khỏi dữ liệu.

(d) Chuẩn hóa: miền giá trị của dữ liệu sẽ được chuẩn hóa.

(e) Rời rạc hóa: các dữ liệu số sẽ được biến đổi ra các giá trị rời rạc.

(f) Rút trích và xây dựng đặc trưng mới từ các thuộc tính đã có.

(g) Giảm chiều: các thuộc tính chứa ít thông tin sẽ được loại bỏ bớt.

Mô hình hóa dữ liệu: Các giải thuật sử dụng các dữ liệu đã được tiền xử lý trong giai đoạn hai để tìm kiếm các qui tắc ẩn và chưa biết. Công việc quan trọng nhất trong giai đoạn này là lựa chọn kỹ thuật phù hợp để giải quyết các vấn đề đặt ra. Các bài toán được phân loại vào một trong những nhóm bài toán chính trong KPDL dựa trên đặc tả của chúng.

Hậu xử lý và đánh giá mô hình: Các mô hình kết quả của giai đoạn ba sẽ được hậu xử lý và đánh giá trong giai đoạn 4. Dựa trên các đánh giá của người dùng sau khi kiểm tra trên các tập thử, các mô hình sẽ được tinh chỉnh và kết hợp lại nếu cần. Chỉ các mô hình đạt được mức yêu cầu cơ bản của người dùng mới

đưa ra triển khai trong thực tế. Trong giai đoạn này, các kết quả được biến đổi từ dạng học thuật sang dạng phù hợp với nghiệp vụ và dễ hiểu hơn cho người dùng.

Triển khai tri thức: các mô hình được đưa vào những hệ thống thông tin thực tế dưới dạng các môđun hỗ trợ việc đưa ra quyết định.

Mối quan hệ chặt chẽ giữa các giai đoạn trong quá trình KPDL là rất quan trọng cho việc nghiên cứu trong KPDL. Một giải thuật trong KPDL không thể được phát triển độc lập, không quan tâm đến bối cảnh áp dụng mà thường được xây dựng để giải quyết một mục tiêu cụ thể. Do đó, sự hiểu biết bối cảnh vận dụng là rất cần thiết. Thêm vào đó, các kỹ thuật được sử dụng trong các giai đoạn trước có thể ảnh hưởng đến hiệu quả của các giải thuật sử dụng trong các giai đoạn tiếp theo.

1.3. Các kỹ thuật khai phá dữ liệu

Phân lớp dữ liệu [3**]**

Khái niệm phân lớp dữ liệu được Han và Kamber đưa ra năm 2000. Phân lớp dữ liệu là xây dựng một mô hình mà có thể phân các đối tượng thành những lớp để dự đoán giá trị bị mất tại một số thuộc tính của dữ liệu hay tiên đoán giá trị của dữ liệu sẽ xuất hiện trong tương lai. Quá trình phân lớp dữ liệu được thực hiện qua hai bước.

Bước thứ nhất: Dựa vào tập hợp dữ liệu huấn luyện, xây dựng một mô hình mô tả những đặc trưng của những lớp dữ liệu hoặc những khái niệm, đây là quá trình học có giám sát, học theo mẫu được cung cấp trước.

Bước thứ hai: Từ những lớp dữ liệu hoặc những khái niệm đã được xác định trước, dự đoán giá trị của những đối tượng quan tâm.

Một kỹ thuật phân lớp dữ liệu được Han và Kamber đưa ra là cây quyết định. Mỗi nút của cây đại diện một quyết định dựa vào giá trị thuộc tính tương ứng. Kỹ thuật này đã được nhiều tác giả nghiên cứu và đưa ra nhiều thuật toán.

Phân nhóm dữ liệu [3, 4**]**

Phân nhóm là kỹ thuật khai phá dữ liệu tương tự như phân lớp dữ liệu. Tuy nhiên, sự phân nhóm dữ liệu là quá trình học không được giám sát, là quá trình nhóm những đối tượng vào trong những lớp tương đương, đến những đối tượng trong một nhóm là tương đương nhau, chúng phải khác với những đối tượng trong những nhóm khác. Trong phân lớp dữ liệu, một bản ghi thuộc về lớp nào là phải xác định trước, trong khi phân nhóm không xác định trước. Trong phân nhóm, những đối tượng được nhóm lại cùng nhau dựa vào sự giống nhau của chúng. Sự giống nhau giữa những đối tượng được xác định bởi những chức năng giống nhau. Thông thường những sự giống nhau về định lượng như khoảng cách hoặc độ đo khác được xác định bởi những chuyên gia trong lĩnh vực của mình.

Đa số các ứng dụng phân nhóm được sử dụng trong sự phân chia thị trường. Với sự phân nhóm khách hàng vào trong từng nhóm, những doanh nghiệp có thể cung cấp những dịch vụ khác nhau tới nhóm khách hàng một cách thuận lợi. Ví dụ, dựa vào chi tiêu, số tiền trong tài khoản và việc rút tiền của khách hàng, một ngân hàng có thể xếp những khách hàng vào những nhóm khác nhau. Với mỗi nhóm, ngân hàng có thể cho vay những khoản tiền tương ứng cho việc mua nhà, mua xe, … Trong trường hợp này ngân hàng có thể cung cấp những dịch vụ tốt hơn, và cũng chắc chắn rằng tất cả các khoản tiền cho vay đều có thể thu hồi được.

Hồi quy (Regression): Là việc học một hàm ánh xạ từ một tập dữ liệu thành một biến dự đoán có giá trị thực. Nhiệm vụ hồi qui tương tự như phân lớp, điểm khác nhau chính là ở chỗ thuộc tính để dự báo là liên tục chứ không rời rạc [4, 5]. Việc dự báo các giá trị số thường được làm bởi các phương pháp thống kê cổ điểm chẳng hạn như hồi qui tuyến tính. Tuy nhiên, phương pháp mô hình hóa cũng được sử dụng [3, 4].

Ứng dụng của hồi quy là rất nhiều, ví dụ: dự đoán số lượng sinh vật phát quang hiện thời trong khi rừng bằng cách dò tìm vi sóng bằng thiết bị cảm biến từ xa; dự đoán khả năng tử vong của bệnh nhân khi biết các kết quả xét nghiệm chuẩn đoán; dự đoán nhu cầu tiêu thụ một sản phẩm mới bằng một hàm chi tiêu quảng cáo…

Tổng hợp (summarization): Là công việc liên quan đến các phương pháp tìm kiếm một mô tả cô đọng cho tập con dữ liệu [3, 5]. Các kỹ thuật tổng hợp thường được áp dụng trong việc phân tích dữ liệu có tính thăm dò và báo cáo tự động.

Phát hiện sự thay đổi và độ lệch (change and deviation dectection): Nhiệm vụ này tập trung vào khám phá những thay đổi có ý nghĩa trong dữ liệu dựa vào các giá trị chuẩn hay độ đo đã biết trước, phát hiện độ lệch đáng kể giữa nội dung của tập con dữ liệu và nội dung mong đợi. Hai mô hình độ lệch thường dùng là lệch theo thời gian và lệch theo nhóm. Độ lệch theo thời gian là sự thay đổi có nghĩa của dữ liệu theo thời gian. Độ lệch theo nhóm là sự khác nhau giữa dữ liệu trong hai tập con dữ liệu, tính cả trường hợp tập con của đối tượng này thuộc tập con kia, nghĩa là xác định dữ liệu trong một nhóm con của đối tượng có khác nhau đáng kể so với toàn bộ đối tượng [3, 4].

1.4. Các bài toán thông dụng trong khai phá dữ liệu

Trong KPDL, các bài toán có thể phân thành bốn loại chính [7]:

Phân lớp (Classification): Là bài toán thông dụng nhất trong KPDL. Với một tập các dữ liệu huấn luyện cho trước và sự huấn luyện của con người, các giải thuật phân loại sẽ học ra bộ phân loại (classifier) dùng để phân các dữ liệu mới vào một trong những *lớp* (còn gọi là *loại*) đã được xác định trước. Nhận dạng cũng là một bài toán thuộc kiểu Phân lớp.

Dự đoán (Prediction): Với mô hình học tương tự như bài toán Phân lớp, lớp bài toán Dự đoán sẽ học ra các bộ dự đoán. Khi có dữ liệu mới đến, bộ dự đoán sẽ dựa trên thông tin đang có để đưa ra một giá trị số học cho hàm cần dự đoán. Bài toán tiêu biểu trong nhóm này là dự đoán giá sản phẩm để lập kế hoạch trong kinh doanh.

Luật kết hợp (Association Rule): Các giải thuật Tìm luật kết hợp tìm kiếm các mối liên kết giữa các phần tử dữ liệu, ví dụ như nhóm các món hàng thường được mua kèm với nhau trong siêu thị.

Phân cụm (Clustering): Các kỹ thuật Phân cụm sẽ nhóm các đối tượng dữ liệu có tính chất giống nhau vào cùng một nhóm. Có nhiều cách tiếp cận với những mục tiêu khác nhau trong phân loại. Các tài liệu [8, 9] giới thiệu khá đầy đủ và chi tiết về các cách tiếp cận trong Phân cụm. Các kỹ thuật trong bài toán này thường được vận dụng trong vấn đề phân hoạch dữ liệu tiếp thị hay khảo sát sơ bộ các dữ liệu.

1.5. Các cơ sở dữ liệu phục vụ khai phá dữ liệu

Dựa vào những kiểu dữ liệu mà kỹ thuật khai phá áp dụng, có thể chia dữ liệu thành các loại khác nhau.

Cơ sở dữ liệu quan hệ

Đến nay, hầu hết dữ liệu được lưu giữ dưới dạng cơ sở dữ liệu quan hệ. Cơ sở dữ liệu quan hệ là một nguồn tài nguyên lớn nhất chứa những đối tượng mà chúng ta cần khai phá. Cơ sở dữ liệu quan hệ có cấu trúc cao, dữ liệu được mô tả bởi một tập những thuộc tính và lưu trong những bảng. Khai phá dữ liệu trên cơ sở dữ liệu quan hệ chủ yếu tập trung khai phá mẫu. Ví dụ, trong cơ sở dữ liệu của một ngân hàng, ta có thể tìm được những khách hàng có mức chi tiêu cao, ta có thể phân loại những khách hàng này dựa vào quá trình chi tiêu của họ. Cũng với việc phân tích những mục chi tiêu của khách hàng, chúng ta có thể cung cấp một số thông tin của khách hàng đến những doanh nghiệp khác. Giả sử rằng một khách hàng chi mỗi tháng 500 đô la cho thời trang, nếu được phép, ngân hàng có thể cung cấp thông tin về khách hàng này cho những cửa hàng thời trang.

Cơ sở dữ liệu giao tác

Cơ sở dữ liệu giao tác là tập hợp những bản ghi giao dịch, trong đa số các trường hợp chúng là những bản ghi các dữ liệu hoạt động của doanh nghiệp, tổ chức. Với tính phổ biến của máy tính và thương mại điện tử, ngày nay có rất nhiều cơ sở dữ liệu giao tác. Khai phá dữ liệu trên cơ sở dữ liệu giao tác tập trung vào khai phá luật kết hợp, tìm mối tương quan giữa những mục dữ liệu của bản ghi giao dịch. Nghiên cứu sâu về cơ sở dữ liệu giao tác được mô tả chi tiết ở phần sau.

Cơ sở dữ liệu không gian

Cơ sở dữ liệu không gian bao gồm hai phần: Phần thứ nhất là dữ liệu quan hệ hay giao tác, phần thứ hai là thông tin định vị hoặc thông tin địa lý. Những luật kết hợp trên cơ sở dữ liệu không gian mô tả mối quan hệ giữa các đặc trưng

trong cơ sở dữ liệu không gian. Dạng của luật kết hợp không gian có dạng X $\Rightarrow$ Y, với X, Y là tập hợp những vị từ không gian. Những thuật toán khai phá luật kết hợp không gian tương tự như khai phá luật kết hợp nhưng thêm những vị từ về không gian.

Cơ sở dữ liệu có yếu tố thời gian

Giống như cơ sở dữ liệu không gian, cơ sở dữ liệu có yếu tố thời gian bao gồm hai phần: Phần thứ nhất là dữ liệu quan hệ hay giao tác, phần thứ hai là thông tin về thời gian xuất hiện dữ liệu ở phần thứ nhất. Những luật kết hợp có yếu tố thời gian có nhiều thông tin hơn những luật kết hợp cơ bản. Ví dụ, từ luật kết hợp cơ bản {Bia} $\Rightarrow$ {Thuốc lá}, với dữ liệu có yếu tố thời gian chúng ta có thể có nhiều luật: Độ hỗ trợ của luật {Bia} $\Rightarrow$ {Thuốc lá} là 20% từ 9 giờ đến 13 giờ, là 50% trong thời gian 19 giờ tới 22 giờ. Rõ ràng rằng, những người bán lẻ có thể xác định chiến lược để buôn bán tốt hơn. Hầu hết nghiên cứu về lĩnh vực này ngày nay hình thành một hướng khai phá dữ liệu mới gọi là khai phá mẫu lặp liên tục, khai phá tập mục dữ liệu thường xuyên trong cơ sở dữ liệu thời gian.

Cơ sở dữ liệu đa phương tiện

Số lượng trang web đang bùng nổ trên thế giới, web có mặt ở khắp mọi nơi, duyệt web đã là nhu cầu của mọi tầng lớp trong xã hội. Thông tin trên web đang phát triển với tốc độ rất cao, khai phá thông tin trên web (web mining) đã trở thành một lĩnh vực nghiên cứu chính của khai phá dữ liệu, được các nhà nghiên cứu đặc biệt quan tâm. Khai phá dữ liệu web thông thường được chia thành ba phạm trù chính: Khai phá cách dùng web (web usage mining), khai phá cấu trúc web (web structure mining) và khai phá nội dung web (web content mining). Khai phá cách dùng web tập trung vào việc khai phá thông tin của

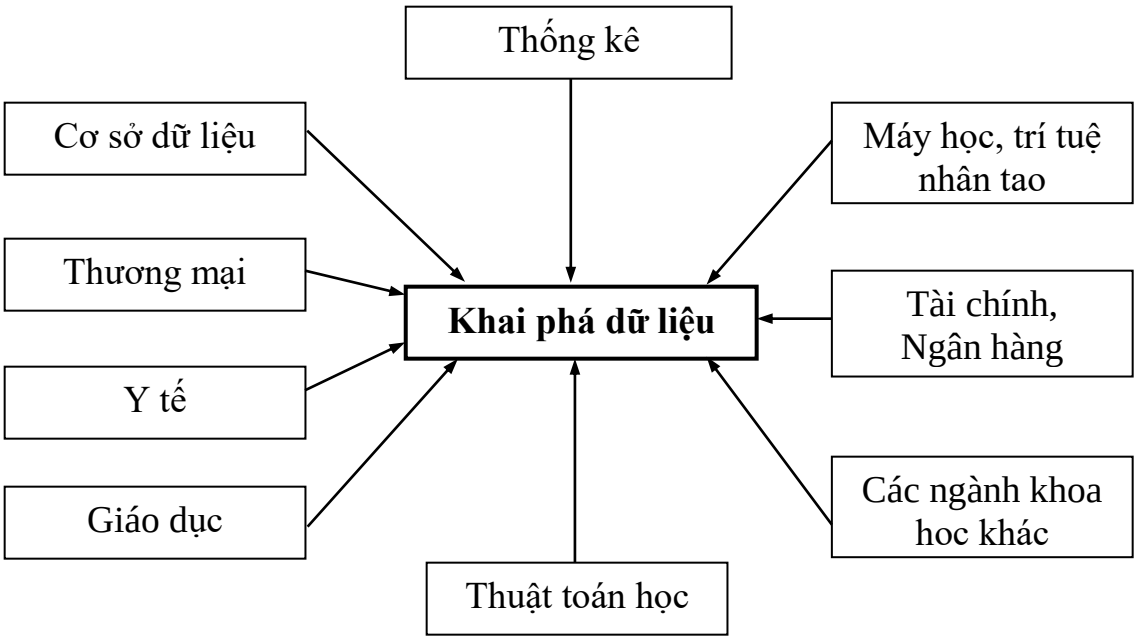
người truy nhập web. Với những thông tin này người khai phá dữ liệu có thể cung cấp những thông tin hữu ích cho người dùng và các nhà kinh doanh.

