

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**GIANG THỊ THU HUYỀN**

**NGHIÊN CỨU CÁC LUẬT KẾT HỢP SONG SONG  
TRONG KHAI PHÁ DỮ LIỆU**

Ngành: Công nghệ thông tin  
Chuyên ngành: Hệ thống thông tin  
Mã số: 60 48 05

**LUẬN VĂN THẠC SĨ**

**NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS Đoàn Văn Ban**

Hà Nội – 2010

## LỜI CẢM ƠN

Để có được kết quả như ngày hôm nay, tôi luôn ghi nhớ công ơn của các thầy cô, bạn bè, đồng nghiệp và gia đình, những người đã dạy bảo và ủng hộ tôi trong suốt quá trình học tập.

Trước hết, tôi muốn gửi lời cảm ơn đến các thầy cô giáo trường Đại học Công Nghệ, Đại học Quốc Gia Hà Nội đã quan tâm tổ chức chỉ đạo và trực tiếp giảng dạy khoá cao học của chúng tôi. Đặc biệt, tôi xin gửi lời cảm ơn sâu sắc đến thầy giáo hướng dẫn PGS.TS Đoàn Văn Ban, người đã tận tình chỉ bảo và góp ý về mặt chuyên môn cho tôi trong suốt quá trình làm luận văn. Nếu không có sự giúp đỡ của thầy thì tôi khó có thể hoàn thành được luận văn này.

Cũng qua đây, tôi xin gửi lời cảm ơn đến ban lãnh đạo Khoa Hệ thống thông tin Kinh tế thuộc Học viện Ngân hàng, nơi tôi đang công tác, đã tạo mọi điều kiện thuận lợi cho tôi trong thời gian hoàn thành các môn học cũng như trong suốt quá trình làm luận văn tốt nghiệp.

Cuối cùng, tôi xin cảm ơn bố mẹ, chồng và các bạn bè, đồng nghiệp đã luôn ủng hộ, động viên để tôi yên tâm nghiên cứu và hoàn thành luận văn.

Trong suốt quá trình làm luận văn, bản thân tôi đã cố gắng tập trung tìm hiểu, nghiên cứu và tham khảo thêm nhiều tài liệu liên quan. Tuy nhiên, do bản thân mới bắt đầu trên con đường nghiên cứu khoa học, chắc chắn bản luận văn vẫn còn nhiều thiếu sót. Tôi rất mong được nhận sự chỉ bảo của các Thầy Cô giáo và các góp ý của bạn bè, đồng nghiệp để luận văn được hoàn thiện hơn.

Hà Nội, tháng 04 năm 2010

Giang Thị Thu Huyền

## **LỜI CAM ĐOAN**

Tôi xin cam đoan đề tài “**Nghiên cứu các luật kết hợp song song trong khai phá dữ liệu**” là kết quả của tự bản thân tôi tìm hiểu, nghiên cứu. Các tài liệu tham khảo được trích dẫn và chú thích đầy đủ. Tôi xin chịu trách nhiệm về luận văn của mình.

## MỤC LỤC

MỞ ĐẦU .....	1
CHƯƠNG 1 TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU .....	3
1. 1. Khai phá dữ liệu .....	3
1. 1. 1. Khái niệm Khai phá dữ liệu .....	3
1. 1. 2. Kiến trúc của một hệ thống khai phá dữ liệu .....	5
1. 1. 3. Một số kỹ thuật khai phá dữ liệu .....	6
1. 1. 4. Lựa chọn phương pháp khai phá dữ liệu.....	8
1. 2. Ứng dụng của khai phá dữ liệu .....	9
1. 3. Một số khó khăn trong khai phá dữ liệu.....	10
1. 4. Kết luận chương 1 .....	11
CHƯƠNG 2 KHAI PHÁ CÁC LUẬT KẾT HỢP SONG SONG .....	12
2. 1. Luật kết hợp trong khai phá dữ liệu.....	12
2. 1. 1. Một số hướng tiếp cận trong khai phá luật kết hợp .....	12
2. 1. 2. Các tính chất của luật kết hợp .....	13
2. 1. 3. Bài toán khai phá luật kết hợp .....	17
2. 1. 4. Một số thuật toán khai phá luật kết hợp.....	17
2. 2. Các thuật toán song song phát hiện luật kết hợp .....	26
2. 2. 1. Thuật toán song song .....	27
2. 2. 2. Khai phá các luật kết hợp song song .....	30
2. 3. Kết luận chương 2 .....	49
CHƯƠNG 3 CÀI ĐẶT THUẬT TOÁN KHAI PHÁ CÁC LUẬT KẾT HỢP SONG SONG TRONG KHAI PHÁ DỮ LIỆU.....	50
3. 1. Cài đặt thuật toán khai phá các luật kết hợp song song .....	50
3. 1. 1. Môi trường cài đặt chương trình thử nghiệm .....	50
3. 1. 2. Mô tả dữ liệu của bài toán .....	51
3. 1. 3. Giao diện chương trình .....	52
3. 2. Đánh giá kết quả.....	58
3. 2. 1. Phương pháp đánh giá các chương trình song song .....	58
3. 2. 2. Kết quả cài đặt chương trình thử nghiệm.....	59
KẾT LUẬN .....	60
TÀI LIỆU THAM KHẢO.....	62
PHỤ LỤC .....	64

## DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

Tên viết tắt	Diễn giải
$C_k$	Tập các k-itemset ứng viên (Candidate sets)
Conf	Độ tin cậy (Confidence)
D	Cơ sở dữ liệu giao dịch
$D_i$	Phần thứ i của cơ sở dữ liệu D
Item	Mục
Itemset	Tập mục
k-itemset	Tập mục gồm k mục
$L_k$	Tập các k-itemset phổ biến
MPI	Truyền thông điệp (Message Passing Interface)
minconf	Nguồn tin cậy tối thiểu (minimum confidence)
minsup	Nguồn hỗ trợ tối thiểu (minimum support)
SC	Số đếm hỗ trợ (Support count)
Sup	Độ hỗ trợ (Support)
T	Giao dịch (Transaction)
TID	Định danh của giao dịch (Unique Transaction Identifier)
Tid-List	Danh sách các định danh của giao dịch
$X \Rightarrow Y$	Luật kết hợp (Với X là tiền đề, Y là hệ quả)

## **DANH MỤC CÁC BẢNG**

<b>Bảng</b>	<b>Trang</b>
Bảng 2. 1. Một số ký hiệu dùng trong thuật toán Apriori .....	18
Bảng 2. 2. Ký hiệu dùng trong các thuật toán song song .....	31

## DANH MỤC CÁC HÌNH VẼ

<b>Hình</b>	<b>Trang</b>
Hình 1. 1. Quá trình khai phá dữ liệu .....	4
Hình 1. 2. Kiến trúc của một hệ thống khai phá dữ liệu .....	6
Hình 1. 3. Mô tả luật kết hợp.....	8
Hình 2. 1. Tập chứa tập mục không phổ biến là không phổ biến .....	15
Hình 2. 2. Minh họa thuật toán Apriori tìm tập mục phổ biến .....	22
Hình 2. 3. Sinh luật từ tập mục phổ biến .....	25
Hình 2. 4. Tính toán tuần tự .....	27
Hình 2. 5. Tính toán song song.....	27
Hình 2. 6. Kiến trúc bộ nhớ chia sẻ .....	29
Hình 2. 7. Kiến trúc bộ nhớ phân tán.....	29
Hình 2. 8. Kiến trúc bộ nhớ lai .....	30
Hình 2. 9. Giải thuật Count Distribution.....	32
Hình 2. 10. Cơ sở dữ liệu D và các tập mục phổ biến .....	33
Hình 2. 11. Tìm tập mục phổ biến theo thuật toán song song Count Distribution .....	33
Hình 2. 12. Tìm tập mục phổ biến theo thuật toán song song Data Distribution.....	36
Hình 2. 13. Tổ chức dữ liệu theo chiều ngang và theo chiều dọc .....	37
Hình 2. 14. Chuyển đổi dữ liệu .....	40
Hình 2. 15. Thuật toán song song Eclat .....	41
Hình 2. 16. Khai phá tập mục phổ biến sử dụng thuật toán song song Eclat .....	42
Hình 2. 17. Cấu trúc FP-tree cục bộ được xây dựng từ các phân hoạch cơ sở dữ liệu ..	46
Hình 2. 18. Khai phá tập mục phổ biến sử dụng thuật toán song song FP-Growth.....	46
Hình 3. 1. Giao diện nhập dữ liệu đầu vào.....	56
Hình 3. 2. Giao diện thực hiện theo thuật toán Apriori .....	56
Hình 3. 3. Giao diện thực hiện theo thuật toán song song Count Distribution .....	57
Hình 3. 4. Giao diện thực hiện theo thuật toán song song Eclat .....	57

## MỞ ĐẦU

### 1. Đặt vấn đề

Ngày nay, con người đang sở hữu kho dữ liệu phong phú, đa dạng và khổng lồ. Đặc biệt sự phát triển của công nghệ thông tin và việc ứng dụng công nghệ thông tin trong nhiều lĩnh vực đã làm cho kho dữ liệu ấy tăng lên nhanh chóng. Sự bùng nổ này đã dẫn tới một yêu cầu cấp thiết là cần có những kỹ thuật và công cụ mới để tự động chuyển đổi lượng dữ liệu khổng lồ kia thành các tri thức có ích. Mặt khác, trong môi trường cạnh tranh thì người ta ngày càng cần có thông tin với tốc độ nhanh để giúp cho việc ra quyết định và ngày càng có nhiều câu hỏi mang tính chất định tính cần phải trả lời dựa trên khối lượng dữ liệu khổng lồ đã có. Tiến hành các công việc như vậy chính là quá trình phát hiện tri thức trong cơ sở dữ liệu, trong đó kỹ thuật khai phá dữ liệu cho phép phát hiện tri thức tiềm ẩn ấy. Từ đó, các kỹ thuật khai phá dữ liệu đã trở thành một lĩnh vực thời sự của nền Công nghệ thông tin thế giới hiện nay nói chung và Việt Nam nói riêng. Rất nhiều tổ chức và công ty lớn trên thế giới đã áp dụng kỹ thuật khai phá dữ liệu vào các hoạt động sản xuất kinh doanh của mình và thu được những lợi ích to lớn.