1.6. Các ứng dụng của khai phá dữ liệu

Khai phá dữ liệu tuy là một lĩnh vực mới nhưng đã thu hút được sự quan tâm của rất nhiều nhà nghiên cứu, nhờ có nhiều những ứng dụng trong thực tiễn, các ứng dụng điển hình, có thể liệt kê như sau:

- Phân tích dữ liệu và hỗ trợ ra quyết định (Analysis & decition support).
- Điều trị trong y học (Medical): mối liên hệ giữa triệu chứng, chuẩn đoán và phương pháp điều trị (chế độ dinh dưỡng, thuốc men, phẫu thuật).
- Phân lớp văn bản, tóm tắt văn bản và phân lớp các trang Web (Text mining & Web mining).
- Tin sinh học (Bio-informatics): Tìm kiếm, đối sánh các hệ gen và thông tin di truyền, mối liên hệ giữa một số hệ gen và một số bệnh di truyền.
- Nhận dạng.
- Tài chính và thị trường chứng khoán (Finance & stock market): Phân tích tình hình tài chính và dự đoán giá cổ phiếu.
- Bảo hiểm (Insurance).
- Giáo dục (Education).

1.7. Khai phá dữ liệu và các lĩnh vực liên quan



Hình 1.1: Một số lĩnh vực liên quan đến khai phá dữ liệu

Phát hiện tri thức và khai phá dữ liệu được coi là trung tâm của nhiều ngành khoa học, nó liên quan đến rất nhiều ngành, nhiều lĩnh vực khác nhau như tài chính, ngân hàng, thương mại, y tế, giáo dục, thống kê, máy học, trí tuệ nhân tạo, cơ sở dữ liệu, thuật toán học, tính toán song song, thu nhận tri thức trong các hệ chuyên gia, quan sát dữ liệu.

Lĩnh vực học máy và nhận dạng mẫu là giống nhau trong khai phá dữ liệu nghiên cứu các lý thuyết và thuật toán của hệ thống trích ra các mẫu và mô hình dữ liệu. Khai phá dữ liệu tập trung vào việc mở rộng các lý thuyết và thuật toán

cho các vấn đề về tìm ra các mẫu đặc biệt, đây được coi là những mẫu hữu ích hoặc tri thức quan trọng tập dữ liệu lớn.

Đặc biệt, phát hiện tri thức và khai phá dữ liệu rất gần gũi với lĩnh vực thống kê, sử dụng các phương pháp thống kê để mô hình dữ liệu và phát hiện các mẫu, luật…, kho dữ liệu và các công cụ xử lý trực tuyến (OLAP – online analytical processing) tập trung vào phân tích dữ liệu đa chiều, tốt hơn SQL trong tính toán và phân tích thống kê đa chiều cũng liên quan chặt chẽ đến khai phá dữ liệu.

Đặc trưng của hệ thống khai phá dữ liệu là nhờ vào các phương pháp thuật toán và kỹ thuật từ những lĩnh vực khác nhau, nhằm mục đích cuối cùng là trích ra tri thức từ dữ liệu trong CSDL khổng lồ.

1.8. Các thách thức trong khai phá dữ liệu

Khai phá dữ liệu ngày càng đóng một vai trò quan trọng trong việc tìm ra các tri thức thực sự có ích, hiệu quả tiềm ẩn trong các khối dữ liệu thông tin khổng lồ vẫn hàng ngày đang được thu thập, lưu trữ để giúp các cá nhân và tổ chức đưa ra được các quyết định chính xác và nhanh chóng. Tuy đã có rất nhiều các giải pháp và phương pháp được ứng dụng trong khai phá dữ liệu nhưng trên thực tế quá trình này vẫn gặp không ít khó khăn và thách thức như:

- Cơ sở dữ liệu lớn
- Số chiều các thuộc tính lớn
- Thay đổi dữ liệu và tri thức có thể làm cho các mẫu đã phát hiện không còn phù hợp
- Dữ liệu bị thiếu hoặc bị nhiễu
- Quan hệ giữa các trường phức tạp

- Giao tiếp với người sử dụng và kết hợp với các tri thức đã có
- Tích hợp với các hệ thống khác

Cơ sở dữ liệu lớn có thể lớn về số lượng các bản ghi, lớn về số lượng các thuộc tính trong CSDL. Số lượng các bản ghi trong CSDL lớn có khi dung lượng tới hàng gigabyte, terabyte; số các thuộc tính trong CSDL có thể rất nhiều và đa dạng. Để giải quyết vấn đề này, người ta thường đưa ra một ngưỡng nào đó cho CSDL bằng các cách như chiết xuất mẫu, xấp xỉ hoặc xử lý song song. Trong CSDL khi mà số các thuộc tính là rất lớn, cùng với số lượng lớn các bản ghi sẽ dẫn đến kích thước độ phức tạp của bài toán tăng lên. Vì vậy, không gian tìm kiếm, không gian trạng thái gia tăng, nhiều mẫu hay mô hình thừa, trùng lặp phát sinh nhiều luật thừa, đây được coi là vấn đề nan giải trong quá trình khai phá dữ liệu. Nhằm giải quyết được những vấn đề trên, phải sử dụng một số các tri thức đã biết trước để loại bỏ và trích lọc ra những dữ liệu thích hợp với yêu cầu của bài toán.

Vấn đề dữ liệu bị thay đổi phụ thuộc theo thời gian, có nghĩa là dữ liệu bị ảnh hưởng và phụ thuộc vào thời điểm quan sát, lấy mẫu, thời điểm khai phá. Kết quả đạt được sau khai phá cũng gây không ít khó khăn cho khai phá dữ liệu, như các mẫu được khai phá ở bước trước, có thể không còn giá trị hay vô nghĩa đối với thời điểm sử dụng, hoặc có thể làm nhiễu hay phát sinh hiệu ứng phụ làm sai lệch kết quả. Để khắc phục được vấn đề này cần phải chuẩn hóa, cải tiến, nâng cấp các mẫu, các mô hình và có thể xem các thay đổi này là mục đích của khai phá và tìm kiếm các mẫu bị thay đổi. Thuộc tính không phù hợp, các bộ giá trị không đầy đủ, bị thiếu giá trị trong các miền thuộc tính đã làm ảnh hưởng rất lớn trong khai phá dữ liệu. Trong quá trình khai phá dữ liệu, khi các hệ thống tương tác với nhau phụ thuộc nhau mà thiếu vắng một vài giá trị nào đó, sẽ dẫn

đến các mẫu không được chính xác, bị thiếu, không đầy đủ. Để giải quyết cho vấn đề này, người ta coi sự thiếu vắng của các dữ liệu này là giá trị ẩn, chưa biết và có thể được tiên đoán bằng một số phương pháp nào đó.

Quan hệ phức tạp giữa các thuộc tính trong CSDL cũng là vấn đề cần được quan tâm. Những bộ thuộc tính có cấu trúc, phân lớp phức tạp, có mối liên hệ phức tạp với nhau trong CSDL đòi hỏi khai phá dữ liệu phải có các giải pháp, các kỹ thuật để có thể áp dụng được, nhận ra được các mối quan hệ này trong quá trình khai phá dữ liệu.

Chương 2
KHAI PHÁ TRI THỨC, KHAI PHÁ LUẬT KẾT HỢP

2.1. Mở đầu

Khai phá dữ liệu là quá trình phát hiện ra các thômg tin có giá trị tiềm ẩn trong CSDL và được xem như là một công đoạn trong quá trình khai thác tri thức. Chức năng của khai phá dữ liệu bao gồm phân lớp, phân cụm, dự đoán và phân tích kết hợp. Ứng dụng khai phá luật kết hợp là một trong những ứng dụng quan trọng của khai phá dữ liệu. Luật kết hợp được giới thiệu đầu tiên vào năm 1993 [19], được sử dụng để xác định các mối quan hệ giữa các tập thuộc tính trong CSDL. Việc xác định các quan hệ này không dựa vào các đặc tính dữ liệu vốn có của chúng (như các phụ thuộc hàm), mà dựa vào sự xuất hiện cùng lúc của các thuộc tính dữ liệu.

Ví dụ 1.1

Trong một hiệu sách lưu lại các phiếu mua sách, người ta phát hiện ra rằng: Trong số những người mua quyển "Các khái niệm và kỹ thuật khai phá dữ liệu" thì có 40% số người đó mua thêm quyển "Hệ quản trị cơ sở dữ liệu", và 25% mua thêm quyển "Kho dữ liệu".

Trong ví dụ trên, tìm được hai luật kết hợp:

- Có 40% số người mua quyển "Các khái niệm và kỹ thuật khai phá dữ liệu" thì đồng thời mua quyển "Hệ quản trị cơ sở dữ liệu".
- Có 25% số người mua quyển "Các khái niệm và kỹ thuật khai phá dữ liệu" thì đồng thời mua quyển "Kho dữ liệu".

Với những quy tắc được khám phá trên, ta có thể sắp xếp các quyển sách có liên quan với nhau ở vị trí gần nhau để giúp cho người mua sách thuận tiện hơn. Những quy tắc đó cũng giúp cho nhà sách có chiến lược kinh doanh tốt hơn.

Luật kết hợp được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau như: Kinh doanh, sản xuất, giao thông, viễn thông, giáo dục, quản lý thị trường, …

Mỗi luật kết hợp được đặc trưng bởi một cặp tỷ lệ, đó là *độ hỗ trợ* và *độ tin cậy*. Thông tin mà luật kết hợp mang lại là rất to lớn và hỗ trợ đáng kể cho quá trình ra quyết định trong kinh doanh cũng như trong nghiên cứu khoa học.

2.2. Các luật trong khai phá tri thức

2.2.1. Luật kết hợp

- Khai phá luật kết hợp: Là tìm các mẫu phổ biến, sự kết hợp, sự tương quan, hay các cấu trúc nhân quả giữa các tập đối tượng trong các cơ sở dữ liệu giao tác, cơ sở dữ liệu quan hệ, và những kho thông tin khác.
- Các ứng dụng: Luật kết hợp có ứng dụng trong nhiều lĩnh vực khác nhau của đời sống như: khoa học, hoạt động kinh doanh, tiếp thị, thương mại, phân tích thị trường chứng khoán, tài chính và đầu tư,...
- Ví dụ về luật kết hợp:
    - ***Bia => Lạc [0,5% ; 60%]***

Luật này có nghĩa: **Nếu** mua bia **thì** mua lạc trong 60% trường hợp. Bia và lạc được mua chung trong 0.5% tổng giao dịch.

- ***Thu nhập***= 60.000.000_max => ***Tài khoản tiết kiệm***= yes [20% ; 100%]

Luật này có nghĩa: **Nếu** thu nhập lớn hơn hoặc bằng 60 triệu một năm **thì** khách hàng có tài khoản tiết kiệm với độ tin cậy là 100%.

Từ các luật kết hợp được trích rút từ chính các cơ sở dữ liệu giao dịch, cơ sở dữ liệu khách hàng mà các siêu thị, các ngân hàng sẽ có chiến lược kinh doanh (sắp xếp các mặt hàng, số lượng các mặt hàng,..), chiến lược tiếp thị, quảng cáo,… để từ đó thúc đẩy hoạt động kinh doanh của mình.

2.2.1.1. Dữ liệu để khai thác

Cho $I = \{i_1, i_2, i_3, \ldots, i_n\}$ là tập bao gồm n mục (Item – còn gọi là thuộc tính - attribute). $X \subseteq I$ được gọi là tập mục (itemset).

$T = \{t_1, t_2, \ldots, t_m\}$ là tập gồm m giao tác (Transaction – còn gọi là bản ghi - record).

R là một quan hệ nhị phân trên I và T (hay $R \subseteq IxT$). Nếu giao tác t có chứa mục i thì ta viết $(i,t) \in R$ (hoặc iRt). Ta sẽ ký hiệu D = (T,I,R) là dữ liệu để khai thác. Về mặt hình thức, D chính là một quan hệ dạng bảng. Về ý nghĩa, một cơ sở dữ liệu là một tập các giao tác (hay giao dịch), mỗi giao dịch t chứa một tập mục $X \subseteq I$.

Ví dụ 1.2:

(Cơ sở dữ liệu dạng giao tác): I ={A, B, C, D, E},

T = {1,2,3,4,5,6}. Thông tin về các giao tác cho ở bảng sau:

Bảng 2.1.a ví dụ về một cơ sở dữ liệu dạng giao tác

D

T	I = {A, B, C, D, E}
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 2.1.a: Cơ sở dữ liệu dạng giao tác

Ý nghĩa của bảng 2.1.a: Trong bảng ta có 6 giao tác được đánh số từ 1 đến 6. Cột bên phải cho biết trong giao tác 1 có các mục A, B, D, E; trong giao tác 2 có các mục B, C, E…Trong một số trường hợp để cho tiện ta biểu diễn bảng 2.1.a dưới dạng bảng nhị phân 0, 1 như sau:

Bảng 2.1.b Ví dụ về một cơ sở dữ liệu dạng giao tác

D

	A	B	C	D	E
t_1	1	1	0	1	1
t_2	0	1	1	0	1
t_3	1	1	0	1	1
t_4	1	1	1	0	1
t_5	1	1	1	1	1
t_6	0	1	1	1	0

Bảng 2.1.b: Cơ sở dữ liệu dạng giao tác

Ý nghĩa của bảng 2.1.b: Trong bảng ta có 6 giao dịch (giao tác) được ký hiệu từ t_1 đến t_6. Các cột A, B, C, D, E là các cột của các mục. Tại giao của t_i và cột A ta ghi 1 nếu t_i có chứa mục A, ngược lại ghi 0. Tương tự cho các cột B, C, D, E. Bảng 2.1.b là bảng nhị phân. Nhìn vào bảng ta thấy giao tác t_1 chứa các mục A, B, D, E và không chứa mục C; giao tác t2 chứa các mục B, C, E và không chứa các mục A, D. Về mặt ngữ nghĩa hai bảng 2.1.a và 2.1.b tương đương nhau, đều biểu thị một tập giao dịch và các mục chứa trong từng giao dịch.

Độ hỗ trợ của tập mục X

Cho dữ liệu D = (T, I, R); X $\subseteq$ I. Gọi T(X) là tập giao tác chứa X.

Định nghĩa 2.1: Độ hỗ trợ (support) của tập mục X, ký hiệu support(X) là tỷ số của số lượng giao tác trong cơ sở dữ liệu D chứa X trên tổng số các giao tác trong cơ sơ dữ liệu D. Hay

$$\text{Support(X)} = \text{Card (T(X))} / \text{Card (T)} = \frac{|T(X)|}{|T|}.$$

Ví dụ 1.3:

Xét dữ liệu trong bảng 2.1.b ta có

Support({A,B,C}) = 2/6 = 1/3 = 0.33 = 33%;

Support({A}) = 4/6 = 2/3 = 0.66 = 66%;

Support({B}) = 6/6 = 1= 100%;

Support({C}) = support({D}) = 4/6 = 0.66 = 66% và support({E}) = 5/6 = 0.83 = 83%

Tập phổ biến

Cho D = (T, I, R); minsup $\in$ (0,1].

Định nghĩa 2.2:Tập mục X $\subseteq$ I được gọi là một tập phổ biến theo ngưỡng minsup (gọi tắt là tập phổ biến) nếu support(X) >= minsup.