Các kỹ thuật phát hiện tri thức và khai phá dữ liệu được thực hiện qua nhiều giai đoạn và sử dụng nhiều kỹ thuật: phân lớp (classification), phân cụm (clustering), phân tích sự tương tự (similarity analysis), tổng hợp (summarization), luật kết hợp (association rules), ... Một trong những nội dung cơ bản và phổ biến trong khai phá dữ liệu là phát hiện các luật kết hợp. Phương pháp này nhằm tìm ra các tập thuộc tính thường xuất hiện đồng thời trong cơ sở dữ liệu và rút ra các luật về ảnh hưởng của một tập thuộc tính dẫn đến sự xuất hiện của một hoặc nhiều tập thuộc tính khác như thế nào? Do đó việc phát hiện ra các luật kết hợp là một bước rất quan trọng trong khai phá dữ liệu.

Mặt khác, hiện nay nhu cầu song song hóa và xử lý phân tán là rất cần thiết bởi kích thước dữ liệu lưu trữ ngày càng lớn nên đòi hỏi tốc độ xử lý cũng như dung lượng bộ nhớ hệ thống phải đảm bảo. Vì vậy, yêu cầu cần có những thuật toán song song hiệu quả cho việc phát hiện các luật kết hợp trong khai phá dữ liệu là rất cần thiết, góp phần thúc đẩy khả năng ứng dụng của việc phát hiện tri thức, hỗ trợ ra quyết định vào trong hoạt động thực tiễn.

Từ những vấn đề nêu trên, tôi chọn đề tài “*Nghiên cứu các luật kết hợp song song trong khai phá dữ liệu*” để làm luận văn tốt nghiệp.

### 2. Mục tiêu của luận văn

- ❖ Tìm hiểu khái quát về khai phá dữ liệu trong đó đi sâu về các luật kết hợp.
- ❖ Tìm hiểu một số mô hình tính toán song song.

- ❖ Nghiên cứu xây dựng các thuật toán luật kết hợp song song trong khai phá dữ liệu.
- ❖ Cài đặt một số thuật toán song song khai phá dữ liệu và phát hiện luật kết hợp.

### 3. **Bố cục của luận văn**

Luận văn chia làm 3 chương:

Chương 1: Tổng quan về khai phá dữ liệu

Chương này giới thiệu quá trình khai phá dữ liệu và phát hiện tri thức, phương pháp khai phá dữ liệu, ứng dụng và một số khó khăn trong khai phá dữ liệu.

Chương 2: Khai phá các luật kết hợp song song

Chương này trình bày tóm tắt luật kết hợp, mô hình của bài toán khai phá luật kết hợp, các khái niệm cơ bản luật kết hợp, các phương pháp khai phá các luật kết hợp và khai phá các luật kết hợp song song.

Chương 3: Cài đặt thuật toán khai phá các luật kết hợp song song ứng dụng cho bài toán khai phá dữ liệu.

## CHƯƠNG 1

### TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

#### **1. 1. Khai phá dữ liệu**

##### **1. 1. 1. Khái niệm Khai phá dữ liệu**

Khai phá dữ liệu (Data Mining) là một khái niệm ra đời vào những năm cuối của thập kỷ 1980. Nó là quá trình khám phá thông tin ẩn được tìm thấy trong các cơ sở dữ liệu và có thể xem như là một bước trong quá trình khám phá tri thức. Data Mining là giai đoạn quan trọng nhất trong tiến trình khai phá tri thức từ cơ sở dữ liệu, các tri thức này hỗ trợ trong việc ra quyết định trong khoa học và kinh doanh, ...

Giáo sư Tom Mitchell [20] đã đưa ra định nghĩa của Khai phá dữ liệu như sau: “Khai phá dữ liệu là việc sử dụng dữ liệu lịch sử để khám phá những qui tắc và cải thiện những quyết định trong tương lai.” Với một cách tiếp cận ứng dụng hơn, Tiến sĩ Fayyad [21] đã phát biểu: “Khai phá dữ liệu, thường được xem là việc khám phá tri thức trong các cơ sở dữ liệu, là một quá trình trích xuất những thông tin ẩn, trước đây chưa biết và có khả năng hữu ích, dưới dạng các qui luật, ràng buộc, qui tắc trong cơ sở dữ liệu.” hay nói cách khác “Khai phá dữ liệu – Data Mining là tiến trình khám phá tri thức tiềm ẩn trong các cơ sở dữ liệu. Cụ thể hơn, đó là tiến trình trích lọc, sản sinh những tri thức hoặc các mẫu tiềm ẩn, chưa biết nhưng hữu ích từ cơ sở dữ liệu lớn” [2].

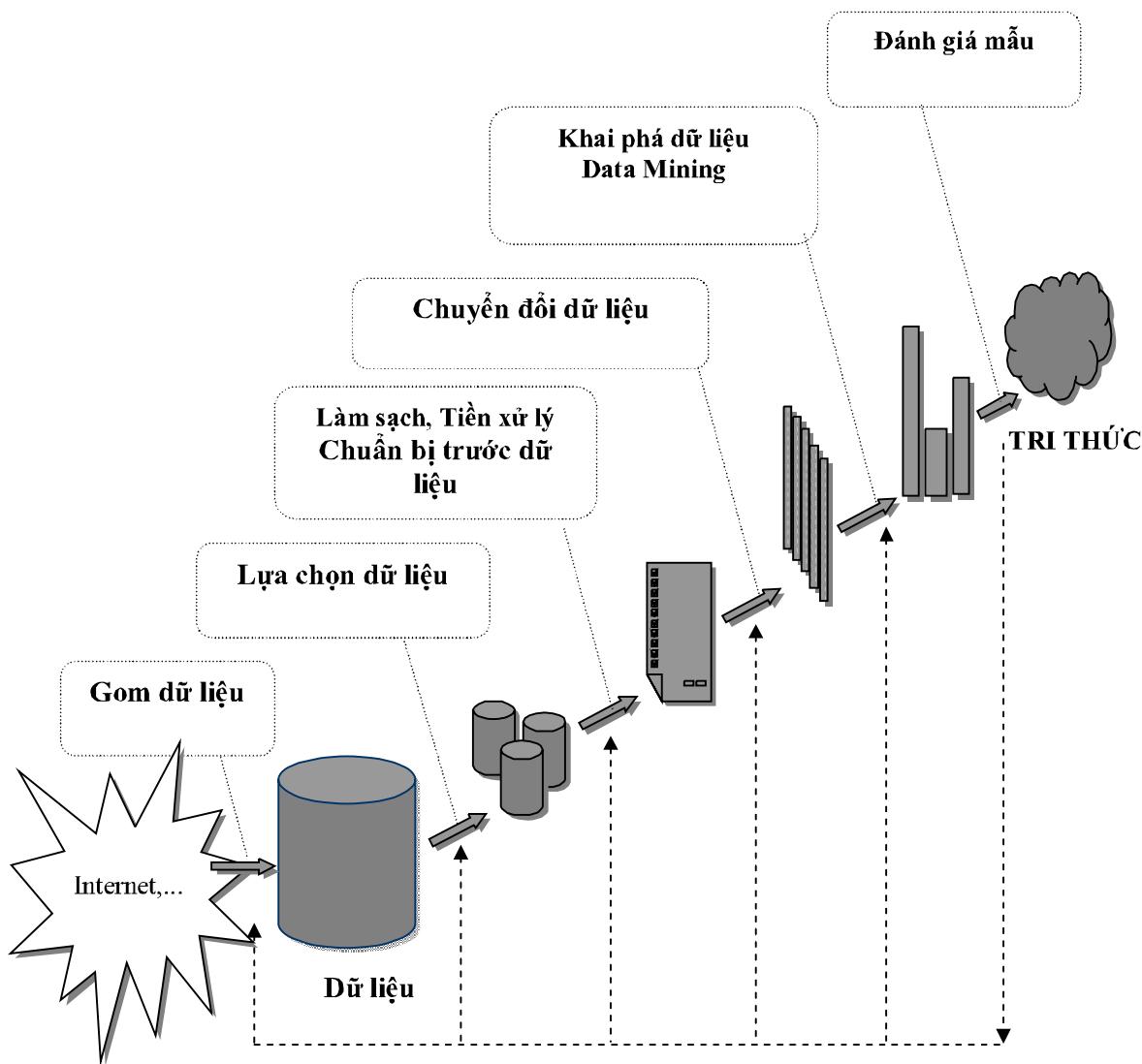
Nói tóm lại, Khai phá dữ liệu là một quá trình học tri thức mới từ những dữ liệu đã thu thập được [8]–[12]–[15].

Khai phá dữ liệu là tiến trình khai quát các sự kiện rời rạc trong dữ liệu thành các tri thức mang tính khai quát, tính quy luật hỗ trợ tích cực cho các tiến trình ra quyết định. Khai phá dữ liệu là việc trích rút tri thức một cách tự động và hiệu quả từ một khối dữ liệu rất lớn. Tri thức đó thường ở dạng các mẫu tin có tính chất không tầm thường, không tưởng minh (ẩn), chưa được biết đến và có tiềm năng mang lại lợi ích.

Để hình dung vấn đề này ta có thể sử dụng một ví dụ đơn giản như sau: Khai phá dữ liệu được ví như tìm một cây kim trong đống cỏ khô. Trong ví dụ này, cây kim là một mảnh nhỏ tri thức hoặc một thông tin có giá trị và đống cỏ khô là một kho cơ sở dữ liệu rộng lớn. Như vậy, những thông tin có giá trị tiềm ẩn trong kho cơ sở dữ liệu sẽ được chiết xuất ra và sử dụng một cách hữu ích nhờ khai phá dữ liệu.

Chức năng khai phá dữ liệu gồm có gộp nhóm phân loại, dự báo, dự đoán và phân tích các liên kết. Năm 1989, Fayyad, Smyth và Piatek-Sapiro đã dùng khái niệm *Phát hiện tri thức từ cơ sở dữ liệu* (Knowledge Discovery in Database-KDD). Trong đó, khai phá dữ liệu là một giai đoạn rất đặc biệt trong toàn bộ quá trình, nó sử dụng các kỹ thuật để tìm ra các mẫu từ dữ liệu. Có thể coi khai phá dữ liệu là cốt lõi của quá trình phát hiện tri thức.