Ký hiệu FX(T, I, R, minsup) là tập hợp các tập phổ biến theo ngưỡng minsup:

FX(T, I, R, minsup) = { X $\subseteq$ I | support(X) $\geq$ minsup}

Ví dụ 1.4:

Lấy D = (T, I, R) như trong cơ sở dữ liệu cho ở bảng 2.1.a, và giá trị ngưỡng minsup = 0.5 = 50% sẽ liệt kê tất cả các tập phổ biến (frequent-itemset) như sau:

Bảng 2.2 Các tập phổ biến trong cơ sở dữ liệu ở bảng 2.1.a với độ hỗ trợ tối thiểu 50%

Các tập mục phổ biến FX(T,I,R)	**Độ hỗ trợ support(X) tương ứng**
B	6/6 =1 hay 100%
E, BE	5/6 = 0.83 hay 83%
A, C, D, AB, AE, BC, BD, ABE	4/6 = 0.67 hay 67%
AD, CE, DE, ABD, ADE, BCE, BDE, ABDE	3/6 = 0.5 hay 50%

Bảng 2.2: Các tập phổ biến trong cơ sở dữ liệu ở bảng 2.1.a

Luật kết hợp X => Y

Cho D = (T, I, R) là dữ liệu để khai thác. X , Y $\subseteq$I là các tập mục thỏa mãn điều kiện $X \cap Y = \varnothing$.

Định nghĩa 2.3: Luật kết hợp của X và Y, ký hiệu X=>Y , đây là luật chỉ khả năng xuất hiện Y khi X xuất hiện.

Luật kết hợp có hai độ đo gắn với nó là: độ hỗ trợ và độ tin cậy (confidence) của luật

Độ hỗ trợ của luật kết hợp X => Y

Định nghĩa 2.4: Độ hỗ trợ của luật kết hợp X => Y, ký hiệu support(X => Y) là tỷ số của số các giao tác trong D có chứa X $\cup$ Y trên số tất cả giao tác trong D.

Hay

Support(X => Y) = card (T(X $\cup$ Y))/card(T) = $\frac{|T(X \cup Y)|}{|T|}$; trong đó T(X) là tập giao tác chứa tập mục X.

Ví dụ 1.5:

Cho D = (T, I, R) như trong bảng 2.1.b

Support({A, B} => {C, D}) = card(T(ABCD))/card(T) = 1/6 = 0.166 =16.6%. Support({B} => {A}) = 5/6 = 83%

Độ tin cậy của luật kết hợp X => Y

Định nghĩa 2.5: Độ tin cậy (confidence) của luật X => Y, ký hiệu

confidence(X => Y) là tỷ số các giao tác trong D có chứa X $\cup$ Y trên số các giao tác chứa X. Hay

Confidence(X => Y) = card(T(X $\cup$ Y))/card(T(X)) = $\frac{|T(X \cup Y)|}{|T(X)|}$;

Ví dụ 1.6:

Cho D như trong bảng 2.1.b

confidence({A, B} => {C, D}) = card(T(ABCD))/card(T(AB)) = 1/4 = 0.25

confidence({A} => {B}) = $\frac{|T(AB)|}{|T(A)|}$ = 5/5 = 1= 100%

Về mặt xác suất, độ tin cậy confidence(X => Y) của một luật là xác suất (có điều kiện) xảy ra Y với điều kiện đã xảy ra X.

Confidence(X => Y) = P(Y| X)

Luật kết hợp tin cậy: Một luật được xem là tin cậy nếu độ tin cậy confidence của nó lớn hơn hoặc bằng một ngưỡng minconf $\in (0,1]$ nào đó do người dùng xác định. Ngưỡng minconf phản ánh mức độ xuất hiện của Y khi cho trước X.

Luật kết hợp cần tìm là luật kết hợp thỏa minsup và minconf cho trước. Chúng ta chỉ quan tâm đến các luật có độ hỗ trợ lớn hơn độ hỗ trợ tối thiểu và độ tin cậy lớn hơn độ tin cậy tối thiểu.

Hầu hết các thuật toán khai phá luật kết hợp thường chia thành hai pha:

- *Pha 1*: Tìm tất cả các tập mục phổ biến từ cơ sở dữ liệu D tức là tìm tất cả các tập mục X thỏa mãn support(X) >= minsup.
- *Pha 2*: Sinh các luật tin cậy từ các tập phổ biến đã tìm thấy ở pha 1.

Cho minconf ; X,Y là các tập mục phổ biến tìm thấy trong pha 1 luật kết hợp được sinh từ X, Y có dạng: X => Y và confidence(X=>Y) >= minconf

Ví dụ : Với minsup = 50% và D cho trong 2.1.a ta có các tập phổ biến như sau:

Các tập phổ biến	Độ hỗ trợ (%)
B	100% = (6/6)
E,BE	83% =(5/6)
A,C,D,AB,AE,BC,BD,ABE	67% =(4/6)
AD,CE,DE,ABD,ADE,BCE,ABDE	50% =(3/6)

Lấy minconf=70% thì chúng ta có thể sinh các luật kết hợp sau đây:

Luật kết hợp và độ tin cậy	**Độ tin cậy >= minconf(70%)**
A=>BE & conf(A=>BE) = 100%	Có
B=>AE & conf(B=>AE) = 67%	Không
E=>AB & conf(E=>AB) =81%	Có
AB=>E & conf(AB=>E) =100%	Có
AE=>B & conf(AE=>B) =100%	Có
BE=>A & conf(BE=>A) = 81%	Có

Bảng 2.3: Luật kết hợp sinh từ các tập phổ biến ABE

Một vài tính chất liên quan đến tập phổ biến:

Tính chất 2.1: Nếu X $\subseteq$ Y, với X, Y là các tập mục thì *support(X)* $\geq$ *support(Y)* vì tất cả các giao dịch của D chứa Y thì chứa X.

Tính chất 2.2: Một tập mục X không có độ hỗ tối thiểu trên D nghĩa là *support(X)* < *minsup* thì mọi tập cha Y của X sẽ không phải là tập mục phổ biến vì *support(Y)* $\leq$ *support(X)* < *minsup.*

Tính chất 2.3: Nếu tập mục Y là một tập mục phổ biến trên D, nghĩa là *support(Y)* ≥ *minsup* thì mọi tập con X của Y đều là tập phổ biến trên D vì *support(X)* ≥ *support(Y)* > *minsup.*

Một số tính chất liên quan đến luật kết hợp:

Tính chất 2.4: (Không hợp luật kết hợp)

Nếu có X ⇒ Z và Y ⇒ Z trong D thì không nhất thiết X ∪ Y ⇒ Z là đúng.

Xét trường hợp X ∩ Y = ∅ và các giao dịch trong D hỗ trợ Z nếu và chỉ nếu chúng hỗ trợ mỗi X hoặc Y, khi đó luật $X \cup Y \rightarrow Z$ có độ hỗ trợ 0%.

Tương tự : X ⇒ Y và X ⇒ Z thì không nhất thiết X ⇒ Y ∪ Z là đúng.

***Tính chất 2.5*:** (Không tách luật)

Nếu X ∪ Y ⇒ Z thì X ⇒ Z và Y ⇒ Z chưa chắc xảy ra.

Ví dụ trường hợp Z có mặt trong một giao tác chỉ khi cả hai X và Y cũng có mặt, tức là *support(X∪Y) = support(Z)*, nếu độ hỗ trợ của X và Y đủ lớn hơn *support(X∪Y)*, tức là *support(X) > support(X∪Y)* và *support(Y) > support(X∪Y)* thì hai luật riêng biệt sẽ không đủ độ tin cậy.

Tuy nhiên đảo lại: X ⇒ Y∪Z thì X⇒Y và X⇒Z

***Tính chất 2.6*:** (Các luật kết hợp không có tính bắc cầu)

Nếu X ⇒ Y và Y ⇒ Z, chúng ta không thể suy ra X ⇒ Z.

Ví dụ: Giả sử T(X) ⊂ T(Y) ⊂ T(Z), ở đó T(X), T(Y), T(Z) tương ứng là các giao dịch chứa X,Y,Z, và *confidence(X⇒Y) = confidence(Y⇒Z) = minconf* (độ tin cậy tối thiểu) thế thì: $confidence(X \Rightarrow Z) = minconf^2 < minconf$ vì *minconf* < 1, do đó luật X⇒Z không đủ độ tin cậy.

Tính chất 2.7: Nếu luật X $\Rightarrow$ (L - X) không thỏa mãn độ tin cậy tối thiểu thì không có luật nào trong các luật Y $\Rightarrow$ (L – Y) có độ tin cậy tối thiểu, trong đó Y $\subseteq$ X; X,Y $\subset$ L.

Thật vậy, theo tính chất 2.1, vì Y $\subseteq$ X nên *support(Y)* $\geq$ *support(X)* và theo định nghĩa độ tin cậy, ta có:

$$confidence(Y \Rightarrow (L-Y)) \quad = \quad \frac{\sup\ port(L)}{\sup\ port(Y)} \quad \leq \quad \frac{\sup\ port(L)}{\sup\ port(X)} \quad \langle \quad \min\ conf$$

Nếu luật (L – X) $\Rightarrow$ X thỏa mãn trên D thì các luật (L – Y) $\Rightarrow$ Y với Y $\subseteq$ X và Y $\neq \varnothing$ cũng thỏa mãn trên D.

Bởi vì Y $\subseteq$ X nên (L - Y) $\supseteq$ (L - X), do đó *support(L - Y)* $\leq$ *support(L-X)*

$$\Rightarrow \quad \frac{\sup\ port(L)}{\sup\ port(L-Y)} \quad \geq \quad \frac{\sup\ port(L)}{\sup\ port(L-X)} \quad \geq \quad \min\ conf$$

2.2.1.2. Khai phá luật kết hợp

Phát biểu bài toán:

Đầu vào: - Cho một tập mục $I = \{I_1, I_2, ..., I_m\}$

- Một cơ sở dữ liệu giao dịch D = (T,I,R) có n giao dịch
- Độ hỗ trợ tối thiểu *minsup* và độ tin cậy tối thiểu *minconf*

Đầu ra: - Tập các luật kết hợp R: X $\Rightarrow$ Y sao cho *support(X*$\Rightarrow$*Y)* $\geq$ *minsup* và *confidence(X*$\Rightarrow$*Y)* $\geq$ *minconf*.

Giải quyết bài toán: Bài toán khai phá luật kết hợp được chia thành hai bài toán nhỏ:

- Bài toán 1: Tìm tất cả các tập mục thỏa mãn độ hỗ trợ tối thiểu *minsup* cho trước, hay tập mục phổ biến.

- Bài toán 2: Tìm tất cả những luật kết hợp từ những tập mục phổ biến thỏa độ tin cậy tối thiểu *minconf* cho trước.

Bài toán 1: Ta có thể phân loại các thuật toán tìm tập phổ biến theo hai tiêu chí:

- Phương pháp duyệt qua không gian tìm kiếm.
- Phương pháp xác định độ hỗ trợ của tập mục dữ liệu.

Với phương pháp duyệt qua không gian tìm kiếm được phân làm hai cách: Duyệt theo chiều rộng (Breadth First Search – BFS) và duyệt theo chiều sâu (Depth First Search – DFS).

Duyệt theo chiều rộng là duyệt qua dữ liệu nguyên bản, để tính độ hỗ trợ của tất cả các tập ứng viên có k-1 mục dữ liệu trước khi tính độ hỗ trợ của các tập ứng viên có k mục dữ liệu. Một cơ sở dữ liệu có n mục dữ liệu, trong lần lặp thứ k để tìm những tập k-mục dữ liệu ứng viên, phải kiểm tra tất cả tập $C_n^k = \frac{n!}{k!(n-k)!}$ k-mục dữ liệu.

Duyệt theo chiều sâu, là duyệt qua cơ sở dữ liệu đã được chuyển thành cấu trúc cây, quá trình duyệt được gọi đệ quy theo chiều sâu của cây.

Với cơ sở dữ liệu có n mục dữ liệu, $I = \{x_1, x_2, \ldots, x_n\}$, thì không gian tìm kiếm là tập tất cả các tập con của I, đây là bài toán NP khó, nếu không có phương pháp duyệt thích hợp thì bài toán không giải được khi n đủ lớn.

Phương pháp xác định độ hỗ trợ của tập mục dữ liệu $X \subseteq I$ được phân làm hai cách: Cách thứ nhất: Đếm số giao tác trong cơ sở dữ liệu chứa X. Cách thứ hai: Tính phần giao của các tập định danh giao tác chứa X.

Bài toán 2: Giả sử ta có tập các mục dữ liệu phổ biến L_k, với $L_k = \{ x_{i1}, x_{i2}, \ldots, x_{ik} \}$, luật kết hợp theo ngưỡng độ tin cậy tối thiểu *minconf* với những mục phổ biến này được phát sinh ra bằng cách:

Luật thứ nhất: $\{ x_{i1}, x_{i2}, \ldots, x_{ik-1} \} \Rightarrow \{ x_{ik} \}$, kiểm tra luật này xem có thỏa mãn *minconf* hay không?

Luật thứ hai: $\{ x_{i1}, x_{i2}, \ldots, x_{ik-2}, x_{ik} \} \Rightarrow \{ x_{ik-1} \}$, kiểm tra luật này xem có thỏa mãn *minconf* hay không?

Luật thứ k+1: $\{ x_{i1}, x_{i2}, \ldots, x_{ik-2} \} \Rightarrow \{ x_{ik-1}, x_{ik} \}$, kiểm tra luật này xem có thỏa mãn *minconf* hay không?

Tổng quát: $\forall X \subset L_k$, ta kiểm tra độ tin cậy của luật $X \Rightarrow L_k \setminus X$ có thỏa ngưỡng *minconf* cho trước hay không?

Trong hai bài toán thì bài toán thứ hai là đơn giản hơn, nên hầu hết các nghiên cứu về luật kết hợp tập trung ở bài toán thứ nhất.

2.2.2. Một số thuật toán khai phá luật kết hợp

2.2.2.1. Thuật toán khai phá luật kết hợp tuần tự

2.2.2.1.1. Thuật toán Apriori

Apriori là thuật toán khai phá luật kết hợp do RaKesh Agrawal, Tomasz Imielinski, Anin Sawami đưa ra vào năm 1993, là nền tảng cho việc phát triển những thuật toán sau này. Thuật toán sinh tập mục ứng cử từ những tập mục phổ biến ở bước trước, sử dụng kĩ thuật "tỉa" để bỏ đi tập mục ứng cử không thỏa mãn ngưỡng hỗ trợ cho trước. Thuật toán được trình bày chi tiết trong [10, 11].

Ý tưởng của thuật toán:

- Dùng tập phổ biến kích thước k-1 phần tử để tạo các tập ứng cử k phần tử
- Duyệt cơ sở dữ liệu và đối sánh mẫu để đếm số lần xuất hiện của các tập ứng cử viên trong các giao tác, nếu số lần xuất hiện của tập ứng cử viên lớn hơn hoặc bằng độ hỗ trợ tối thiểu minsup thì là tập phổ biến, ngược lại không phải tập phổ biến.
- Quá trình lặp lại cho đến khi không còn tập phổ biến nào được tạo ra.

Phân tích và đánh giá:

- Thuật toán tạo ra các tập ứng cử viên đồ sộ:
  - 10^4 tập phổ biến kích thước 1 sẽ tạo ra 10^7 tập ứng viên kích thước 2.
  - Để phát hiện một mẫu phổ biến kích thước 100, cần tạo $2^{100} \approx 10^{30}$ ứng viên.
- Duyệt cơ sở dữ liệu nhiều lần: Cần duyệt (n+1) lần, n là chiều dài của mẫu dài nhất.