Quá trình khai phá dữ liệu sẽ tiến hành qua 6 giai đoạn như hình 1. 1 [7]



Hình 1.1. Quá trình khai phá dữ liệu

Bắt đầu của quá trình là kho dữ liệu thô và kết thúc với tri thức được chiết xuất ra. Về lý thuyết thì có vẻ rất đơn giản nhưng thực sự đây là một quá trình rất khó khăn gấp phải rất nhiều vướng mắc như: quản lý các tập dữ liệu, phải lặp đi lặp lại toàn bộ quá trình, ...

- 1. Gom dữ liệu (Gathering):** Tập hợp dữ liệu là bước đầu tiên trong quá trình khai phá dữ liệu. Đây là bước được khai thác trong một cơ sở dữ liệu, một kho dữ liệu và thậm chí các dữ liệu từ các nguồn ứng dụng Web.
- 2. Trích lọc dữ liệu (Selection):** Ở giai đoạn này dữ liệu được lựa chọn hoặc phân chia theo một số tiêu chuẩn nào đó, ví dụ chọn tất cả những người có tuổi đời từ 25 – 35 và có trình độ đại học.
- 3. Làm sạch, tiền xử lý và chuẩn bị trước dữ liệu (Cleaning, Pre-processing and Preparation):** Giai đoạn thứ ba này là giai đoạn hay bị sao lãng, nhưng thực tế nó là một bước rất quan trọng trong quá trình khai phá dữ liệu. Một số

lỗi thường mắc phải trong khi gom dữ liệu là tính không đủ chặt chẽ, logíc. Vì vậy, dữ liệu thường chứa các giá trị vô nghĩa và không có khả năng kết nối dữ liệu. Ví dụ: tuổi = 273. Giai đoạn này sẽ tiến hành xử lý những dạng dữ liệu không chặt chẽ nói trên. Những dữ liệu dạng này được xem như thông tin dư thừa, không có giá trị. Bởi vậy, đây là một quá trình rất quan trọng vì dữ liệu này nếu không được “làm sạch - tiền xử lý - chuẩn bị trước” thì sẽ gây nên những kết quả sai lệch nghiêm trọng.

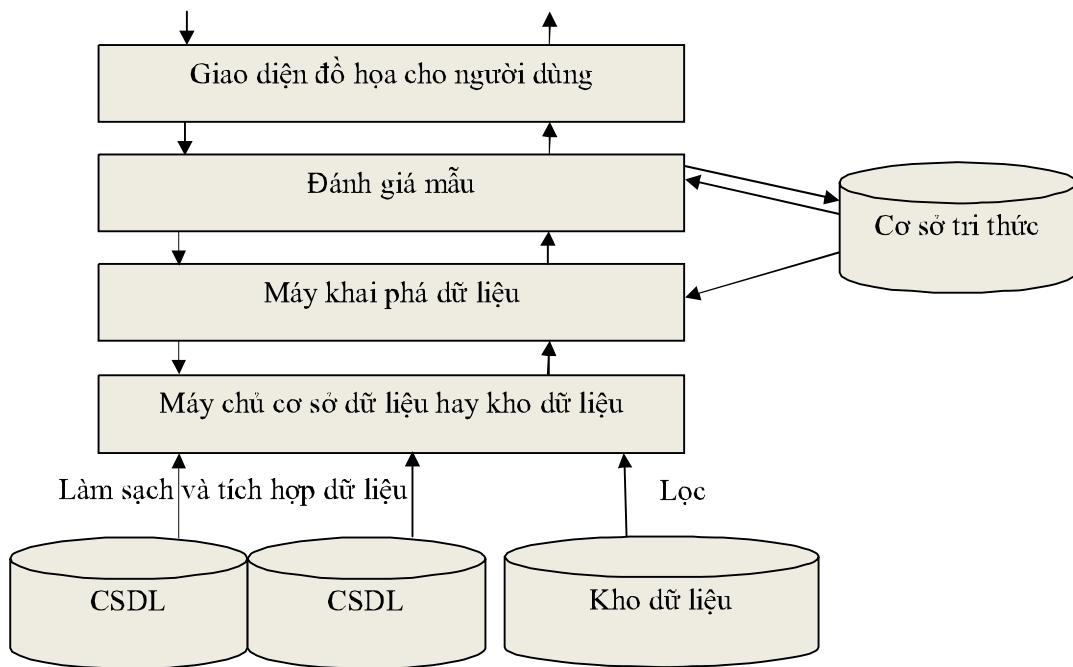
4. **Chuyển đổi dữ liệu (Transformation):** Tiếp theo là giai đoạn chuyển đổi dữ liệu, dữ liệu đưa ra có thể sử dụng và điều khiển được bởi việc tổ chức lại nó. Dữ liệu đã được chuyển đổi phù hợp với mục đích khai thác.
5. **Phát hiện và trích mẫu dữ liệu (Pattern Extraction and Discovery):** Đây là bước mang tính tư duy trong khai phá dữ liệu. Ở giai đoạn này nhiều thuật toán khác nhau đã được sử dụng để trích ra các mẫu từ dữ liệu. Thuật toán thường dùng là nguyên tắc phân loại, nguyên tắc kết hợp hoặc các mô hình dữ liệu tuần tự, ...
6. **Đánh giá kết quả mẫu (Evaluation of Result):** Đây là giai đoạn cuối trong quá trình khai phá dữ liệu. Ở giai đoạn này, các mẫu dữ liệu được chiết xuất ra bởi phần mềm khai phá dữ liệu. Không phải bất cứ mẫu dữ liệu nào cũng đều hữu ích, đôi khi nó còn bị sai lệch. Vì vậy, cần phải ưu tiên những tiêu chuẩn đánh giá để chiết xuất ra các tri thức (Knowledge).

Trên đây là 6 giai đoạn trong quá trình khai phá dữ liệu, trong đó giai đoạn 5 là giai đoạn được quan tâm nhiều nhất, đó là khai phá dữ liệu.

### **1. 1. 2. Kiến trúc của một hệ thống khai phá dữ liệu**

- ❖ Máy chủ cơ sở dữ liệu hay máy chủ kho dữ liệu (Database or warehouse server): Máy chủ này có trách nhiệm lấy dữ liệu thích hợp dựa trên những yêu cầu khai phá của người dùng.
- ❖ Cơ sở tri thức (Knowledge base): Đây là miền tri thức được dùng để tìm kiếm hay đánh giá độ quan trọng của các hình mẫu kết quả.
- ❖ Máy khai phá dữ liệu (Data mining engine): Một hệ thống khai phá dữ liệu cần phải có một tập các modun chức năng để thực hiện công việc, chẳng hạn như đặc trưng hóa, kết hợp, phân lớp, phân cụm, phân tích sự tiến hoá...
- ❖ Modun đánh giá mẫu (Pattern evaluation): Bộ phận này tương tác với các modun khai phá dữ liệu để tập trung vào việc duyệt tìm các mẫu đáng được quan tâm. Cũng có thể modun đánh giá mẫu được tích hợp vào modun khai phá tùy theo sự cài đặt của phương pháp khai phá được dùng.
- ❖ Giao diện đồ họa cho người dùng (Graphical user interface): Thông qua giao diện này, người dùng tương tác với hệ thống bằng cách đặc tả một yêu cầu

khai phá hay một nhiệm vụ, cung cấp thông tin trợ giúp cho việc tìm kiếm và thực hiện khai phá thăm dò trên các kết quả khai phá trung gian.



*Hình 1.2. Kiến trúc của một hệ thống khai phá dữ liệu*

### 1. 1. 3. Một số kỹ thuật khai phá dữ liệu

Các kỹ thuật khai phá dữ liệu thường được chia thành 2 nhóm chính [12]:

- ❖ Kỹ thuật khai phá dữ liệu mô tả: có nhiệm vụ mô tả về các tính chất hoặc các đặc tính chung của dữ liệu trong CSDL hiện có. Các kỹ thuật này gồm có: phân cụm (*clustering*), tóm tắt (*summarization*), trực quan hóa (*visualization*), phân tích sự phát triển và độ lệch (*Evolution and deviation analysis*), phát hiện luật kết hợp (*association rules*), ...
- ❖ Kỹ thuật khai phá dữ liệu dự đoán: có nhiệm vụ đưa ra các dự đoán dựa vào các suy diễn trên dữ liệu hiện thời. Các kỹ thuật này gồm có: phân lớp (*classification*), hồi quy (*regression*), ...

Tuy nhiên, do khuôn khổ có hạn nên tôi chỉ giới thiệu 3 phương pháp thông dụng nhất là: phân lớp dữ liệu, phân cụm dữ liệu và khai phá luật kết hợp.

#### 1. 1. 3. 1. Phân lớp

Phân lớp dữ liệu (Classification) là chia các đối tượng dữ liệu thành các lớp dựa trên các đặc trưng của tập dữ liệu. 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. Phương pháp này rất có ích trong giai đoạn đầu của quá trình nghiên cứu khi ta biết rất ít về đối tượng cần nghiên cứu, nó là tiền đề để tiến hành các phương

pháp phát hiện tri thức. Có nhiều phương pháp phân lớp: phân lớp dựa trên cây quyết định, phân lớp Bayesian, ... Quá trình phân lớp dữ liệu thường gồm hai bước [12]:

- ❖ *Bước 1:* Xây dựng mô hình dựa trên việc phân tích các mẫu dữ liệu có sẵn. Mỗi mẫu tương ứng với một lớp, được quyết định bởi một thuộc tính gọi là thuộc tính phân lớp. Các mẫu dữ liệu này còn được gọi là tập dữ liệu huấn luyện (training dataset). Nhãn lớp của tập dữ liệu huấn luyện phải được xác định trước khi xây dựng mô hình, vì vậy phương pháp này còn được gọi là học có giám sát (supervised learning).
- ❖ *Bước 2:* Sử dụng mô hình để phân lớp dữ liệu. Chúng ta phải tính độ chính xác của mô hình, nếu độ chính xác là chấp nhận được thì mô hình sẽ được sử dụng để dự đoán lớp cho các mẫu dữ liệu khác trong tương lai.