2.2.2.1.2. Thuật toán Apriori-TID

Như đã đề cập ở phần trên, thuật toán Apriori quét toàn bộ cơ sở dữ liệu trong mỗi giai đoạn để tính độ hỗ trợ. Việc quét toàn bộ cơ sở dữ liệu có thể là không cần thiết đối với tất cả các giai đoạn. Với ý tưởng này, Agrawal đã đề xuất một thuật toán khác, gọi là thuật toán Apriori-TID.

Tương tự thuật toán Apriori, thuật toán Apriori-TID cũng sử dụng tập phổ biến (k-1) phần tử để tạo ra các tập mục ứng cử k phần tử trước khi bắt đầu mỗi giai đoạn.

Điểm khác nhau chủ yếu của thuật toán này so với thuật toán Apriori là: nó không sử dụng cơ sở dữ liệu để tính độ hỗ trợ trong các giai đoạn k > 1. Thay vào đó nó sử dụng các tập mục ứng cử đã sử dụng trong giai đoạn trước. Nhiều thí nghiệm trên nhiều cơ sở dữ liệu chỉ ra rằng thuật toán Apriori cần ít thời gian hơn giải thuật Apriori-TID trong các giai đoạn đầu, nhưng mất nhiều thời gian cho các giai đoạn sau, được mô tả chi tiết trong [11, 12].

Ý tưởng chính của thuật toán Apriori-TID:

Cho dữ liệu D = (T, I, R). Minsup ; Tìm các tập phổ biến.

Input: D, minsup.

Output: Các tập phổ biến F = {X $\subseteq$ I: support(X) $\geq$ minsup}.

Thuật toán Apriori-TID tìm các tập phổ biến với ngưỡng minsup được tiến hành qua hai pha:

- Pha 1: Tìm các tập phổ biến một phần tử F_1:

  $F_1 = \{X \subseteq I: |X| = 1 \ \& \ support(X) \geq minsupp\}$.

- Pha 2: Lặp quá trình tìm F_k có k phần tử được tổ hợp từ 2 phần tử của F_{k-1}:

  $F_k = \{X \subseteq I: |X| = k \ \& \ X = Y \cup Z$ trong đó $Y, Z \in F_{k-1}$ & support(X) $\geq$ minsupp} với k = 2, 3, ... lặp cho đến khi $F_k = \phi$.

Tập tất cả các tập phổ biến $F = F_1 \cup F_2 \cup ... \cup F_{k-1}$

Ví dụ:

Xét D = (T, I, R) như trong bảng 2.1.b ở trên:

D

	A	B	C	D	E
t_1	1	1	0	1	1
t_2	0	1	1	0	1
t_3	1	1	0	1	1
t_4	1	1	1	0	1
t_5	1	1	1	1	1
t_6	0	1	1	1	0

Bảng 2.4: Cơ sở dữ liệu dạng giao tác

Giải với ngưỡng minsup = 0.5 = 50%.

Thuật toán thực hiện bằng hai pha:

Pha 1: Tìm các tập phổ biến 1 phần tử F_1

F_1 = {{A}, {B}, {C}, {D}, {E}}.

Pha 2:

F_2 = {{A,B}, {A,D},{A,E},{B,C}, {B,D}, {D,E}, {C,E}}

F_3 = {{A,B,E}, {A,B,D}, {A,D,E}, {B,C,E}, {A,B,D,E}}

$F_4 = Ø$

$F = F_1 \cup F_2 \cup F_3$. Như vậy trong 32 tập con của I có 17 tập phổ biến.

2.2.2.1.3. Thuật toán Apriori-Hybrid

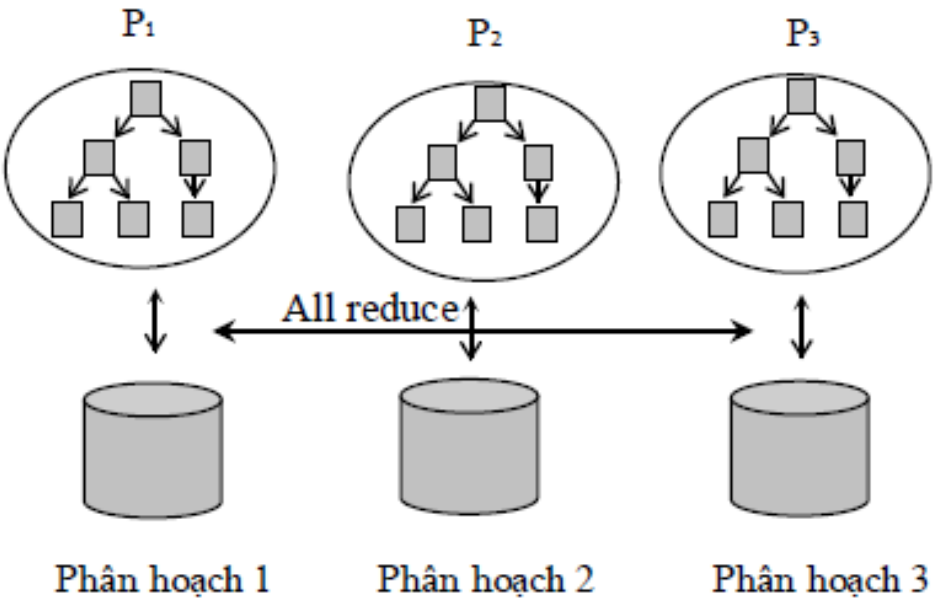
Thuật toán này dựa vào ý tưởng "không cần thiết phải sử dụng cùng một thuật toán cho tất cả các giai đoạn lên trên dữ liệu". Như đã đề cập ở trên, thuật toán Apriori thực thi hiệu quả ở các giai đoạn đầu, thuật toán Apriori-TID thực

thi hiệu quả ở các giai đoạn sau. Phương pháp của thuật toán Apriori-Hybrid là sử dụng thuật toán Apriori ở các giai đoạn đầu và chuyển sang sử dụng thuật toán Apriori-TID ở các giai đoạn sau, được trình bày chi tiết trong [11].

2.2.2.2. Thuật toán khai phá luật kết hợp song song

2.2.2.2.1. Thuật toán Count Distribution (CD)

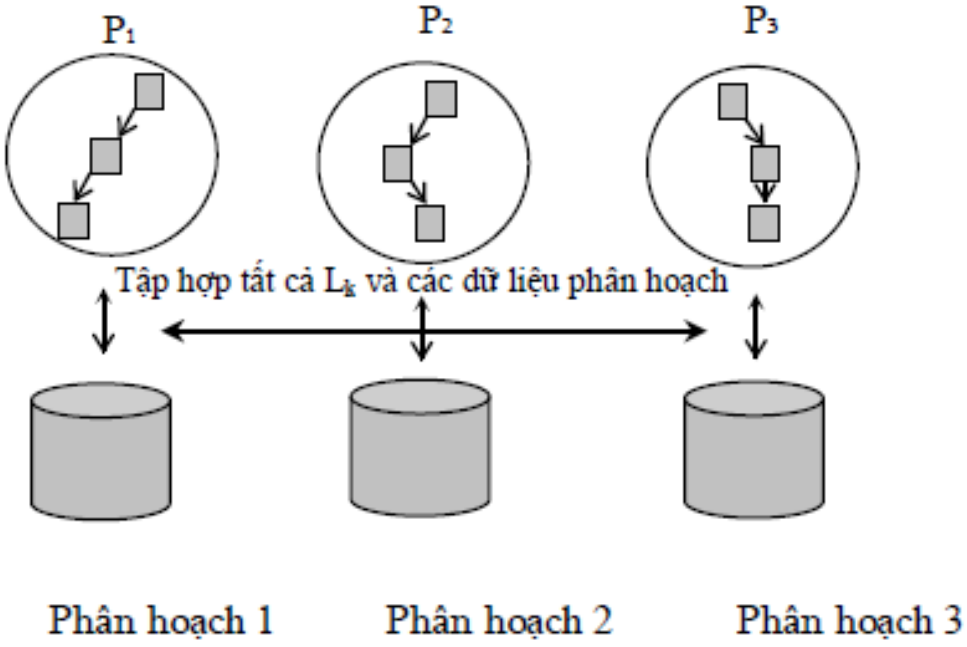
Thuật toán sử dụng kiến trúc không chia sẻ, mỗi bộ xử lý có một bộ xử lý chính và bộ nhớ phụ riêng. Các bộ xử lý này được kết nối với nhau bởi một mạng truyền thông và có thể được truyền thông tin cho nhau bằng việc truyền thông điệp. Dựa trên mô hình song song dữ liệu, dữ liệu được phân hoạch cho các bộ xử lý, mỗi bộ xử lý thực thi công việc giống như thuật toán Apriori tuần tự nhưng thông tin bởi các bộ xử lý trên các phân hoạch dữ liệu của nó.



Hình 2.1: Sơ đồ mô tả thuật toán Count Distribution

2.2.2.2.2. Thuật toán Data Distribution (DD)

Trong thuật toán DD, cơ sở dữ liệu D được phân hoạch thành {D1, D2,…, Dp} nên mỗi bộ xử lý làm việc với một tập dữ liệu không đầy đủ, do đó việc trao đổi dữ liệu giữa các bộ xử lý là cần thiết. Ngoài ra, các tập mục ứng cử cũng được phân hoạch và phân bố cho tất cả các bộ xử lý, mỗi bộ xử lý làm việc với tập mục ứng cử khác nhau.



Hình 2.2: Sơ đồ mô tả thuật toán Data Distribution

2.2.2.2.3. Thuật toán Candidate Distribution

Thuật toán Candidate Distribution thực hiện phân hoạch cả dữ liệu lẫn các tập mục ứng cử. Theo cách này, mỗi bộ xử lý có thể xử lý độc lập. Trong giai đoạn m (m là giá trị heuristic), thuật toán này chia các tập mục phổ biến L_{m-1} cho

các bộ xử lý sao cho mỗi bộ xử lý P_i có thể sinh ra một C_p^i (p > m) duy nhất độc lập với các bộ xử lý khác ($C_p^i \cap C_p^j = Ø$, i≠j).

Trong cùng một thời điểm, dữ liệu được phân chia lại sao cho một bộ xử lý có thể sinh các tập mục ứng cử trong C_p^i một cách độc lập với tất cả các bộ xử lý khác. Tùy vào tính tối ưu của việc phân chia tập mục, một số phần cơ sở dữ liệu có thể có các bản sao trên một số bộ xử lý.

2.2.2.2.4. Thuật toán song song FP-Growth

Dựa vào thuật toán Fp-Tree tuần tự được trình bày trong [13]. Thuật toán này, ta xây dựng một số Fp-Tree cục bộ trong môi trường bộ nhớ phân tán và sử dụng mô hình "*Chủ - Tớ*". Dựa trên chiến lược lập lịch làm việc động trong giai đoạn hợp nhất các mẫu điều kiện cơ sở và giai đoạn khai phá để cân bằng khối lượng công việc trong quá trình thực thi.

Quá trình khai phá tập mục phổ biến song song gồm hai bước chính:

Xây dựng song song cây FP-Tree: Trong bước này cơ sở dữ liệu D sẽ được chia thành p phần cho p bộ vi xử lý, mỗi bộ vi xử lý sẽ xây dựng cây FP-Tree dựa trên cơ sở dữ liệu cục bộ của mình giống như thuật toán tuần tự.

Khai phá song song và sinh tập mục phổ biến: Phương pháp khai phá bao gồm một số giai đoạn sau: Trong giai đoạn đầu, ta xét toàn bộ FP-Tree và tạo ra các mẫu điều kiện cơ sở. Trong giai đoạn tiếp theo, ta tập hợp các mẫu điều kiện cơ sở từ các bộ vi xử lý để xây dựng FP-Tree điều kiện cơ sở (CFPT) cho mỗi mục phổ biến. Giai đoạn cuối cùng là thực thi việc khai phá bằng cách xây dựng đệ qui các mẫu điều kiện cơ sở cho đến khi nó sinh tất cả các tập mục phổ biến.

2.2.2.3. Thuật toán khai phá luật kết hợp phân tán

2.2.2.3.1. Thuật toán khai phá luật kết hợp phân tán nhanh(FDM)

Thuật toán được trình bày chi tiết trong [14].

Cho cơ sở dữ liệu DB chứa D giao dịch, giả sử có một hệ thống phân tán gồm n điểm S_1, S_2,…,S_n và DB được phân mảnh vào n điểm đó {DB_1,DB_2,…,DB_n}, mỗi DB_i có D_i giao dịch. Cho một ngưỡng hỗ trợ tối thiểu s, nhiệm vụ của thuật toán là tìm tất cả tập phổ biến toàn cục L, trong đó L_k là tập phổ biến toàn cục k phần tử.

Trong thuật toán có sử dụng các kỹ thuật: Tạo tập ứng cử, cắt tỉa cục bộ các tập ứng cử, cắt tỉa toàn cục các tập ứng cử.

Tạo tập ứng cử

Nếu tạo tập ứng cử như thuật toán Apriori, ta có:

CA_k = Apriori_gen($L_{(k-1)}$)

Còn trong thuật toán này tập ứng cử được tạo theo công thức sau:

$$L_k \subseteq CG_k = \bigcup_{i=1}^{n} CG_{i(k)} = \bigcup_{i=1}^{n} \text{Apriori\_gen}(GL_{i(k-1)})$$

Trong đó:

CA_k: tập ứng cử k phần tử

$L_{(k-1)}$: tập phổ biến (k-1) phần tử

$GL_{i(k-1)}$: tập phổ biến toàn cục (k-1) phần tử tại S_i

$CG_{i(k)}$: tập ứng cử k phần tử được tạo ra từ $GL_{i(k-1)}$ tại S_i

CG_k: tập ứng cử k phần tử

Với CG_k nhỏ hơn CA_k, việc sử dụng kỹ thuật này sẽ làm giảm số lượng tập ứng cử.

Cắt tỉa cục bộ các tập ứng cử: Với mỗi $X \in CG_{i(k)}$, quét cơ sở dữ liệu DB_i để tính độ hỗ trợ cục bộ. Nếu X không là phổ biến cục bộ thì X sẽ bị loại trừ khỏi tập ứng cử $LL_{i(k)}$, $LL_{i(k)}$ là tập phổ biến cục bộ k phần tử trong $CG_{i(k)}$. Chú ý là

việc cắt tỉa này chỉ loại X ra khỏi tập ứng cử tại S_i, X vẫn có thể là tập ứng cử tại các điểm khác.

Cắt tỉa toàn cục các tập ứng cử:

Giả sử X là một tập ứng cử k phần tử tại vòng thứ k. Với độ hỗ trợ cục bộ của mọi tập con (k-1) phần tử của X là có sẵn tại mọi điểm. Tại DB_i ($1 \le i \le n$), ta ký hiệu $maxsup_i(X)$ là giá trị hỗ trợ cục bộ nhỏ nhất của mọi tập con (k-1) phần tử của X, hay $maxsup_i(X)=min\{Y.sup_i \mid Y \subset X$ và $|Y|=k-1\}$. Ta thấy rằng $maxsup_i(X)$ là cận trên của độ hỗ trợ cục bộ của $X.sup_i$ và tổng những giá trị cận trên ở mọi điểm ký hiệu maxsup(X) là cận trên của độ hỗ trợ toàn cục X.sup.

$X.sup < maxsup(X) = \sum_{i=1}^{n} maxsup_i(X)$

Chú ý là maxsup(X) có thể được tính ở mọi điểm tại thời điểm đầu của vòng thứ k. Giá trị maxsup(X) được sử dụng cho việc cắt tỉa, nếu maxsup(X) < s x D thì X không là một tập ứng cử. Kỹ thuật này được gọi là cắt tỉa toàn cục.

Đánh giá:

Kết quả đánh giá thử nghiệm được so sánh với thuật toán Candidate Distribution (CD) như sau: Số lượng tập ứng cử trong FDM tại mỗi điểm giảm 10 – 25% so với CD. Tổng số lượng bản tin trao đổi trong FDM giảm 10 – 15% so với CD. Thời gian thực hiện của FDM bằng 65 – 75% của CD. Việc giảm số lượng tập ứng cử và số lượng bản tin trao đổi trong FDM là rất đáng kể.