### **1. 1. 3. 2. Phân cụm**

Phân cụm (Clustering) là việc nhóm các đối tượng dữ liệu thành các lớp đối tượng có sự tương tự nhau dựa trên các thuộc tính của chúng. Mỗi lớp đối tượng được gọi là một cụm (cluster). Một cụm bao gồm các đối tượng mà giữa bản thân chúng có sự ràng buộc lẫn nhau và khác biệt so với các lớp đối tượng khác. Phân cụm dữ liệu là một ví dụ của phương pháp học không có giám sát (unsupervised learning). Phân cụm dữ liệu không đòi hỏi phải định nghĩa trước các mẫu dữ liệu huấn luyện. Vì thế, có thể coi phân cụm dữ liệu là một cách học bằng quan sát (learning by observation), trong khi phân lớp dữ liệu là học qua ví dụ (learning by example). Trong phương pháp này ta không thể biết kết quả các cụm thu được sẽ như thế nào khi bắt đầu quá trình. Các cụm có thể tách riêng hay phân cấp hoặc gối lên nhau, có nghĩa là một mục dữ liệu có thể vừa thuộc cụm này vừa thuộc cụm kia. Vì vậy, thông thường cần có một chuyên gia về lĩnh vực đó để đánh giá các cụm thu được.

Phân cụm dữ liệu được sử dụng nhiều trong các ứng dụng về phân loại thị trường, phân loại khách hàng, nhận dạng mẫu, phân loại trang Web, ... Ngoài ra, phân cụm còn được sử dụng như một bước tiền xử lý cho các thuật toán khai phá dữ liệu khác.

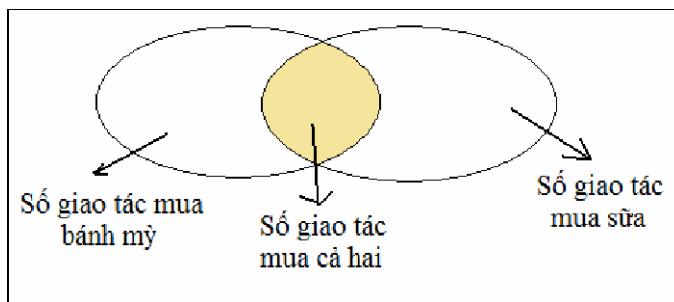
### **1. 1. 3. 3. Luật kết hợp**

Phương pháp phát hiện các luật kết hợp (Association Rules) nhằm phát hiện ra các luật kết hợp giữa các thành phần dữ liệu trong cơ sở dữ liệu [4]. Các giải thuật Tìm luật liên kết 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. Đầu ra của thuật toán là tập luật kết hợp tìm được. Cho trước một tập các giao tác, trong đó mỗi giao tác là một tập các mục, tìm sự tương quan giữa các mục như là một luật và kết quả của giải thuật khai phá dữ liệu là tập luật kết hợp tìm được. Luật kết hợp thường có dạng  $X \Rightarrow Y$ . Trong đó:  $X$  là tiền đề,  $Y$  là hệ quả ( $X, Y$  là hai tập của mục). Ý nghĩa trực quan của

luật là các giao tác của cơ sở dữ liệu mà trong đó nội dung X có khuynh hướng đến nội dung Y.

Có hai thông số quan trọng của luật kết hợp là độ hỗ trợ (support) và độ tin cậy (confidence). Độ hỗ trợ và độ tin cậy là hai độ đo của sự đáng quan tâm của luật. Chúng tương ứng phản ánh sự hữu ích và sự chắc chắn của luật đã khám phá. Khai phá các luật kết hợp từ cơ sở dữ liệu là việc tìm các luật có độ hỗ trợ và độ tin cậy lớn hơn ngưỡng mà người dùng xác định trước.

Ví dụ: Phân tích giỏ hàng của người mua hàng trong một siêu thị ta thu được luật: “68% khách hàng mua sữa thì cũng mua bánh mỳ, 21% mua cả hai thứ. Trong ví dụ trên thì 68% là độ tin cậy của luật (số phần trăm giao dịch thỏa mãn về trái thì thỏa mãn về phải), 21% là độ hỗ trợ (số phần trăm giao dịch thỏa mãn cả hai về trái và phải).



*Hình 1.3. Mô tả luật kết hợp*

Luật kết hợp mang lại những thông tin vô cùng quan trọng, nó hỗ trợ không nhỏ trong quá trình ra quyết định. Phương pháp này được sử dụng rất nhiều trong các lĩnh vực như marketing có chủ đích, phân tích thị trường, quản lý kinh doanh, ... Khai phá luật kết hợp được thực hiện qua hai bước:

- ❖ *Bước 1:* Tìm tất cả các tập mục phổ biến, một tập mục phổ biến được xác định thông qua việc tính độ hỗ trợ và thỏa mãn độ hỗ trợ cực tiểu.
- ❖ *Bước 2:* Sinh ra các luật kết hợp mạnh từ tập mục phổ biến, các luật này phải thỏa mãn độ hỗ trợ cực tiểu và độ tin cậy cực tiểu.

Phương pháp này được sử dụng rất hiệu quả trong các lĩnh vực như marketing có chủ đích, phân tích quyết định, quản lý kinh doanh, phân tích thị trường, ...

#### **1. 1. 4. Lựa chọn phương pháp khai phá dữ liệu**

Cấu trúc của thuật toán khai phá dữ liệu có ba thành phần chính sau: Biểu diễn mô hình, đánh giá mô hình và phương pháp tìm kiếm.

- ❖ *Biểu diễn mô hình:* Mô hình được biểu diễn bằng ngôn ngữ L nào đó để mô tả các mẫu có thể khai phá được. Nếu việc biểu diễn mô hình hạn chế thì không có thời gian học tập hoặc không có các mẫu để tạo ra mô hình chính xác cho dữ liệu. Người phân tích dữ liệu cần phải hiểu đầy đủ các giả thiết mô tả,

người thiết kế thuật toán phải diễn tả được giả thiết mô tả nào được tạo ra bởi thuật toán nào một cách rõ ràng.

- ❖ *Dánh giá mô hình:* Đánh giá xem mẫu có đáp ứng được các tiêu chuẩn của quá trình phát hiện tri thức hay không. Đánh giá độ chính xác dự đoán dựa trên đánh giá chéo.
- ❖ *Phương pháp tìm kiếm:*

- **Tìm kiếm tham số:** Các thuật toán tìm kiếm các tham số để tối ưu hóa các tiêu chuẩn đánh giá mô hình với dữ liệu quan sát được và với một biểu diễn mô hình đã định.
- **Tìm kiếm mô hình:** Giống như một vòng lặp qua phương pháp tìm kiếm tham số, biểu diễn mô hình bị thay đổi tạo nên họ các mô hình. Với một biểu diễn mô hình, phương pháp tìm kiếm tham số được áp dụng để đánh giá chất lượng mô hình.

Hiện nay, người ta chưa đưa ra được một tiêu chuẩn nào trong việc quyết định sử dụng phương pháp nào vào trong trường hợp nào thì có hiệu quả, có nhiều kỹ thuật và mỗi kỹ thuật được sử dụng cho nhiều bài toán khác nhau. Các thuật toán khai phá dữ liệu tự động chỉ đang ở giai đoạn phát triển ban đầu, các kỹ thuật khai phá dữ liệu còn mới với lĩnh vực kinh doanh. Rõ ràng là để trả lời câu hỏi “khai phá dữ liệu dùng kỹ thuật nào là tốt?” thật không đơn giản vì mỗi phương pháp thì có điểm mạnh và điểm yếu riêng, thậm chí chúng ta còn phải kết hợp các phương pháp trong quá trình khai phá.

## 1. 2. Ứng dụng của khai phá dữ liệu

Khai phá dữ liệu được vận dụng trong nhiều lĩnh vực khác nhau nhằm khai thác nguồn dữ liệu phong phú được lưu trữ trong các hệ thống thông tin. Tùy theo bản chất của từng lĩnh vực, việc vận dụng khai phá dữ liệu có những cách tiếp cận khác nhau.

**Ngân hàng:** Xây dựng mô hình dự báo rủi ro tín dụng. Tìm kiếm tri thức, quy luật của thị trường chứng khoán và đầu tư bất động sản.

**Thương mại điện tử:** Tìm hiểu, định hướng thúc đẩy, giao tiếp với khách hàng. Phân tích hành vi mua sắm trên mạng và cho biết thông tin tiếp thị phù hợp với nhiều loại khách hàng.

**Marketing:** Phân tích nhu cầu khách hàng dựa trên mẫu dữ liệu mua bán hàng từ đó xác định chiến lược kinh doanh, quảng cáo, kế hoạch sản xuất, ...

Khai phá dữ liệu cũng được vận dụng hiệu quả để giải quyết các bài toán phức tạp trong các ngành đòi hỏi kỹ thuật cao [11], như tìm kiếm mỏ dầu từ ảnh viễn thám, cảnh báo hỏng hóc trong các hệ thống sản xuất, ... Các kỹ thuật Khai phá dữ liệu đã được áp dụng thành công trong việc dự đoán tải sử dụng điện năng cho các công ty cung cấp điện, lưu lượng viễn thông cho các công ty điện thoại, mức độ tiêu thụ sản

phẩm cho các nhà sản xuất, giá trị của sản phẩm trên thị trường cho các công ty tài chính, ...

Ngoài ra, Khai phá dữ liệu còn được áp dụng cho các vấn đề xã hội như phân tích các kết quả phòng chống và điều trị một số loại bệnh, phân tích tác hại của ma tuý, phát hiện tội phạm hay tăng cường an ninh xã hội, ... Việc vận dụng thành công đã mang lại những hiệu quả thiết thực cho các hoạt động diễn ra hàng ngày trong đời sống.

### **1. 3. Một số khó khăn trong khai phá dữ liệu**

- *Cơ sở dữ liệu lớn*: Các tập dữ liệu cần xử lý trong khai phá dữ liệu thường có kích thước cực kỳ lớn về cả số lượng các bản ghi và số lượng các thuộc tính. Trong thực tế, kích thước của các tập dữ liệu trong khai phá dữ liệu thường ở mức tera-byte (hàng ngàn giga-byte). Với kích thước như thế, thời gian xử lý thường cực kỳ dài. Mặc dù kích thước bộ nhớ trong của máy tính đã gia tăng đáng kể trong thời gian gần đây, việc gia tăng này cũng không thể đáp ứng kịp với việc tăng kích thước dữ liệu. Vì vậy, việc vận dụng các kỹ thuật xác suất, lấy mẫu, đệm, song song, ... vào các giải thuật để tạo ra các phiên bản phù hợp với yêu cầu của khai phá dữ liệu trở nên ngày càng quan trọng.

- *Dữ liệu thiếu và nhiễu*: Mức độ nhiễu cao trong dữ liệu điều này dẫn đến việc dự đoán thiếu chính xác.

- *Vấn đề “quá phù hợp” (Overfitting)*: Khi thuật toán khai phá tìm kiếm với các tham số tốt nhất cho một mô hình đặc biệt và một giới hạn của tập dữ liệu. Mô hình đó có thể “Quá phù hợp” trên tập dữ liệu đó nhưng lại thi hành không chính xác trên tập dữ liệu kiểm tra.

- *Sự thay đổi của dữ liệu và tri thức*: Dữ liệu là không tĩnh, dữ liệu thay đổi nhanh chóng có thể dẫn đến những tri thức đã khai phá trước đây trở nên không còn phù hợp thậm chí là vô giá trị.

- *Dánh giá các mẫu dữ liệu tìm được*: Nhiều mẫu phát hiện không thực sự hữu ích với người sử dụng và thách thức với các hệ khai phá dữ liệu.

- *Làm việc với các dữ liệu quan hệ phức tạp*: Do các hệ cơ sở dữ liệu quan hệ được sử dụng rộng rãi nên vấn đề làm tốt với các hệ cơ sở dữ liệu này là vấn đề cần quan tâm đối với các hệ khai phá dữ liệu.

- *Khai phá thông tin trong các hệ cơ sở dữ liệu hỗn hợp và hệ thống thông tin toàn cầu*: Với sự ra đời của mạng máy tính, dữ liệu có thể được thu thập từ nhiều nguồn khác nhau với định dạng khác nhau với số lượng rất lớn. Việc phát hiện tri thức từ các dạng dữ liệu hỗn hợp này là một thách thức đối với khai phá dữ liệu.

#### **1. 4. Kết luận chương 1**

Khai phá dữ liệu là sự vận dụng học thuật vào các vấn đề thiết thực đang diễn ra. Khai phá dữ liệu là tiến trình khái quát các sự kiện rời rạc trong dữ liệu thành các tri thức mang tính khái quát, tính quy luật, hỗ trợ tích cực cho việc ra quyết định. Nghiên cứu nhằm xây dựng và cải thiện các kỹ thuật trong khai phá dữ liệu là một lĩnh vực hứa hẹn và phù hợp với điều kiện nghiên cứu ở Việt Nam. Khai phá dữ liệu là một ngành khá non trẻ, các kỹ thuật của ngành còn chưa có khả năng giải quyết hiệu quả tốt các bài toán thực tế. Việc nghiên cứu cải thiện các giải thuật nhằm đưa ra các kỹ thuật mới là một khả năng có thể thực hiện trong môi trường làm việc còn thiêng thốn ở Việt Nam. Một số hướng nghiên cứu về lý thuyết trong khai phá dữ liệu đang được nghiên cứu hiện nay: Áp dụng các chiến lược để cải thiện hiệu quả các giải thuật. Phát triển các phiên bản mới của các giải thuật có khả năng giải quyết các tập dữ liệu lớn bằng kỹ thuật sử dụng bộ đệm. Song song và phân bố các giải thuật trong khai phá dữ liệu để tận dụng khả năng tính toán mạnh của tính toán lướt, ...

## CHƯƠNG 2

### KHAI PHÁ CÁC LUẬT KẾT HỢP SONG SONG

#### **2. 1. Luật kết hợp trong khai phá dữ liệu**

Luật kết hợp là một hướng quan trọng trong khai phá dữ liệu. Luật kết hợp giúp chúng ta tìm được các mối liên hệ giữa các mục dữ liệu (items) của cơ sở dữ liệu. Luật kết hợp là dạng khá đơn giản nhưng lại mang khá nhiều ý nghĩa. Thông tin mà dạng luật này đem lại là rất đáng kể và hỗ trợ không nhỏ trong quá trình ra quyết định. Tìm các luật kết hợp mang nhiều thông tin từ cơ sở dữ liệu tác nghiệp là một trong những hướng tiếp cận chính của lĩnh vực khai phá dữ liệu [12].

##### **2. 1. 1. Một số hướng tiếp cận trong khai phá luật kết hợp**

Lĩnh vực khai phá luật kết hợp cho đến nay đã được nghiên cứu và phát triển theo nhiều hướng khác nhau.

###### **2. 1. 1. 1. Luật kết hợp nhị phân**

Luật kết hợp nhị phân (binary association rules hoặc boolean association rules) là hướng nghiên cứu đầu tiên của luật kết hợp. Hầu hết các nghiên cứu ở thời kỳ đầu về luật kết hợp đều liên quan đến luật kết hợp nhị phân. Trong dạng luật kết hợp này, các thuộc tính chỉ được quan tâm là có hay không xuất hiện trong giao tác của cơ sở dữ liệu chứ không quan tâm về “mức độ” xuất hiện. Ví dụ như khách hàng A mua 10 sản phẩm B hay 1 sản phẩm B được xem là như nhau. Thuật toán tiêu biểu nhất khai phá dạng luật này là thuật toán Apriori và các thuật toán thuộc họ Apriori [16]. Đây là dạng luật đơn giản và các luật khác cũng có thể chuyển về dạng luật này nhờ một số phương pháp như rời rạc hoá, mờ hoá,... Ví dụ về dạng luật này: “Nếu khách hàng mua sản phẩm A thì sẽ mua sản phẩm B với độ hỗ trợ 20% và độ tin cậy 80%”.

###### **2. 1. 1. 2. Luật kết hợp có thuộc tính số và thuộc tính danh mục**

Các thuộc tính của cơ sở dữ liệu thực tế có kiểu rất đa dạng: nhị phân, số, danh mục, ... Để phát hiện luật kết hợp có thuộc tính số và thuộc tính danh mục (quantitative and categorial association rules), các nhà nghiên cứu đã đề xuất một số phương pháp rời rạc hoá nhằm chuyển dạng luật này về dạng nhị phân để có thể áp dụng các thuật toán đã có [16]. Ví dụ về dạng luật này “Nếu là nữ và tuổi từ [30..50] thì mua thực phẩm”, với độ hỗ trợ là 20%, và độ tin cậy là 80%.

###### **2. 1. 1. 3. Luật kết nhiều mức**

Luật kết nhiều mức (multi-level association rules), với cách tiếp cận theo luật này sẽ tìm kiếm thêm những luật có dạng tổng quát hóa. Ví dụ ta diễn tả “áo măng tô” là một loại “áo mặc bên ngoài”, “áo len” là một loại “áo mặc bên ngoài”. Từ thực tế “người mua áo măng tô thì mua giày ống” và “người mua áo len thì mua giày ống”. Ta có thể phỏng đoán một luật tổng quát hơn: “Người mua áo mặc bên ngoài thì mua giày ống”. Như vậy dạng luật này là dạng luật tổng quát hóa của 2 luật trước. Luật “Người

mua áo mặc bên ngoài thì mua giày ống” là một luật có giá trị đối với nhu cầu của người sử dụng hiện thời, còn luật “người mua áo măng tô thì mua giày ống” và “người mua áo len thì mua giày ống” thì không có giá trị bằng luật tổng quát.Thêm vào đó, luật tổng quát có thể ở nhiều mức khác nhau.

#### **2. 1. 1. 4. Luật kết hợp mờ**

Với những hạn chế còn gặp phải trong quá trình rời rạc hoá các thuộc tính số (quantitative attributes), các nhà nghiên cứu đã đề xuất luật kết hợp mờ (fuzzy association rules) [16] nhằm khắc phục các hạn chế trên và chuyển luật kết hợp về một dạng tự nhiên hơn, gần gũi hơn với người sử dụng.

#### **2. 1. 1. 5. Luật kết với thuộc tính được đánh trọng số**

Trong thực tế, các thuộc tính trong cơ sở dữ liệu không phải lúc nào cũng có vai trò như nhau. Có một số thuộc tính được chú trọng hơn và có mức độ quan trọng cao hơn các thuộc tính khác. Khi đó, trong quá trình tìm kiếm luật, chúng ta có thể gán thuộc tính này có trọng số lớn hơn thuộc tính kia. Đây là hướng nghiên cứu rất thú vị và đã được một số nhà nghiên cứu đề xuất cách giải quyết bài toán này. Với luật kết hợp có thuộc tính được đánh trọng số, chúng ta sẽ khai thác được những luật “hiếm” (tức là có độ hỗ trợ thấp, nhưng có ý nghĩa đặc biệt hoặc mang rất nhiều ý nghĩa).

#### **2. 1. 1. 6. Luật kết hợp song song**

Bên cạnh khai phá luật kết hợp tuần tự, các nhà làm tin học cũng tập trung vào nghiên cứu các thuật giải song song để phát hiện luật kết hợp, đó là Luật kết hợp song song (parallel mining of association rules) [16]. Nhu cầu song song hoá và xử lý phân tán là cần thiết bởi kích thước dữ liệu ngày càng lớn hơn nên đòi hỏi tốc độ xử lý cũng như dung lượng bộ nhớ của hệ thống phải được đảm bảo. Có rất nhiều thuật toán song song khác nhau đã đề xuất để có thể không phụ thuộc vào phần cứng. Bên cạnh những nghiên cứu về những biến thể của luật kết hợp, các nhà nghiên cứu còn chú trọng đề xuất những thuật toán nhằm tăng tốc quá trình tìm kiếm tập phô biến từ cơ sở dữ liệu.

Ngoài ra, còn có một số hướng nghiên cứu khác về khai phá luật kết hợp như: Khai phá luật kết hợp trực tuyến, khai phá luật kết hợp được kết nối trực tuyến đến các kho dữ liệu đa chiều (Multidimensional data, data warehouse) thông qua công nghệ OLAP (On-Line Analysis Processing), MOLAP (multidimensional OLAP), ROLAP (Relational OLAP), ...

#### **2. 1. 2. Các tính chất của luật kết hợp**

Cho D là cơ sở dữ liệu giao dịch

$I = \{i_1, i_2, \dots, i_n\}$  là tập bao gồm n mục phân biệt (Item - còn gọi là các thuộc tính - attribute).  $X \subseteq I$  được gọi là tập mục (itemset).