2.2.2.3.2. Thuật toán khai phá phân tán luật kết hợp(DMAR)

Thuật toán được trình bày chi tiết trong [15].

Thuật toán DMAR cho việc khai phá luật kết hợp phân tán sử dụng kỹ thuật meta-learning. Đó là khai phá các tập phổ biến cục bộ mà chúng được sử dụng như là siêu tri thức tại mọi điểm trong hệ thống phân tán và tạo ra các tập ứng cử phổ biến toàn cục từ những siêu tri thức này, sau đó quét cơ sở dữ liệu giao dịch một lần để thu được các tập phổ biến toàn cục. Thuật toán này có hiệu năng khai phá cao hơn và yêu cầu số lượng các giao tiếp thông điệp ít hơn.

Kỹ thuật phân tán cho khai phá luật kết hợp sử dụng meta-learning:

Meta-learning là để thu được kết quả cuối cùng thông qua việc học phổ biến. Tập các tập phổ biến cục bộ L(i) (i=1,..,n) tại mọi điểm có thể được nhìn nhận như là kết quả của quá trình học đầu tiên sử dụng kỹ thuật meta-learning. Có thể tạo các tập ứng cử phổ biến toàn cục C bằng cách sử dụng L(i) và thu được các tập phổ biến toàn cục L bằng cách sử dụng C. Đó là, tạo các tập phổ biến cục bộ L(i) trong cơ sở dữ liệu giao dịch tại mọi điểm và tính toán các tập ứng cử phổ biến toàn cục $C = L(1) \cup L(2) \cup \ldots \cup L(n)$, quét DB_i (i=1,..,n) một lần và thu được độ hỗ trợ của bất kỳ tập mục C-L(i) trong DB_i. Thêm độ hỗ trợ của mọi tập mục trong C tại mọi điểm, thu được tập các tập mục phổ biến toàn cục: $L=\{A \mid sup(A) \geq |DB|*minsup, A \subset C\}$

Thuật toán:

Thuật toán khai phá luật kết hợp phân tán được thực hiện qua vài bước, có sử dụng kỹ thuật meta-learning:

- Bước đầu tiên là khai phá tập các tập phổ biến cục bộ, tập hợp tất cả các tập phổ biến cục bộ tại mọi điểm, sau đó tạo tập các tập ứng cử phổ biến toàn cục.

- Bước hai là quét cơ sở dữ liệu tại mọi điểm, tính toán giá trị hỗ trợ của các tập không phổ biến tại mọi điểm trong tập các tập ứng cử phổ biến toàn cục. Cuối cùng là tính giá trị hỗ trợ của các tập ứng cử phổ biến toàn cục tại mọi điểm, và tạo các tập phổ biến toàn cục.

Thuật toán khai phá phân tán sử dụng một bảng để tập hợp các tập phổ biến cục bộ tại mỗi điểm và tạo các tập ứng cử toàn cục tại mọi điểm.

Itemset	**S_1**	**S_2**	**…**	**S_n**	**S**	**Support σ**
Itemset 1						
Itemset 2						
…						
Itemset m						

Bảng các tập phổ biến

Trong đó:

- S_i: là độ hỗ trợ của tập mục phổ biến cục bộ trong DB_i, i=1,2,..,n
- $S=\sum_{i=1}^{n} S_i$ là độ hỗ trợ của tập mục trong DB
- σ=S/|DB| là độ hỗ trợ của các tập ứng cử phổ biến toàn cục, nếu σ≥minsup thì tập mục là phổ biến toàn cục.

Algorithm: DMAR

Input: DB phân tán {DB_1,DB_2,..,DB_n}, minsup

Output: Tập các tập phổ biến toàn cục L trong DB

(1) For all sites do /* khai phá độc lập tại mỗi điểm*/

{

(2) L(i)=Local-Mining(DBi, minsup) ; /* tạo tập phổ biến cục bộ L(i) */

(3) receive(i,L(i)); /*tập hợp các tập phổ biến cục bộ tại mỗi điểm */

(4) Làm đầy mọi tập mục trong L(i) và độ hỗ trợ tương ứng vào bảng các tập mục phổ biến;

}

(5) For i=l to m do {

(6) Tạo tập các tập ứng cử phổ biến C của mọi điểm từ bảng các tập mục phổ biến nơi mà C(i)={M | S_i=0, S_i tương đương với tập mục M trong bảng}; /*C(i)=C-L(i)*/

(7) send(i,C(i)); /* Truyền tập các tập mục ứng cử phổ biến toàn cục tới điểm thứ i (tri thức meta) */

}

(8) For all sites do (

(9) Quét DB_i một lần và tạo ra độ hỗ trợ của mọi tập trong C(i)

(10) receive(i,C(i)); /*tập hợp độ hỗ trợ của các tập ứng cử phổ biến toàn cục tại mọi điểm */

(11) Làm đầy độ hỗ trợ của các tập mục trong C(i) vào trong cột S_i trong bảng các tập mục phổ biến */

}

(12) L=Φ;

(13) $|DB|=|DB_1|+|DB_2|+ . .. +|DB_n|$; /* tính tổng các giao dịch */

(14) Với mọi tập mục M trong bảng

{

(15) $S=\sum_{i=1}^{n} S_i$;

(16) $\sigma=S/|DB|$;

(17) if $\sigma \geq$ minsup then $L=L \cup \{M\}$;

}

(18) END DMAR

Đánh giá:

Sử dụng bảng các tập mục phổ biến trong thuật toán này cho phép chúng ta dễ dàng thêm bớt các điểm. Khi xóa một điểm i, chỉ cần thiết lập độ hỗ trợ cột S_i trong bảng về 0 và tính lại giá trị S và σ, sau đó sẽ thu được tập phổ biến toàn cục. Khi thêm một điểm, sẽ thêm một cột vào bảng và đăng ký độ hỗ trợ của các tập phổ biến cục bộ. Nếu các tập phổ biến cục bộ tại điểm đó là phần tử thêm mới vào bảng, thì cần phải sử dụng các tập mục mới này để quét tại mọi điểm một lần, sau đó sẽ thu được độ hỗ trợ của các tập mục, tính lại S và σ, sau đó sẽ thu được tập phổ biến toàn cục.

Truyền thông: để khai phá được tập các tập mục phổ biến toàn cục k phần tử trong mỗi vòng lặp, thuật toán FDM cần phải trao đổi độ hỗ trợ của tập các tập ứng cử phổ biến k phần tử và các tập cục bộ, và kết quả là có một số lượng lớn truyền thông và giao thức truyền thông phức tạp. Trong khi đó DMAR truyền và đăng ký tập các tập mục phổ biến cục bộ tại bước (3), với một lần truyền thông tại bước (7) và (10). Vì vậy, thuật toán DMAR có tần suất ít hơn (3 lần) và số lượng truyền thông nhỏ hơn (tập các tập phổ biến cục bộ và các tập không phổ biến trong các tập ứng cử phổ biến toàn cục tại mọi điểm và độ hỗ trợ của chúng).

Khai phá hiệu quả: Với thuật toán FDM, mọi điểm phải được đồng bộ trong quá trình đợi việc tập hợp các độ hỗ trợ của các tập phổ biến tại các điểm tại thời điểm kết thúc mỗi vòng lặp. Trong khi đó với DMAR, quá trình khai phá tại mọi điểm là độc lập và đôi khi có thể off-line với một số lượng nhỏ truyền thông. Vì vậy, tính hiệu quả trong khai phá của thuật toán DMAR là cao hơn so với của thuật toán FDM.

Chương 3
NGHIÊN CỨU CÁC THUẬT TOÁN KHAI PHÁ LUẬT KẾT HỢP TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN

3.1. Giới thiệu

Khai phá luật kết hợp đang trở thành một trong các nhiệm vụ quan trọng của khai phá dữ liệu và nó thu hút được sự quan tâm của rất nhiều các nhà nghiên cứu. Hầu hết các nghiên cứu trước đó về khai phá luật kết hợp đều dựa trên các thuật toán giống Apriori. Chúng được thực hiện trong hai bước [10]:

- Tìm tất cả các tập phổ biến có chứa trong các giao dịch với độ hỗ trợ lớn hơn hoặc bằng ngưỡng hỗ trợ tối thiểu.
- Tạo các luật mong muốn từ các tập phổ biến nếu chúng thỏa mãn ngưỡng độ tin cậy tối thiểu.

Các thuật toán giống Apriori thực hiện lặp lại để đạt được các tập ứng cử (k+1) phần tử từ các tập phổ biến k phần tử. Mỗi một vòng yêu cầu một lần quét cơ sở dữ liệu gốc. Việc lặp lại việc quét cơ sở dữ liệu và đếm tần suất của các tập ứng cử là mất nhiều thời gian và không hiệu quả. Hơn nữa, khi một dữ liệu mới được thêm vào, chúng ta phải chạy lại toàn bộ thuật toán để cập nhật lại các luật. Có nhiều cách khác nhau để cải tiến hiệu năng của Apriori. Đã có nhiều nghiên cứu cải tiến để nâng cao hiệu năng của Apriori [11, 12], nhưng hiện tại vẫn còn nhiều vấn đề cần giải quyết. Gần đây, phương pháp khai phá tập phổ biến dựa trên FP-tree (Frequent Pattern Tree) đã được phát triển bởi nhóm của Han trong [13] và đã đạt được hiệu quả cao hơn so với thuật toán Apriori. Phương pháp này tránh được việc lặp lại quá trình tạo các tập ứng cử, nhưng nó có một số vấn đề:

- Quá trình tạo FP-tree điều kiện là riêng biệt và đệ quy trong quá trình khai phá.

- Quá trình cập nhật FP-tree yêu cầu hai lần quét dữ liệu mới và dữ liệu đã tồn tại.

Các ưu điểm trong kỹ thuật tính toán và công nghệ mạng hiện nay đã dẫn đến việc phân tán nguồn dữ liệu. Các kho dữ liệu thường được chia thành các cơ sở dữ liệu tách rời và đặt ở các điểm khác nhau. Một người dùng có thể cảm thấy thú vị trong việc tạo ra một mô hình toàn cục của dữ liệu, vì vậy các điểm phải trao đổi các thông tin về các mô hình cục bộ của chúng. Tuy nhiên, thông tin trao đổi phải được tạo ra theo cách giảm thiểu truyền thông. Quá trình phân tích những cơ sở dữ liệu phân tán này yêu cầu các phương pháp sao cho: sử dụng đúng đắn các nguồn tài nguyên phân tán, giảm thiểu đến mức thấp nhất yêu cầu truyền thông. Trong chương này nghiên cứu một thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán. Thuật toán này có các đặc điểm:

- Một thuật toán phân tán cho phép tại các điểm quá trình khai phá là độc lập.
- Chỉ yêu cầu hai lần quét cơ sở dữ liệu và hai bước đồng bộ.
- Giảm thiểu chi phí truyền thông cho quá trình khai phá một cơ sở dữ liệu phân tán, điều này làm tăng hiệu năng của quá trình khai phá và quá trình cập nhật các tập phổ biến toàn cục.

3.2. Thuật toán khai phá luật kết hợp FP-Growth

3.2.1. Bản chất

- Khai thác tập phổ biến không sử dụng hàm tạo ứng viên
- Nén CSDL thành cấu trúc cây FP (Frequent Pattern)
- Duyệt đệ quy cây FP để tạo thành tập phổ biến

3.2.2. Qui trình

Bước 1: Thiết lập cây FP

Bước 2: Thiết lập cơ sở mẫu điều kiện (Conditional Pattern Bases) cho mỗi hạng mục phổ biến (mỗi nút trên cây).

Bước 3: Thiết lập cây FP điều kiện (Conditional FP tree) từ mỗi cơ sở mẫu điều kiện

Bước 4: Khai thác đệ qui cây FP điều kiện và phát triển mẫu phổ biến cho đến khi cây FP điều kiện chỉ chứa 1 đường dẫn duy nhất – tạo ra tất cả các tập hợp của mẫu phổ biến.

Ví dụ:

Cho bảng dữ liệu D = (T, I, R) với I = { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, w}; T = {1, 2, 3, 4, 5} và R được cho trong bảng sau:

D

T	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	s	p	w
1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0
2	1	1	1	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0
3	0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1
4	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0
5	1	0	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0

Bảng 3.1: Dữ liệu dạng giao tác

Với minsup = 60% và dùng thuật toán Apriori-TID ta tìm được:

Các tập phổ biến 1 phần tử là F_1 = {{a}, {b}, {c}, {f}, {m}, {p}}.

Các tập phổ biến 2 phần tử là F_2 = { {a, c}, {a, f}, {a, m}, {c,f}, {c, m}, {c, p}, {f,m}}.

Các tập phổ biến 3 phần tử F_3 ={{ a, c, m}, {a, c, f}, {a,f, m}, { c, f, m}}.Các tập phổ biến 4 phần tử F_4 = { a, c, f, m}.

Các tập phổ biến 5 phân tử F_5 = Ø.

Vậy F = $F_1 \cup F_2$ $F_3 \cup F_4$.

Dùng các bước của thuật toán FP-Growth ta cũng tìm được các tập phổ biến như trên.

3.3. Thuật toán khai phá dữ liệu trong cơ sở dữ liệu phân tán

3.3.1. Khái niệm phân tán

Phân tán một CSDL là chia CSDL thành nhiều nhóm và đặt chúng trên các vị trí khác nhau. Như vậy phân tán CSDL gồm hai bước:

Bước một*:* phân mảnh (phân rã) dữ liệu thành các nhóm.

Bước hai*:* đặt các nhóm dữ liệu tại các vị trí theo yêu cầu bài toán.

Trong thực tế chúng ta luôn phải phân tán CSDL vì do nhu cầu thực tế: 'chia để trị' , phạm vi địa lí quá rộng, nếu không phân tán có thể làm sai lệch hoặc dư thừa dữ liệu khi cập nhật thêm sửa xóa trên CSDL.

Thông thường có 3 phương pháp phân rã dữ liệu.

Phân rã ngang

Phân rã ngang là phân rã theo chiều ngang của bảng dữ liệu, giả sử ta có bảng dữ liệu R trên tập thuộc tính U = { A_1, A_2, ..., A_n} phân rã R thành k nhóm R_1, R_2, ..., R_k. Khi đó các bảng Ri là các bảng dữ liệu trên U và R = $R_1 \cup R_2 \cup ... \cup R_k$.

Điều kiện bắt buộc khi phân rã ngang bảng dữ liệu là: $R_i \cap R_j = \emptyset \ \forall \ i \neq j$

Ví dụ:

Ta có bảng dữ liệu D như trong bảng 3.1 ta có thể phân rã ngang D thành 3 nhóm D_1, D_2, D_3 như sau:

D_1

T	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	s	p	w
1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0
2	1	1	1	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0

Bảng 3.2: Dữ liệu dạng nhóm giao tác khi phân rã ngang

D_2

T	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	s	p	w
3	0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1

Bảng 3.3: Dữ liệu dạng nhóm giao tác khi phân rã ngang

D_3

T	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	s	p	w
4	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0
5	1	0	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0

Bảng 3.4: Dữ liệu dạng nhóm giao tác khi phân rã ngang

Rõ ràng ta có $D = D_1 \cup D_2 \cup D_3$ và $D_i \cap D_j = Ø, \forall\ i \neq j$

Phân rã dọc

Phân rã dọc là phân rã theo chiều dọc của bảng dữ liệu, giả sử ta có bảng dữ liệu R trên tập thuộc tính $U = \{ A_1, A_2, ..., A_n\}$ phân rã R thành k nhóm R_1, R_2, ..., R_k. Khi đó các bảng R_i là các bảng dữ liệu trên U_1, U_2, ..., U_k với $U = U_1 \cup U_2 \cup ... \cup U_k$ và R có thể được khôi phục qua phép nối của các R_i. Trong phân rã dọc các U_i không bắt buộc rời nhau.

Ví dụ:

Ta có bảng dữ liệu D như trong bảng 3.1 ta có thể phân rã dọc D thành 3 nhóm D_1, D_2, D_3 như sau:

D_1

T	a	b	c	d	e
1	1	0	1	1	0
2	1	1	1	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	1	0	1	0	1

D_2

T	f	g	h	i	j
1	1	1	0	1	0
2	1	0	0	0	0
3	1	0	1	0	1
4	0	0	0	0	0
5	1	0	0	0	0

D_3

T	k	l	m	n	o	s	p	w
1	0	0	1	0	0	0	1	0
2	0	1	1	0	1	0	0	0
3	0	0	0	0	1	0	0	1
4	1	0	0	0	0	1	1	0
5	0	1	1	1	0	0	1	0

Dễ thấy rằng $D = D_1 * D_2 * D_3$ với * là phép nối bảng

Phân rã hỗn hợp

Phép phân rã hỗn hợp là phép phân rã tổ hợp từ hai phép phân rã ngang và dọc. Đầu tiên phân tã ngang sau phân rã dọc hoặc ngược lại lại.

3.3.2. Khai phá luật kết hợp trong môi trường phân tán

3.3.2.1. Các định nghĩa

Cho tập mục $I=\{i_1, i_2, \ldots, i_m\}$ và một cơ sở dữ liệu giao dịch DB = (T, I, R), ở đó mỗi giao dịch $t \in T$ chứa một tập các mục $X \subseteq I$.

Chúng ta khảo sát quá trình khai phá luật kết hợp trong một môi trường phân tán. Giả sử DB được phân tán ngang có n nhóm đặt trong n vị trí (điểm) $S_1, S_2, \ldots, S_n$. DB được phân mảnh ngang vào n điểm ($DB_1, DB_2, \ldots, DB_n$), trong đó $DB = \bigcup_{i=1}^{n} DB_i$, kích cỡ của DB là D và DB_i là D_i với i=1,2,..,n.

Gọi X.sup là độ hỗ trợ toàn cục của tập X trong DB và $X.sup_i$ là độ hỗ trợ cục bộ của tập X trong DB_i tại điểm S_i.

Với một ngưỡng độ hỗ trợ tối thiểu cho trước minsup, X là một tập phổ biến toàn cục (trên DB) nếu X.sup >= minsup x D và X là một tập phổ biến cục bộ (trên DB_i) nếu $X.sup_i$ >= minsup x D_i.

Nhận xét: Từ định nghĩa X.sup và $X.sup_i$ ta dễ dàng thấy rằng X.sup là số giao dịch trong DB chứa X và $X.sup_i$ là số giao dịch trong DB_i chứa X.

Vì $DB = \bigcup_{i=1}^{n} DB_i$ và các DB_i rời nhau nên ta luôn có:

$$X.sup = \sum_{i=1}^{n} X.sup_i \quad (3.1)$$

Chúng ta ký hiệu GFI là những tập phổ biến toàn cục trong DB và LFI^i là những tập phổ biến cục bộ trong DB_i. Nhiệm vụ chính yếu của thuật toán là tìm ra các

tập mục phổ biến toàn cục GFI, từ đó sinh ra tập các luật kết hợp mong muốn, ký hiệu là AR.

Bổ đề 1: Nếu một tập mục X là phổ biến toàn cục thì tồn tại một S_i (i=1,2,…,n) với X và mọi tập con của nó là phổ biến cục bộ tại S_i.

Chứng minh:

Nếu X không là phổ biến cục bộ tại mọi điểm, điều này tương ứng với $X.sup_i < minsup \times D_i$ với mọi i=1,2,…,n và như vậy theo công thức (3.1) thì X.sup < minsup x D, điều này đồng nghĩa với việc X không phải là phổ biến toàn cục. Trái với giả thiết X là phổ biến toàn cục, như vậy X phải là phổ biến cục bộ tại một số điểm S_i và như vậy X là phổ biến cục bộ tại S_i. Hiển nhiên mọi tập con của X cũng phải là các tập phổ biến cục bộ tại S_i.

Rút ra từ bổ đề 1:

- Một tập X là phổ biến toàn cục thì X sẽ là phổ biến cục bộ tại ít nhất một S_i.
- Nếu X không là phổ biến cục bộ tại mọi S_i thì chắc chắn X không là phổ biến toàn cục.

Bổ đề 2: Nếu tập mục $X \in \bigcap_{i=1}^{n} LFI^i$, thì X và mọi tập con của nó là phổ biến toàn cục.

Chứng minh:

Nếu tập mục $X \in \bigcap_{i=1}^{n} LFI^i$, thì X là phổ biến cục bộ tại mọi điểm, tương ứng với $X.sup_i \geq minsup \times D_i$ với mọi i=1,2,…,n và như vậy thì:

X.sup= $\sum_{i=1}^{n} X.\sup_i \geq$ minsup x $\sum_{i=1}^{n} D_i$ = minsup x D, điều này đồng nghĩa với việc X là phổ biến toàn cục và mọi tập con của X cũng là phổ biến toàn cục.

Rút ra từ bổ đề 2:

- Nếu X là phổ biến cục bộ tại mọi S_i, thì X và mọi tập con của nó là phổ biến toàn cục.

Định nghĩa 1: Với bất kỳ X $\in \bigcup_{i=1}^{n} LFI^i - \bigcap_{i=1}^{n} LFI^i$, thì X là tập ứng cử viên (candidate) phổ biến toàn cục. Kí hiệu CGFI.

Rút ra từ định nghĩa:

X là ứng cử viên phổ biến toàn cục khi:

X là phổ biến cục bộ tại ít nhất một S_i (Nhưng không phải phổ biến cục bộ tại mọi S_i, vì khi đó X là phổ biến toàn cục rồi).

Bổ đề 3: Với bất kỳ X$\in$ CGFI, nếu $\sum_{i=1}^{n} X.\sup_i \geq \min\ \sup \times D$ thì X là phổ biến toàn cục.

Dựa vào bổ đề 1, các tập phổ biến toàn cục có thể được đưa ra từ mọi tập phổ biến cục bộ, mà là tập con của hợp của các tập phổ biến cục bộ. Như vậy, giảm thiểu chi phí truyền thông và các bước dồng bộ thông qua việc gửi ít hơn các tập ứng cử phổ biến toàn cục sẽ là vấn đề chính cho việc năng cao hiệu quả của thuật toán khai phá luật kết hợp phân tán. Từ bổ đề 2, nếu các tập mục X là phổ biến cục bộ tại mọi điểm thì X phải là phổ biến toàn cục. Từ bổ đề 3, nếu điều kiện của bổ đề 2 không được thỏa mãn, nhưng độ hỗ trợ của X lớn hơn hoặc bằng độ hỗ trợ toàn cục thì X vẫn là phổ biến toàn cục. Nếu độ hỗ trợ của X nhỏ hơn độ hỗ trợ toàn cục, chúng ta phải tính lại độ hỗ trợ của X tại các điểm mà nó

không là phổ biến cục bộ để quyết định tính chất của nó. Điều quan trọng là trong thuật toán này yêu cầu hai bước đồng bộ. Bước đồng bộ thứ nhất thực hiện sau khi khai phá các tập phổ biến cục bộ tại mọi điểm, mỗi điểm gửi tập LFI^i tới điểm chính. Sau quá trình trích lọc các tập phổ biến toàn cục theo các bổ đề và tính chất ở trên, điểm chính gửi toàn bộ các tập ứng cử phổ biến toàn cục tới mọi điểm. Mỗi điểm sẽ tính toán độ hỗ trợ của các tập ứng cử phổ biến toàn cục và gửi chúng tới điểm chính, đây chính là quá trình đồng bộ thứ hai.

3.3.2.2. Thuật toán

Trong thuật toán này, quá trình khai phá các tập mục phổ biến cục bộ LFI^i tại mọi điểm S_i với i=1,2,…,n sử dụng thuật toán FP-Growth với đầu vào là cây FP-tree được xây dựng từ cơ sở dữ liệu DB_i.

Giải thuật chính được tham khảo tại [16].

Giải thuật chính:

Input: DB_i (i=1,2,..,n), minsup, minconf

Output: AR (Tập các luật kết hợp)

B1: /**Khai phá LFI^i tại từng S_i*/*

For i=1 to n do

{

+ Xây dựng Prefix-tree^i

+ Khai phá LFI^i dựa trên FP-tree^i (FP-Growth)

+ Gửi LFI^i tới main site

}

B2: */*Khai phá GFI và CGFI*/*

*/*Tại main site thực hiện*/*

+ Tính GFI = $\bigcap_{i=1}^{n} LFI^i$

CGFI = $\bigcup_{i=1}^{n} LFI^i$ - $\bigcap_{i=1}^{n} LFI^i$

+ Với mỗi X $\in$ CGFI thực hiện:

If $\sum_{i=1}^{n} X.\sup_i \geq \min \sup xD$ then

{

GFI = GFI $\cup$ {X}

CGFI = CGFI – {X}

}

+ Gửi CGFI tới mọi S_i

B3: */*Tính lại độ hỗ trợ cục bộ X.sup$_i$ của mọi X $\in$ CGFI*/*

*/*Tại từng Site S_i thực hiện*/*

For i=1 to n do

{

+ Với mỗi X $\in$ CGFI, đếm lại giá trị X.sup$_i$

+ Gửi X.sup$_i$ tới main site

}

B4: */*Tính độ hỗ trợ toàn cục X.sup của mọi X $\in$ CGFI*/*

*/*Tại main site thực hiện*/*

+ Với mỗi X $\in$ CGFI thực hiện:

If *X.sup* = $\sum_{i=1}^{n} X.\sup_i \geq \min \sup xD$ then

{

$$GFI = GFI \cup \{X\}$$

}

B5: /* *Tạo tập luật kết hợp AR từ tập các tập mục phổ biến GFI*/*

/* *Tại main site thực hiện* */

+ Tạo tập luật AR bằng cách gọi thủ tục GenRules(GFI, minconf);

Giải thuật xây dựng FP-tree^i:

Input: DB_i (i=1,2,..,n), minsup

Output: FP-tree^i

1. Duyệt DB_i lần đầu để thu được tập 1-phần tử phổ biến 1-itemset F, sắp xếp F theo thứ tự giảm dần của độ hỗ trợ, ta thu được danh sách L.
2. + Tạo nút gốc R của FP-tree với nhãn là "null".
   + Tạo bảng Header có |F| dòng, đặt mọi node-link trỏ đến null.
3. For each giao tác $T \in DB_i$
   { // Duyệt DB_i lần 2
   + Chọn các item phổ biến của T đưa vào P;
   + Sắp các item trong P theo trật tự L;
   + Call Insert_Tree(P, R);
   }

Procedure Insert_Tree(P, R)

1. Đặt P=[p|P-p] , với p là phần tử đầu tiên và P-p là phần còn lại của danh sách.
2. if (R có một con N sao cho N.item-name=p)

{

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;

}

3. //Tăng biến count của p trong bảng Header thêm 1

H[p].count ++;

If ((P-p) != null)

{

Call Insert_Tree(P-p, N) ;

}

Giải thuật FP-Growth:

Input: FP-tree^i, minsup

Output: LFI^i

Procedure FP_Growth(FP-tree^i, α)

1. LFI^i=Ø;
2. If (FP-tree^i chỉ chứa một đường dẫn đơn P)

{

for each tổ hợp β của các nút trong P do

{

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

support_count(p) = minsup các nút trong β

$LFI^i = LFI^i \cup p$;

}

}else{

for each a_i trong Header của *FP-tree*^{i} do

{

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

support_count (β)=a_i*.support_count;*

$LFI^i = LFI^i \cup \beta$;

Xây dựng cơ sở có điều kiện của β;

Xây dựng FP-Tree có điều kiện *FP-tree*$^i \beta$ của β;

If (*FP-tree*^{i} β !=Ø)

{

call FP_Growth(*FP-tree*^{i} , β);

}

}

}

Giải thuật tạo luật kết hợp GenRules:

Input: GFI, minconf

Output: AR

Procedure GenRules(GFI, minconf)

forall tập mục phổ biến $l_k \in$ GFI, $k \geq 2$ do

call Gen(l_k, l_k);

Procedure Gen(l_k: k-itemset phổ biến; a_m: m-itemset phổ biến)

A = {(m - 1)-itemset a_{m-1} | $a_{m-1} \subset a_m$};

forall $a_{m-1} \in$ A do

{

conf = *sup*(l_k)/*sup*(a_{m-1});

if (*conf* ≥ *minconf*) then

{

Xuất ra luật a_{m-1} => ($l_k - a_{m-1}$);

if ((m - 1) > 1)

{

call Gen(l_k, a_{m-1});

}

}

}

3.3.3. Đề xuất phương pháp cài đặt

3.3.3.1. Xây dựng cấu trúc lưu trữ cây FP-tree

Cấu trúc lưu trữ cây FP-tree bao gồm các lớp:

- Myheadertable: là lớp chứa tập 1-phần tử có độ hỗ trợ lớn hơn minsup,và có khả năng trỏ đến phần tử cùng tên cuối cùng được xây dựng trên cây.
    - Có thuộc tính là một mảng các phần tử node.
- Mynode: bao gồm các trường:

- Name: tên của node.
- Count: số lần duyệt qua- đối với node trên cây hoặc là độ hỗ trợ nếu là node trong headertable.
- Parents: node cha của nó, điều này đảm bảo có thể lần ngược trở lên gốc từ node lá.
- Node: phần tử cùng tên xuất hiện trước nó trong quá trình xây dựng cây.

- Mytree:
  - Roof: gốc của mỗi cây, cây to nhất có roof là node mang giá trị tên là “roof”, count =1.
  - Childtree[]: mảng chứa các nhánh của cây đi ra từ node roof.

3.3.3.2. Cài đặt thuật toán xây dựng cây FP-tree

Từ cấu trúc xây dựng cây đã trình bày ở trên, để xây dựng cây FP-tree chúng ta phải cài đặt một số hàm chính sau (các hàm khác có sẵn trong thư viện của C++):

- createHeadertable: Tạo bảng headertable.
- addtrans: Đưa các phần tử trong danh sách vào cây.

3.3.3.3. Cài đặt thuật toán khai phá dữ liệu trên cây FP-tree (FP-Growth)

Để khai phá dữ liệu trên cây FP-tree, chúng ta phải cài đặt một số hàm chính sau:

- search_frequence

- Set_header_table: Xây dựng
- SplitString
- HaiChieu
- Create_FPTree: Xây dựng cây FP-tree
- sup_item
- Clobal
- sup_clobal
- sapxep
- Find_frequence: Tìm tập phổ biến

3.3.3.4. Cài đặt thuật toán sinh luật kết hợp

Để cài đặt thuật toán sinh luật kết hợp, chúng ta xây dựng một số lớp sau:

- ItemSet: Là lớp chứa tập các phần tử và độ hỗ trợ của tập các phần tử đó. Bao gồm các trường:
    - m_items: Lưu các phần tử
    - count: Số lượng các phần tử
    - m_support: Lưu độ hỗ trợ
- ARSetOfRules: Lưu trữ các luật. Bao gồm các trường:
    - m_premise: Lưu phần tiền đề
    - m_consequence: Lưu phần kết quả
    - m_support: Lưu độ hỗ trợ
    - m_confidence: Lưu độ tin cậy
    - next: Con trỏ

Một số hàm chính trong thuật toán:

- genrules

- ruler
- add

3.3.3.5. Cài đặt thuật toán chính

Thuật toán chính được cài đặt qua các bước sau:

- Bước 1: tính toán LFI^i trên từng site S_i. Quá trình này có thể thực hiện song song, diễn ra đồng thời trên từng site.
  - Sau khi tính toán các S_i gởi danh sách các LFI^i cũng như độ hổ trợ của chúng lên site chính. Trong chương trình quy định site chính là site S_0.
- Bước 2: tuần tự, chỉ có một site S_0 thực hiện.
  - Dựa vào bổ đề 2, site S_0 tổng hợp được GFI.
  - Tính toán tập CGFI dựa theo định nghĩa 1.
  - Trong tập CGFI: duyệt tìm phần tử GFI theo bổ đề 3.
  - Gởi tập CGFI còn lại cho các site S_i.
- Bước 3: song song trên tất cả các site.
  - Tính toán độ hổ trợ của mỗi phần tử trong tập CGFI mà site chính trả về, gởi sup của các phần tử đó trở lại site chính.
- Bước 4: tuần tự, chỉ thực hiện trong một site chính.