$T = \{t_1, t_2, \dots, t_m\}$  là tập gồm m giao dịch (Transaction - còn gọi là bản ghi - record), mỗi giao dịch có một định danh duy nhất được ký hiệu là TID (Transaction

**Identification).** Mỗi giao dịch được định nghĩa như một tập con (subset) các mục trong  $I$  ( $T \subseteq I$ ) và có dạng  $\langle TID, i_1, i_2, \dots, i_k \rangle$

Một giao dịch  $T \in D$  hỗ trợ (support) cho một tập mục  $X$ ;  $X \subseteq I$  nếu nó có chứa tất cả các mục của  $X$ , nghĩa là  $X \subseteq T$ . Trong một số trường hợp, người ta dùng ký hiệu  $T(X)$  để chỉ tập các giao dịch hỗ trợ cho  $X$ .

Ký hiệu  $\text{support}(X)$  (Viết gọn là  $\text{sup}(X)$ ) - Độ hỗ trợ (support) của một tập mục  $X$  – là tỷ lệ phần trăm số giao dịch trong cơ sở dữ liệu  $D$  chứa  $X$  trên tổng số các giao dịch trong cơ sở dữ liệu  $D$ .

$$\text{sup}(X) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|} \quad (1)$$

**Định nghĩa 1:** Tập mục phổ biến: Cho một tập mục  $X \subseteq I$  và ngưỡng hỗ trợ tối thiểu (minimum support)  $\text{minsup} \in (0, 1]$  ( $\text{minsup}$  được xác định bởi người sử dụng). Tập mục  $X$  được gọi là một tập phổ biến (hay Frequent Itemset hoặc Large Itemset) theo ngưỡng  $\text{minsup}$  nếu và chỉ nếu độ hỗ trợ của nó lớn hơn hoặc bằng ngưỡng  $\text{minsup}$ :  $\text{sup}(X) \geq \text{minsup}$ .

Một tập mục phổ biến được sử dụng như là một tập đáng quan tâm trong các thuật toán, các tập mục không phải là tập mục phổ biến là những tập không đáng quan tâm. Người ta dùng cụm từ “ $X$  có độ hỗ trợ tối thiểu” hoặc “ $X$  không có độ hỗ trợ tối thiểu” để nói lên  $X$  thỏa mãn hay không thỏa mãn  $\text{sup}(X) \geq \text{minsup}$ .

Một tập mục  $X$  được gọi là  $k$ -Itemset nếu lực lượng của  $X$  bằng  $k$  ( $|X| = k$ ).

### Tính chất liên quan đến tập mục phổ biến

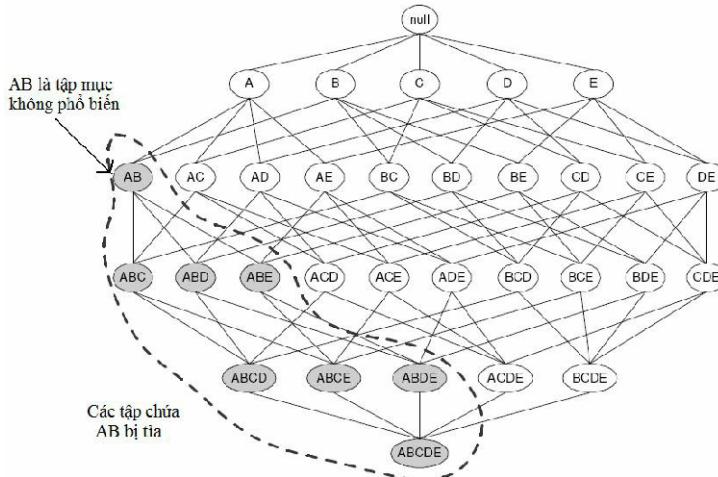
#### Tính chất 1: Độ hỗ trợ cho các tập con (Support for Subsets)

Giả sử  $A, B$  là các tập mục, nếu  $A \subseteq B$  thì  $\text{sup}(A) \geq \text{sup}(B)$  vì tất cả các giao dịch của  $D$  hỗ trợ  $B$  thì cũng hỗ trợ  $A$ .

**Tính chất 2:** Nếu một tập mục là tập mục không phổ biến thì mọi tập chứa nó không là tập mục phổ biến (Supersets of Infrequent Sets are Infrequent).

Nếu một tập mục  $B$  không có độ hỗ trợ tối thiểu trên  $D$ , tức là  $\text{sup}(B) < \text{minsup}$  thì mọi tập cha  $A$  của  $B$  cũng không phải là tập mục phổ biến vì  $\text{sup}(A) \leq \text{sup}(B) < \text{minsup}$ .

Tính chất này được áp dụng rất hiệu quả trong các thuật toán khai phá luật kết hợp ví dụ như Apriori.



Hình 2.1. Tập chứa tập mục không phổ biến là không phổ biến

**Tính chất 3:** Tập con của tập mục phổ biến cũng là tập mục phổ biến (Subsets of Frequent Sets are Frequent).

Nếu một tập mục B là một tập mục phổ biến trên D nghĩa là  $\text{sup}(B) \geq \text{minsup}$  thì mọi tập con A của B đều là tập phổ biến trên D vì  $\text{sup}(A) \geq \text{sup}(B) > \text{minsup}$ .

**Định nghĩa 2:** Một luật kết hợp là một quan hệ có dạng  $X \Rightarrow Y$ ; trong đó  $X, Y \subset I$  là các tập mục hay còn gọi là itemset và  $X \cap Y = \emptyset$ . Trong đó X là tiền đề, Y là hệ quả của luật. Luật kết hợp có hai thông số quan trọng là độ hỗ trợ và độ tin cậy.

**Định nghĩa 3:** Độ hỗ trợ (support) của luật kết hợp  $X \Rightarrow Y$  là tỷ lệ phần trăm giữa các giao dịch chứa  $X \cup Y$  và tổng số các giao dịch có trong cơ sở dữ liệu, được ký hiệu và tính theo công thức:

$$\text{sup}(X \Rightarrow Y) = P_r(X \cup Y) = \frac{|T \in D | X \cup Y \subseteq T|}{|D|} \quad (2)$$

Khi nói độ hỗ trợ của luật bằng 6% nghĩa là có 6% tổng số giao dịch có chứa  $X \cup Y$ . Độ hỗ trợ mang ý nghĩa thống kê của luật kết hợp.

**Định nghĩa 4:** Độ tin cậy (confidence) đối với một giao dịch của  $X \Rightarrow Y$  là tỷ lệ phần trăm giữa các giao dịch chứa  $X \cup Y$  và số giao dịch có chứa X, được ký hiệu và tính theo công thức:

$$\text{conf}(X \Rightarrow Y) = p(Y \subseteq I | X \subseteq I) = \frac{p(Y \subseteq T \wedge X \subseteq T)}{p(X \subseteq T)} = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)} \quad (3)$$

Khi nói một luật có độ tin cậy  $\text{conf} = 80\%$  có nghĩa là 80% các giao dịch hỗ trợ X thì cũng hỗ trợ cho Y. Về mặt xác suất, độ tin cậy của một luật kết hợp là xác suất (có điều kiện) xảy ra Y với điều kiện đã xảy ra X. Độ tin cậy của luật cho biết mức độ tương quan trong tập dữ liệu giữa hai tập mục X và Y, là tiêu chuẩn đánh giá độ tin cậy một luật.

Khai phá luật kết hợp từ cơ sở dữ liệu D chính là tìm tất cả các luật có độ hỗ trợ lớn hơn ngưỡng hỗ trợ (độ hỗ trợ tối thiểu minsup) và có độ tin cậy lớn hơn ngưỡng tin cậy (độ tin cậy tối thiểu minconf), có nghĩa là phải có  $\text{sup}(X \Rightarrow Y) \geq \text{minsup}$  và  $\text{conf}(X \Rightarrow Y) \geq \text{minconf}$ . Nếu luật  $X \Rightarrow Y$  thỏa mãn trên D thì cả X và Y đều là các tập mục phổ biến trên D.

### Tính chất liên quan đến luật kết hợp

**Tính chất 4:** Giả sử  $X, Y, Z \subseteq I$  là những tập mục, sao cho  $X \cap Y = \emptyset$ . Thì:  $\text{conf}(X \Rightarrow Y) \geq \text{conf}(X \setminus Z \Rightarrow Y \cup Z)$ .

Thật vậy, từ  $X \cup Y \subseteq X \cup Y \cup Z$  và  $X \setminus Z \subseteq X$  ta có:

$$\frac{\text{sup}(X \cup Y \cup Z)}{\text{sup}(X \setminus Z)} \leq \frac{\text{sup}(X \cup Y)}{\text{sup}(X)}$$

**Tính chất 5:** Luật kết hợp không có tính chất hợp thành.

Tức là nếu  $X \Rightarrow Z$  và  $Y \Rightarrow Z$  thỏa mãn trên D thì không nhất thiết  $X \cup Y \Rightarrow Z$  là đúng.

Thật vậy, xét trường hợp  $X \cap Y = \emptyset$  và các giao dịch trong D có hỗ trợ cho Z nếu và chỉ nếu chúng chỉ chứa mỗi X hoặc Y, khi đó  $\text{conf}(X \cup Y \Rightarrow Z) = 0$ .

Tương tự ta cũng có: nếu  $X \Rightarrow Y$  và  $Z \Rightarrow Y$  thỏa mãn trên D thì ta cũng có thể suy ra  $X \Rightarrow Y \cup Z$ .

**Tính chất 6:** Luật kết hợp không có tính chất tách

Nếu  $X \cup Y \Rightarrow Z$  thỏa mãn trên D thì không nhất thiết  $X \Rightarrow Z$  và  $Y \Rightarrow Z$  thỏa mãn trên D.

Ví dụ, khi Z có mặt trong giao dịch chỉ khi cả X và Y đều có mặt trong giao dịch đó, nghĩa là  $\text{sup}(X \cup Y) = \text{sup}(Z)$ . Nếu  $\text{sup}(X) \geq \text{sup}(X \cup Y)$  và  $\text{sup}(Y) \geq \text{sup}(X \cup Y)$  thì hai luật trên sẽ không có độ tin cậy yêu cầu. Nhưng nếu  $X \Rightarrow Y \cup Z$  thỏa trên D thì có thể suy ra  $X \Rightarrow Y$  và  $X \Rightarrow Z$  cũng thỏa trên D vì  $\text{sup}(XY) \geq \text{sup}(XYZ)$  và  $\text{sup}(XZ) \geq \text{sup}(XYZ)$ .

**Tính chất 7:** Luật kết hợp không có tính bắc cầu.

Có nghĩa là, nếu  $X \Rightarrow Y$  và  $Y \Rightarrow Z$  thỏa mãn trên D thì không thể khẳng định là  $X \Rightarrow Z$  cũng thỏa mãn trên D.

Giả sử  $T(X) \subseteq T(Y) \subseteq T(Z)$  và  $\text{conf}(X \Rightarrow Y) = \text{conf}(Y \Rightarrow Z) = \text{minconf}$ .

Khi đó,  $\text{conf}(X \Rightarrow Z) = \text{minconf}^2 < \text{minconf}$  (vì  $0 < \text{minconf} < 1$ ), suy ra luật  $X \Rightarrow Z$  không có độ tin cậy tối thiểu, tức là  $X \Rightarrow Z$  không thỏa trên D.

**Tính chất 8:** Nếu luật  $A \Rightarrow (L - A)$  không có độ tin cậy tối thiểu thì cũng không có luật nào trong các luật  $B \Rightarrow (L - B)$  có độ tin cậy tối thiểu, với  $L, A, B$  là các tập mục và  $B \subseteq A$ .

Thật vậy, sử dụng tính chất 1 ta có  $\text{sup}(B) \geq \text{sup}(A)$  (vì  $B \subseteq A$ ) và theo định nghĩa của độ tin cậy, ta có:

$$\text{conf}(B \Rightarrow (L - B)) = \frac{\text{sup}(L)}{\text{sup}(B)} \leq \frac{\text{sup}(L)}{\text{sup}(A)} \leq \text{minconf}$$

Tương tự, nếu luật  $(L - C) \Rightarrow C$  thỏa trên  $D$ , thì các luật  $(L - K) \Rightarrow K$  với  $K \subseteq C, K \neq \emptyset$  cũng thỏa trên  $D$ .

### 2. 1. 3. Bài toán khai phá luật kết hợp

Cho trước một tập các mục  $I$ , một cơ sở dữ liệu  $D$ , ngưỡng hỗ trợ  $\text{minsup}$  và ngưỡng tin cậy  $\text{minconf}$ . Tìm tất cả các tập luật kết hợp  $X \Rightarrow Y$  trên  $D$  thỏa mãn:

$$\text{sup}(X \Rightarrow Y) \geq \text{minsup} \text{ và } \text{conf}(X \Rightarrow Y) \geq \text{minconf}$$

*Bài toán có thể phát biểu như sau:*

**Dữ liệu vào:**  $I, D, \text{minsup}, \text{minconf}$ .

**Dữ liệu ra:** Các luật kết hợp thỏa mãn  $\text{minsup}$  và  $\text{minconf}$ .

Thuật toán thực hiện qua 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 có độ hỗ trợ lớn hơn hoặc bằng  $\text{minsup}$ .
- ❖ Pha 2: Sinh các luật kết hợp từ các tập phổ biến đã tìm thấy ở pha 1 sao cho độ tin cậy của luật lớn hơn hoặc bằng  $\text{minconf}$ .

### 2. 1. 4. Một số thuật toán khai phá luật kết hợp

#### 2. 1. 4. 1. Tìm tập mục phổ biến (Pha 1)

Phát hiện tập mục phổ biến là bước quan trọng và mất nhiều thời gian nhất trong quá trình khai phá luật kết hợp trong cơ sở dữ liệu.

##### 2. 1. 4. 1. 1. Thuật toán Apriori

Apriori là thuật toán được Rakesh Agrawal, Tomasz Imielinski, Arun Swami đề xuất lần đầu vào năm 1993 [5].

Ký hiệu	Ý nghĩa
k-itemset	Tập mục có k mục
$L_k$	Tập các k-mục phổ biến (large k-itemset) (tức tập các itemset có độ hỗ trợ lớn hơn hoặc bằng $\text{minsup}$ và có lực lượng bằng k). Mỗi

	phần tử của tập này có hai trường: - Tập mục (itemset). - Độ hỗ trợ tương ứng (support-count (SC)).
$C_k$	Tập các tập k-itemset ứng cử viên (gọi là tập các tập phổ biến tiềm năng). Mỗi phần tử của tập này có hai trường: - Tập mục (itemset). - Độ hỗ trợ tương ứng (support-count (SC)).

Bảng 2. 1. Một số ký hiệu dùng trong thuật toán Apriori

Nội dung thuật toán Apriori được trình bày như sau:

**Dữ liệu vào:** Tập các giao dịch D, ngưỡng support tối thiểu minsup

**Dữ liệu ra:** L- tập mục phổ biến trong D

**Phương pháp:**

$L_1 = \{\text{large 1-itemset}\}$  //tìm tất cả các tập mục phổ biến: nhận được  $L_1$

**for** ( $k=2$ ;  $L_{k-1} \neq \emptyset$ ;  $k++$ ) **do**

**begin**

$C_k = \text{apriori-gen}(L_{k-1})$ ; //sinh ra tập ứng cử viên từ  $L_{k-1}$

**for** (mỗi một giao dịch  $T \in D$ ) **do**

**begin**

$C_T = \text{subset}(C_k, T)$ ; //lấy tập con của T là ứng cử viên trong  $C_k$

**for** (mỗi một ứng cử viên  $c \in C_T$ ) **do**

$c.\text{count}++$ ; //tăng bộ đếm lần xuất 1 đơn vị

**end**

$L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}^* |D|\}$ ;

**end**

**return**  $\cup_k L_k$ ;

Trong thuật toán này, giai đoạn đầu đơn giản chỉ là việc đếm support-count cho các item. Để xác định tập 1-item phổ biến ( $L_1$ ), người ta chỉ giữ lại các item mà độ hỗ trợ của nó lớn hơn hoặc bằng  $minsup$ .

Trong mỗi giai đoạn tiếp theo, người ta bắt đầu với tập các tập phổ biến đã tìm được trong giai đoạn trước để lại sinh ra tập các tập mục có khả năng là phổ biến mới (gọi là tập các ứng cử viên - candidate itemset) và thực hiện đếm support-count cho mỗi tập các ứng cử viên trong tập này bằng một phép duyệt trên cơ sở dữ liệu. Tại

điểm kết của mỗi giai đoạn, người ta xác định xem trong các tập ứng viên này, tập nào là phô biến và lập thành tập các tập phô biến cho giai đoạn tiếp theo. Cụ thể là, trong các giai đoạn thứ k sau đó ( $k > 1$ ), mỗi giai đoạn gồm có 2 pha. Trước hết các  $(k-1)$ -itemset trong tập  $L_{k-1}$  được sử dụng để sinh ra các ứng viên  $C_k$ , bằng cách thực hiện hàm *apriori\_gen*. Tiếp theo cơ sở dữ liệu D sẽ được quét để tính độ hỗ trợ cho mỗi ứng viên trong  $C_k$ . Để việc đếm được nhanh, cần phải có một giải pháp hiệu quả để xác định các ứng viên trong  $C_k$  là có mặt trong một giao dịch T cho trước. Tiến trình này sẽ được tiếp tục cho đến khi không tìm được một tập phô biến nào mới hơn nữa.

Để tìm hiểu các thuật toán, ta giả sử rằng, các item trong mỗi giao dịch đã được sắp xếp theo thứ tự từ điển (người ta sử dụng khái niệm từ điển ở đây để diễn đạt một thứ tự quy ước nào đó trên các item của cơ sở dữ liệu). Mỗi bản ghi - record của cơ sở dữ liệu D có thể coi như là một cặp  $\langle TID, itemset \rangle$  trong đó TID là định danh cho giao dịch. Các item trong một itemset cũng được lưu theo thứ tự từ điển, nghĩa là nếu kí hiệu  $c$  item cùm một  $k$ -itemset  $c$  là  $c[1], c[2], \dots, c[k]$ , thì  $c[1] < c[2] < \dots < c[k]$ . Nếu  $c = X.Y$  và  $Y$  là một  $m$ -itemset thì  $Y$  cũng được gọi là  $m$ -extension (mở rộng) của  $X$ . Trong lưu trữ, mỗi itemset có một trường support-count tương ứng, đây là trường chứa số đếm độ hỗ trợ cho tập mục này.

Vấn đề sinh tập ứng viên (candidate) của Apriori dùng hàm *apriori\_gen*:

Hàm *apriori\_gen* với đối số là  $L_{k-1}$  (tập các large( $k-1$ )-itemset) sẽ cho lại kết quả là một siêu tập - superset, tức là tập của tất cả các large  $k$ -itemset. Sơ đồ sau là thuật toán cho hàm này.

**Dữ liệu vào:** tập mục phô biến  $L_{k-1}$  có kích thước  $(k-1)$

**Dữ liệu ra:** tập ứng cử viên  $C_k$

**Phương pháp:**

```

function apriori-gen( $L_{k-1}$ : tập mục phô biến có kích thước  $k-1$ )
begin
    // bước nối
    for (mỗi  $L_1 \in L_{k-1}$ ) do
        for (mỗi  $L_2 \in L_{k-1}$ ) do
            begin
                if  $((L_1[1]=L_2[1]) \cap (L_1[2]=L_2[2]) \cap \dots \cap (L_1[k-2]=L_2[k-2]) \cap$ 
                 $(L_1[k-1]=L_2[k-1]))$  then
                     $c = L_1 \oplus L_2$ ; // kết nối  $L_1$  với  $L_2$  sinh ra ứng cử viên  $c$ 
                    if has_infrequent_subset( $c, L_{k-1}$ ) then
                        remove ( $c$ ); // bước tia (xoá ứng cử viên  $c$ )
            
```

```

else  $C_k = C_k \cup \{c\}$ ; //kết tập c vào  $C_k$ 
end
return  $C_k$ ;
end

```

Hàm **has\_infrequent\_subset** kiểm tra tập con (k-1)-item của ứng cử viên k-item không là tập phô biến:

```

function has_infrequent_subset( $c$ : ứng viên k-item;  $L_{k-1}$  tập phô biến (k-1)-item)
begin
    //sử dụng tập mục phô biến trước
    for (mỗi tập con (k-1)-item  $s \in c$ ) do
        if  $s \notin L_{k-1}$  then return TRUE;
    end

```

Có thể mô tả hàm apriori\_gen gồm 2 bước sau:

**Bước nối (join step):** tìm  $L_k$  là tập k-item ứng viên được sinh ra bởi việc kết nối  $L_{k-1}$  với chính nó cho kết quả là  $C_k$ . Giả sử  $L_1, L_2$  thuộc  $L_{k-1}$ . Ký hiệu  $L_i^j$  là mục thứ j trong  $L_i$ . Điều kiện là các tập mục hay các mục trong giao dịch có thứ tự. Bước kết nối như sau: Các thành phần  $L_{k-1}$  kết nối (nếu có chung (k-2)-item đầu tiên) tức là:

$$(L_1[1]=L_2[1]) \cap (L_1[2]=L_2[2]) \cap \dots \cap (L_1[k-2]=L_2[k-2]) \cap (L_1[k-1]=L_2[k-1]).$$

**Bước tia (prune step):**  $C_k$  là tập chứa  $L_k$  (có thể là tập phô biến hoặc không) nhưng tất cả tập k-item phô biến được chứa trong  $C_k$ . Bước này, duyệt lần hai cơ sở dữ liệu để tính độ hỗ trợ cho mỗi ứng cử trong  $C_k$  sẽ nhận được  $L_k$ . Tuy nhiên để khắc phục khó khăn, giải thuật Apriori sử dụng các tính chất:

- 1 - Tất cả các tập con khác rỗng của một tập mục phô biến là phô biến;
- 2 - Nếu  $L$  là tập mục không phô biến thì mọi tập chứa nó không phô biến.