- Tổng hợp tất cả các sup của các phần tử CGFI, xét lại một lần nữa xem có phần tử nào là GFI không.

➢ Bước 5: tuần tự, chỉ thực hiện trong một site chính

- Tạo tập luật kết hợp AR từ tập GFI

Giả mã:

```
#pragma omp parallel
{
  #pragma omp for
      For each site
            Xây dựng cây.
            Tìm LFI^i.
            Gửi LFI^i lên site chính.
   #pragma omp barrier
      If (tid==0) // site chính
            Tính GFI.
            CGFI ➔ cập nhật GFI
            Gửi CGFI về mỗi site con
    #pragma omp for
      For each site Si
            For each X là phần tử trong CGFI
```

Tính độ hỗ trợ trên site đó và gửi về site chính.

#pragma omp barrier

If (tid==0)//site chinh

Tính tổng sup mà site con gửi về

Cập nhật GFI.

Tạo tập luật kết hợp AR từ tập GFI

}

3.4. Giao diện chương trình và các chức năng chính.

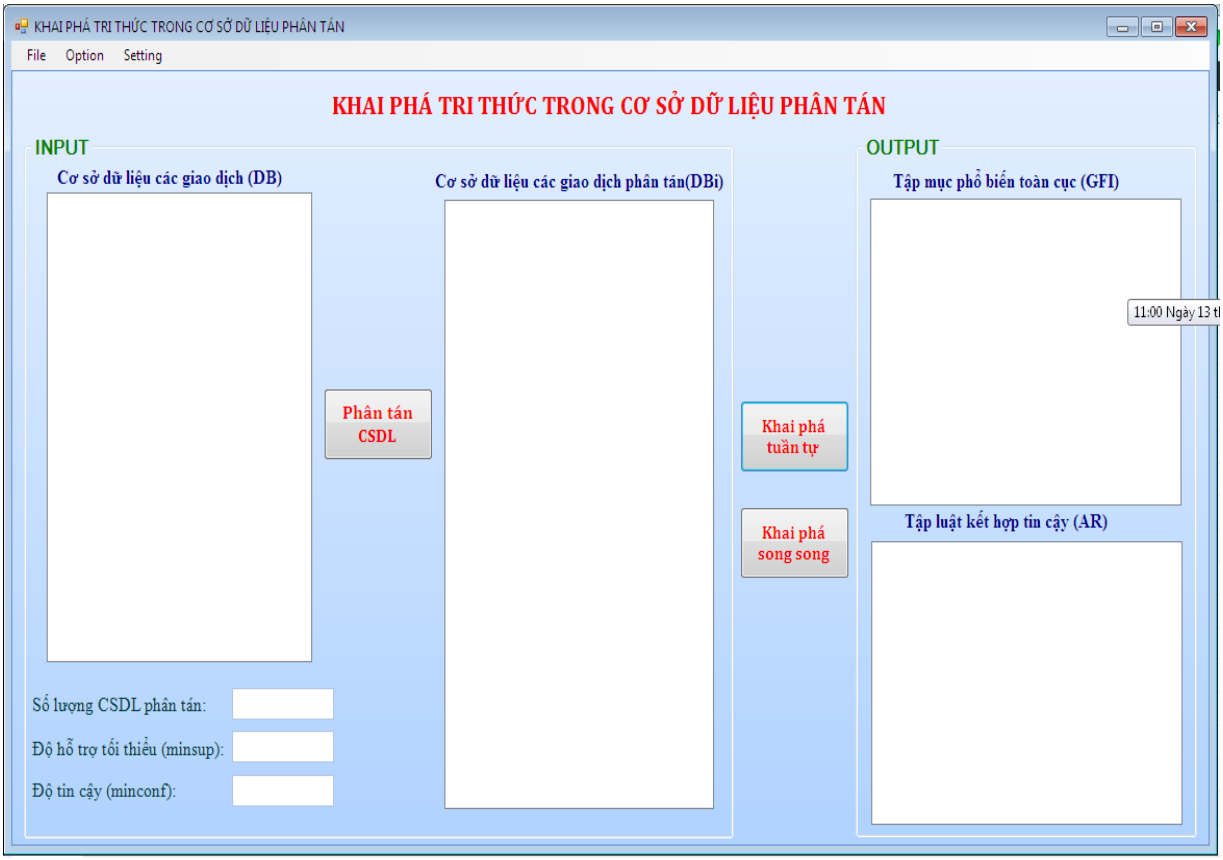
Chương trình được xây dựng để mô phỏng quá trình khai phá tri thức trong môi trường CSDL phân tán, mà cụ thể ở đây là tìm các luật tin cậy từ các CSDL chứa các giao tác.

Trên thực tế, với môi trường CSDL phân tán, các CSDL chứa các giao tác được phân tán trên các máy khác nhau. Việc tính toán các tập mục phổ biến cục bộ được thực hiên độc lập trên mỗi máy, sau khi tính toán xong các máy này sẽ gửi kết quả lên máy chủ chính. Thông thường, với mô hình tính toán song song trên các bộ nhớ phân tán như vậy người ta thường sử dụng chuẩn hỗ trợ MPI (Message Passing Interface). Tuy nhiên, do điều kiện không cho phép nên trong khuôn khổ luận văn, tác giả sử dụng chuẩn OpenMP (Open MultiProcessing) hỗ trợ lập trình song song trên mô hình chia sẻ bộ nhớ chung (luận văn sử dụng một máy tính đơn) để song song hóa bài toán trên.

Chương trình được xây dựng trên ngôn ngữ Visual C++, sử dụng thư viện hỗ trợ lập trình song song "*omp.h".* Để tiện cho việc so sánh kết quả của thuật toán khai phá song song với thuật toán khai phá tuần tự (sử dụng thuật toán

FPGrowth), chương trình thực hiện việc đọc dữ liệu đầu vào (CSDL các giao tác) tại một file input, sau đó để mô phỏng quá trình tính toán song song trên các site phân tán, chương trình sẽ thực hiện phân tán (ngẫu nhiên) các giao tác này vào các CSDL khác nhau (số lượng CSDL phân tán do người dùng quy định).

Giao diện chính của chương trình như sau:

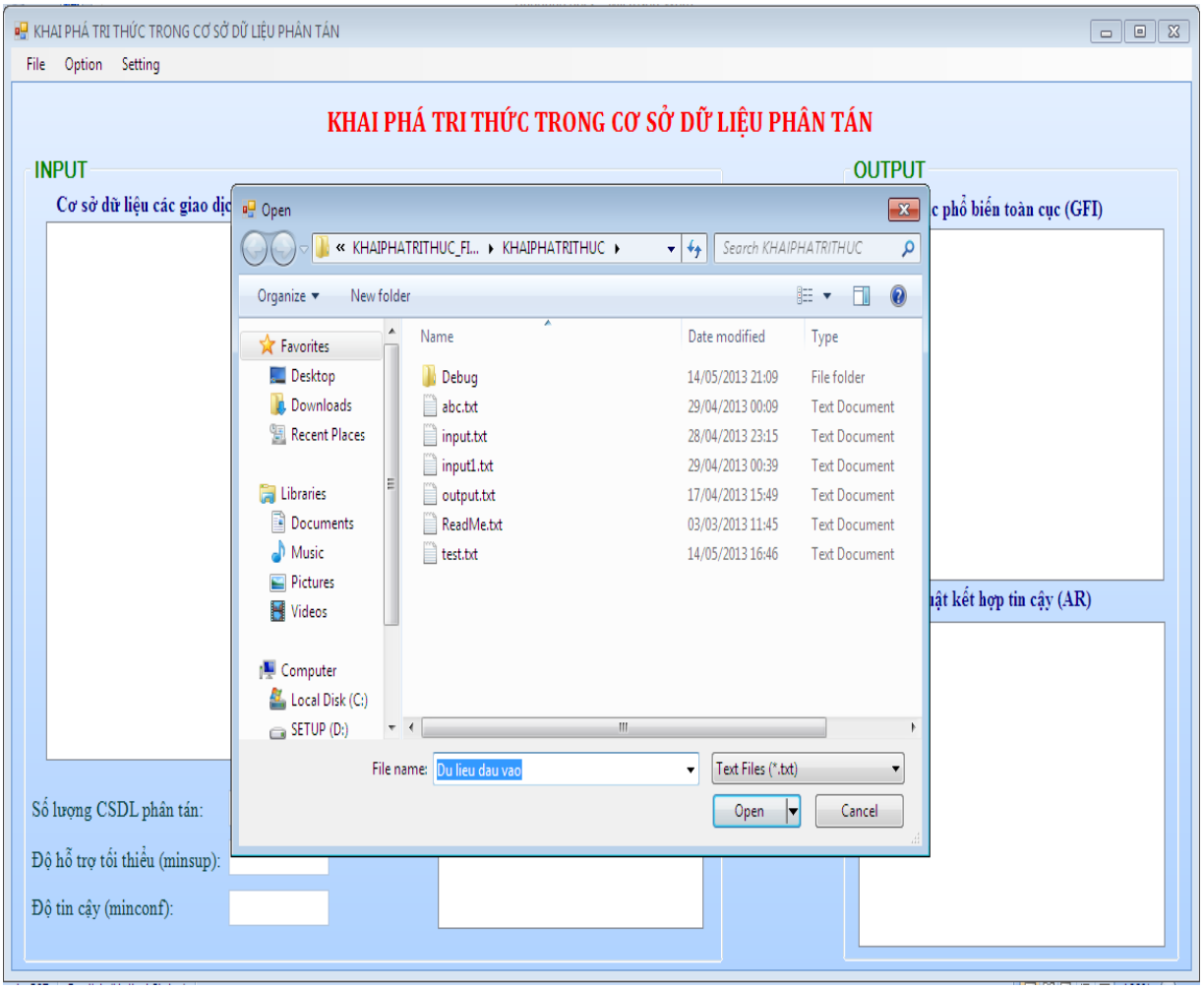


Hình 3.1 Giao diện chính của chương trình

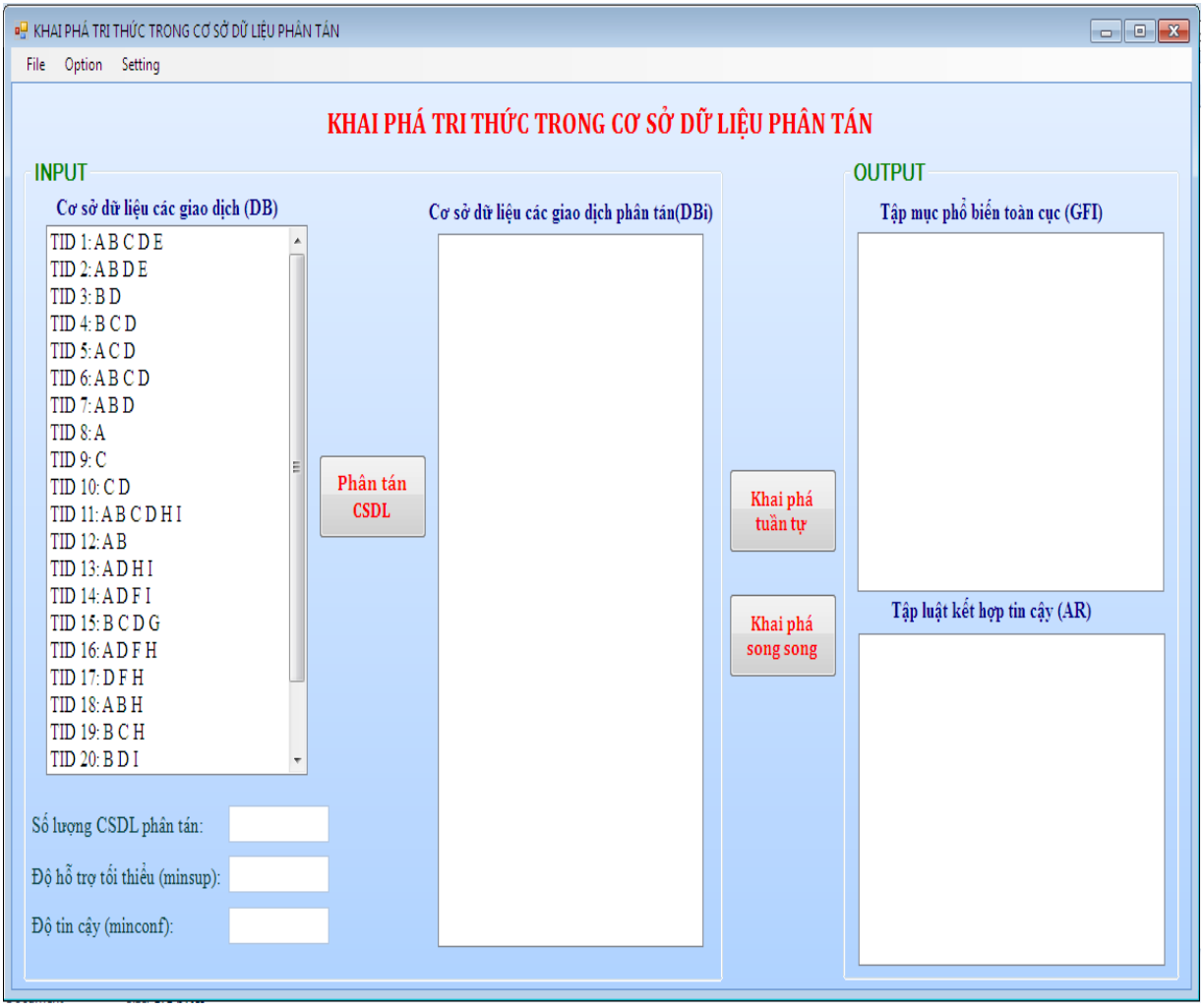
Dưới đây là một số chức năng chính của chương trình.

3.4.1 Mở file chứa CSDL các giao tác

Vào menu *File/ Open* và tìm đến file chứa CSDL (định dạng *.txt)



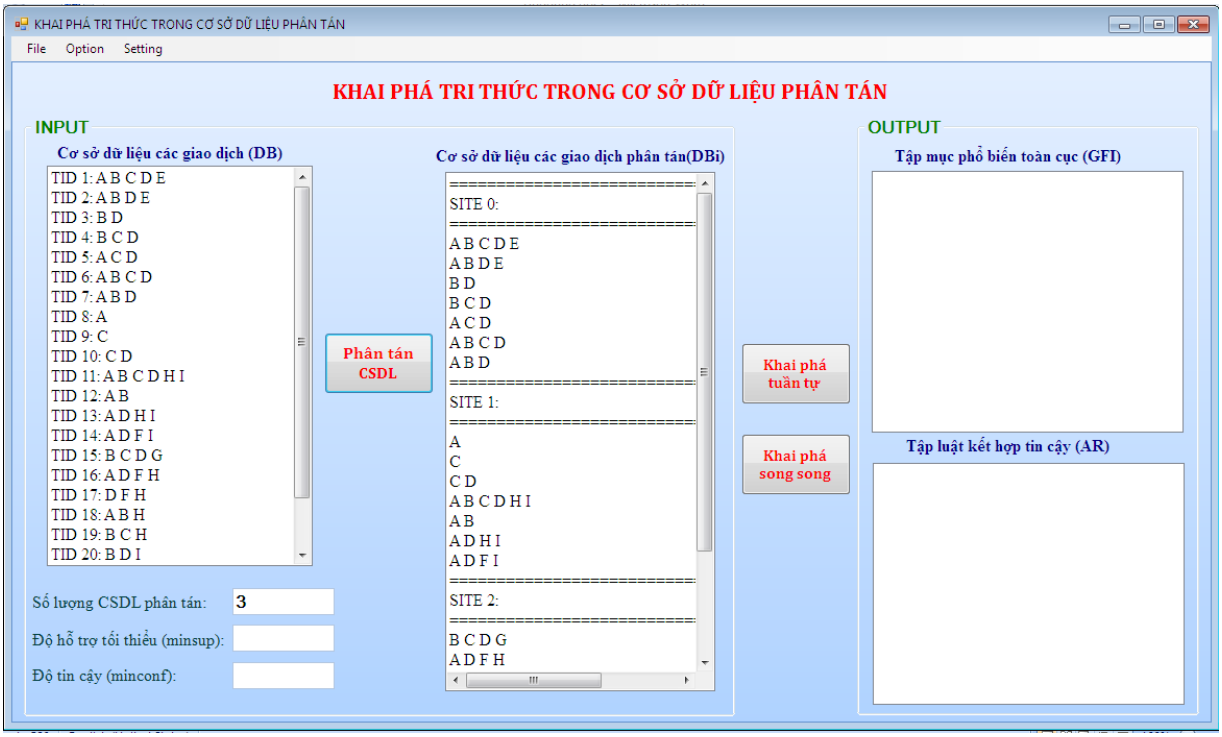
Hình 3.2 Giao diện mở file



Hình 3.3 Nội dung file đã mở

3.4.2 Phân tán CSDL

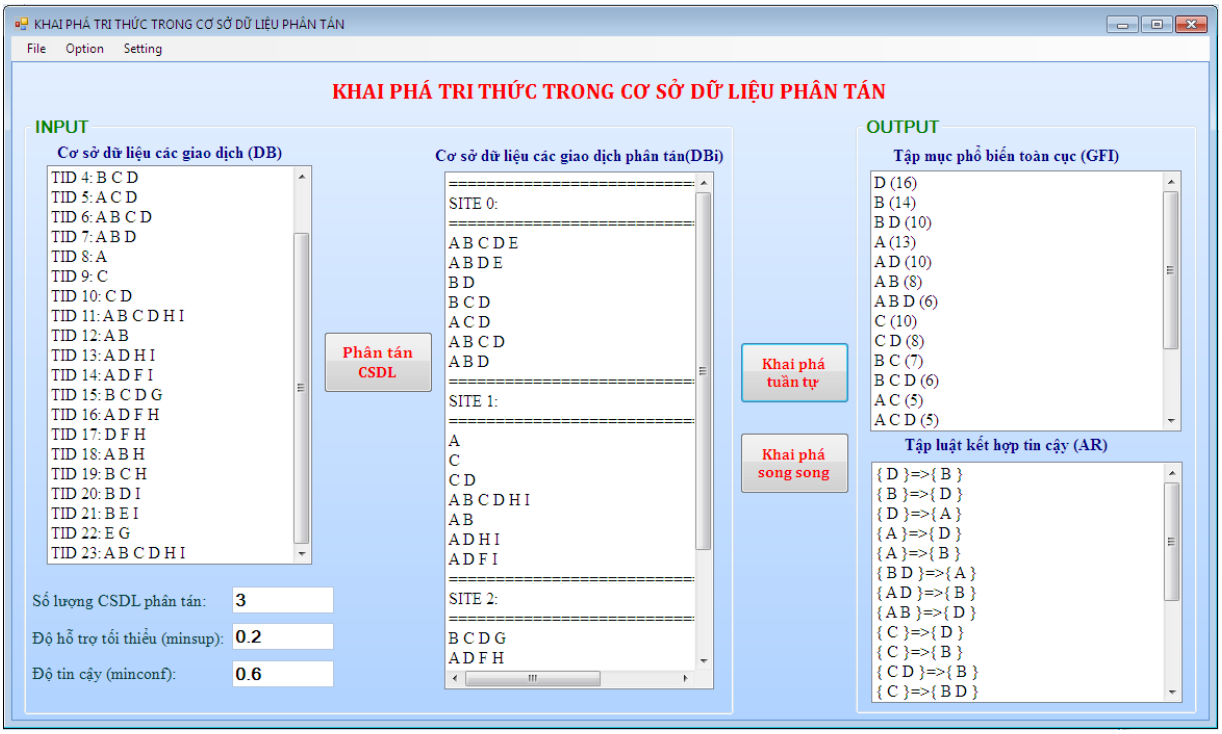
Sau khi mở file, ta nhập vào số lượng CSDL cần phân tán trong textbox *"Số lượng CSDL phân tán"* và click nút *"Phân tán CSDL"* (chức năng này nhằm mô phỏng cho việc khai phá dữ liệu trong môi trường phân tán).



Hình 3.4 Phân tán CSDL

3.4.3 Khai phá tuần tự

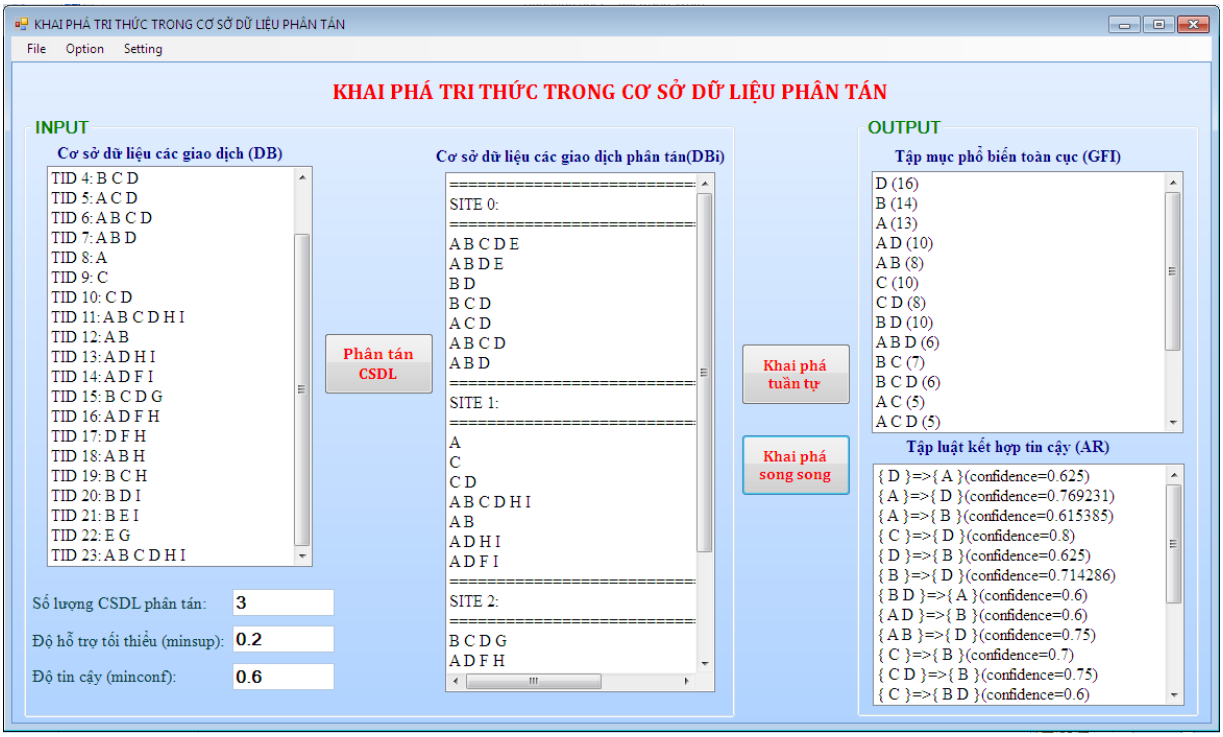
Sau khi nhập vào độ hỗ trợ tối thiểu (minsup) và độ tin cậy (minconf) ở các textbox tương ứng, click nút *"Khai phá tuần tự"*, chương trình sẽ tự động tính toán tập mục phổ biến và sinh các tập luật kết hợp tin cậy từ các tập mục phổ biến đã tìm được bằng thuật toán FPGrowth tuần tự.



Hình 3.5 Khai phá tuần tự

3.4.4 Khai phá song song

Sau khi nhập vào độ hỗ trợ tối thiểu (minsup) và độ tin cậy (minconf) ở các textbox tương ứng, click nút *"Khai phá song song"*, chương trình sẽ tự động tính toán tập mục phổ biến toàn cục (*GFI*) từ CSDL các giao tác ở các site đã phân tán và sinh các tập luật kết hợp tin cậy (có hiển thị độ tin cậy bên cạnh) từ các tập mục phổ biến đã tìm được bằng thuật toán FPGrowth song song (sử dụng OpenMP).



Hình 3.6 Khai phá song song

3.5 Thử nghiệm, đánh giá

Dưới đây là bảng tổng hợp một số kết quả chạy thử nghiệm chương trình với các bộ dữ liệu do học viên tự xây dựng:

STT	DB	minsup	minconf	Khai phá tuần tự	Khai phá song song
01	A B D E B C E A B D E A B C E A B C D E B C D	0.6	0.9	{E}=>{B} {A}=>{B} {A}=>{E} {AE}=>{B} {A}=>{BE} {AB}=>{E} {D}=>{B} {C}=>{B}	{E}=>{B} {A}=>{B} {A}=>{E} {AE}=>{B} {A}=>{BE} {AB}=>{E} {D}=>{B} {C}=>{B}

Qua quá trình thử nghiệm cho thấy chương trình chạy nhanh, cho kết quả chính xác. Việc khai phá tuần tự và song song (sử dụng FPGrowth) cho kết quả trùng khớp nhau. Do việc khai phá song song sử dụng OpenMP (chia sẻ bộ nhớ chung) nên sự chênh lệch về tốc độ tính toán của hai thuật toán trên chưa được thể hiện một cách rõ rệt.

Trong thời gian tới, tác giả sẽ cố gắng cài đặt thử nghiệm chương trình bằng việc sử dụng chuẩn lập trình song song MPI để thấy được sự khác nhau rõ rệt về tốc độ khai phá dữ liệu giữa hai thuật toán tuần tự và song song khi kích thước dữ liệu lớn.

KẾT LUẬN

1. Kết quả đạt được trong luận văn

Sau một thời gian tìm hiểu, nghiên cứu đến nay luận văn đã được hoàn thành. Về cơ bản luận văn đáp ứng được các nội dung đã đăng ký trong đề cương. Cụ thể luận văn đã đạt được một số kết quả chính sau:

- Tìm hiểu, nghiên cứu được một số thuật toán khai phá luật kết hợp theo hướng: tuần tự, song song.
- Đi sâu vào nghiên cứu một thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán. Phân tích, đánh giá các ưu nhược điểm của thuật toán từ đó đưa ra các đề xuất cải tiến nhằm tăng hiệu quả của thuật toán.

2. Hướng nghiên cứu tiếp theo

Với việc có nhiều nghiên cứu, đề xuất được thử nghiệm và ứng dụng thành công vào cuộc sống đã chứng tỏ KPDL nói chung, khai phá luật kết hợp nói riêng là một lĩnh vực nghiên cứu ổn định có nền tảng lý thuyết vững chắc thu hút được nhiều sự quan tâm.

Ngày nay với sự phát triển mạnh mẽ của công nghệ mạng, công nghệ tính toán đã dẫn tới việc phân tán nguồn dữ liệu như là một tất yếu. Và cùng với nó là sự phát triển của các thuật toán khai phá dữ liệu phân tán. Mặc dù đã có một số thuật toán khai phá luật kết hợp trong cơ sở dữ liệu phân tán được đề xuất, nhưng để đáp ứng được các nhu cầu ngày càng cao: thời gian xử lý nhanh, xử lý trên các CSDL phân tán lớn, … thì vẫn đòi hỏi phải có các thuật toán nhanh và mạnh hơn. Và chính vì vậy hướng nghiên cứu khai phá luật kết hợp trong môi trường phân tán vẫn là một hướng nghiên cứu thú vị và thực tế.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Đỗ Phúc (2005), *Giáo trình khai phá dữ liệu*, Nxb Đại học Quốc gia TP Hồ Chí Minh.

2. Đoàn Văn Ban, Nguyên Mậu Hân (2006), *Xử lý song song và phân tán*, Nxb Khoa học & Kỹ thuật, Hà Nội.

Tiếng Anh

3. Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth, (2002) *From Data Minning To Discory Knowledge in Database.*

4. Kwok-Leung Tsui, Victoria C,P. Chen, Wei Jiang, Y. Alp Aslandogan, (2001), *Data Minning Methods and applications*.

5. Two Crows (2005), *Introduction to Data Minning and Knowledge Discovery*, Edition third.

6. P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer and R. Wirth, *CRISP-DM 1.0 Process and User Guide*, http://www.crisp-dm.org , (2000).

7. I. H. Witten and E. Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, New York, (2000).

8. P. Berkhin: Survey of Clustering Data Mining Techniques. Research paper. Accrue Software, Inc, http://www.accrue.com , (2001).

9. R. O. Duda, P. E. Hart and D. G. Stork: Pattern Classification, Second Edition (2001), John Wiley & Sons, Inc, pp. 517-599.

10. R. Agrawal, T. Imielinski and A. Swami (1993), *Minning association rules between sets of items i large databases*, In ACM SIGMOD Intil. C@ Managenment of Data, May.

11. R. Agrawal and R. Srikant, (1994), *Fast algorithms for minning association rules,* In 20th VL.DBConf, Sept.

12. R Agrawal, H.Manila, R. Srikant, H. Toivonen and A. 1. Verkamo (1996), *Fast discovery of association rules,* In U.Fayyad and et al, editors, Advances in Knowledge Discovery and Data Minning. MIT Press.

13. J. Han, J. Pei, Y. Yin, and R. Mao. *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*. Data Mining and Knowledge Discovery, 2003.

14. David W. Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu, Yongjian Fu, *A Fast Distributed Algorithm for Mining Association Rules*, 1996 IEEE.

15. Ji-Fu Zhang, Hong Shi, Lian Zheng, *a method and algorithm of distributed mining associationrules in synchronisms,* Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002.

16. You-Lin Ruan, Gan Liu, Qing-Hua Li , *Parallel Algorithm For Mining Frequent Itemsets*, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

LÝ LỊCH TRÍCH NGANG

Họ và tên: Vũ Thị Quyên

Ngày, tháng, năm sinh: 08/03/1986 Nơi sinh: Ninh Bình

Địa chỉ liên lạc: Trung tâm Thư viện – Thiết bị Trường Đại Học Hoa Lư – Ninh Nhất – TP. Ninh Bình

QUÁ TRÌNH ĐÀO TẠO:

2005-2009: Đại học Vinh, Chính quy, Cử nhân, Tin học

2011 - 2013: Học viện Kỹ thuật quân sự, Chính quy, Thạc sĩ, Công nghệ thông tin

QUÁ TRÌNH CÔNG TÁC:

2009 - đến nay: Phòng Đào tạo – Quản lý khoa học. Trường Đại Học Hoa Lư

XÁC NHẬN QUYỂN LUẬN VĂN ĐỦ ĐIỀU KIỆN NỘP LƯU CHUYỂN

CHỦ NHIỆM BỘ MÔN **CÁN BỘ HƯỚNG DẪN**

QUẢN LÝ CHUYÊN NGÀNH

PGS.TS. Nguyễn Bá Tường