Trong bước này, ta cần loại bỏ tất cả các k-itemset  $c \in C_k$  mà chúng tồn tại một (k-1)-itemset không có mặt trong  $L_{k-1}$ . Giải thích điều này như sau: giả sử  $s$  là một (k-1)-itemset của  $c$  mà không có mặt trong  $L_{k-1}$ . Khi đó,  $sup(s) < minsup$ . Mặt khác,  $c \supset s$  nên  $sup(c) < sup(s) < minsup$ . Vậy  $c$  không thể là một tập phô biến, nó cần phải loại bỏ khỏi  $C_k$ . Việc kiểm tra các tập con (k-1)-itemset có thể được thực hiện một cách nhanh chóng bằng cách duy trì một cây băm của tất cả các tập mục phô biến tìm thấy.

Ví dụ:  $L_3 = \{abc, abd, acd, ace, bcd\}$

Bước nối:  $L_3 * L_3$  ta có: abcd từ abc và abd; acde từ acd và ace

Bước tia: acde bị tia vì ade không có trong  $L_3$ . Vậy  $C_4 = \{abcd\}$

### Hàm subset( $C_k$ , T)

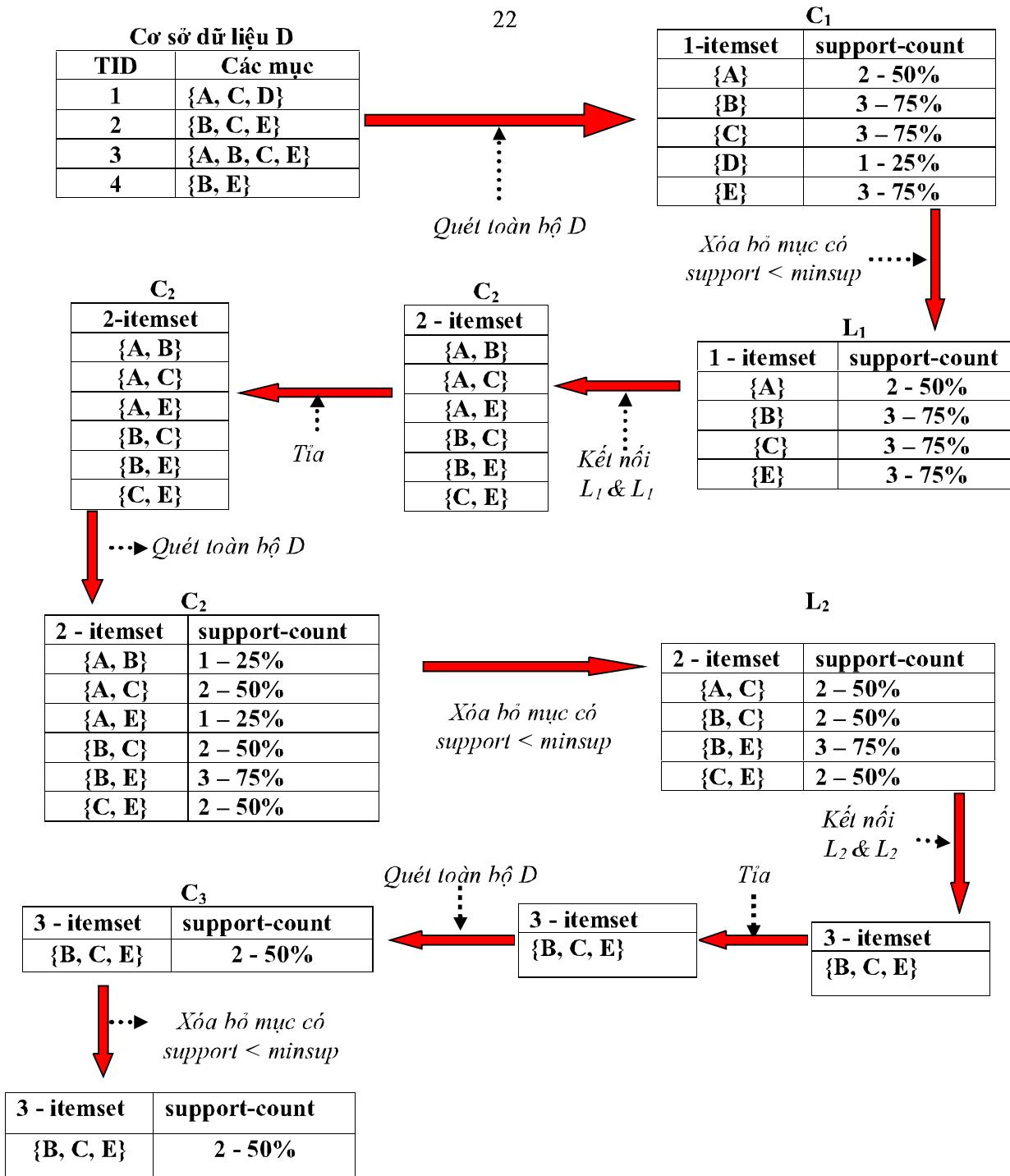
Hàm subset( $C_k$ , T) tìm tất cả các tập mục ứng viên trong  $C_k$  có chứa trong giao dịch T. Để tìm tập mục ứng viên ta bắt đầu từ nút gốc: nếu nút gốc là nút lá thì ta xem các tập mục trong nút lá đó có chứa trong giao dịch T không. Trường hợp là nút trong và là kết quả của việc áp dụng hàm băm cho mục thứ i của giao dịch T thì ta tiếp tục thực hiện hàm băm cho mục thứ i + 1 của giao dịch T cho đến khi gặp một nút lá. Thủ tục này được thực hiện đệ quy.

Nhận thấy, thuật toán Apriori với n là độ dài lớn nhất của tập mục được sinh ra, thuật toán sẽ duyệt toàn bộ cơ sở dữ liệu n + 1 lần. Vì thế, nếu bỏ qua thời gian so sánh để tìm sự xuất hiện của một tập mục trong một giao dịch thì độ phức tạp của thuật toán là  $O(n^*L)$ , trong đó L là kích thước cơ sở dữ liệu. Ngoài ra, nếu độ hỗ trợ tối thiểu minsup thay đổi thì thuật toán lại phải thực hiện lại từ đầu nên rất mất nhiều thời gian.

Ví dụ : Giả sử tập các item  $I = \{A, B, C, D, E\}$  và cơ sở dữ liệu giao dịch:

$$D = \{\langle 1, \{A, C, D\} \rangle, \langle 2, \{B, C, E\} \rangle, \langle 3, \{A, B, C, E\} \rangle, \langle 4, \{B, E\} \rangle\}.$$

Với minsup = 0.5 (tức tương đương 2 giao dịch). Khi thực hiện thuật toán Apriori trên ta có hình 2. 2:



Hình 2. 2. Minh họa thuật toán Apriori tìm tập mục phổ biến

#### 2. 1. 4. 1. 2. Các thuật toán thuộc họ Apriori

Một số thuật toán thuộc họ Apriori được đề xuất để tìm tập mục phổ biến.

##### Thuật toán AprioriTID [5] – [6]

Thuật toán AprioriTID là phần mở rộng theo hướng tiếp cận cơ bản của giải thuật Apriori. Thay vì dựa vào cơ sở dữ liệu thô giải thuật AprioriTID biểu diễn bên trong mỗi giao tác bởi các ứng viên hiện hành.

```

L1= {Large 1-itemset};

C' 1 = Database D;

for (k=2; Lk-1 ≠ Ø ; k++) do

begin

    Ck = apriori_gen(Lk-1);

    C' k = Ø;

    for tất cả t ∈ C' k-1 do

        begin

            // xác định tập ứng viên trong Ck chứa trong giao dịch với định

            // danh t. Tid (Transaction Code)

            Ct = {c ∈ Ck | (c-c[k]) ∈ t.Set_of_ItemSets ^ (c-c[k-1] ∈

            t.Set_of_ItemSets}

            for những ứng viên c ∈ Ct do c.count ++;

            if (Ct≠Ø) then C' k+= <t.Tid, Ct>

        end

        Lk = {c ∈ Ck | c.count ≥ minsup};

    end

    return = ∪kLk;

```

Thuật toán này cũng sử dụng hàm apriori\_gen để sinh ra các tập ứng cử viên cho mỗi giai đoạn. Nhưng thuật toán này không dùng cơ sở dữ liệu D để đếm các support-count với các giai đoạn k > 1 mà sử dụng tập C' <sub>k</sub>. Mỗi phần tử của C' <sub>k</sub> có dạng < Tid, {X<sub>k</sub>}>, trong đó mỗi X<sub>k</sub> là một tập phô biến k\_itemset tiềm năng trong giao dịch Tid. Khi k = 1, C' <sub>k</sub> tương ứng với D, trong đó mỗi item i được coi là một itemset {i}. Với k>1, C' <sub>k</sub> được sinh ra bởi C' <sub>k</sub> = <t.Tid, C<sub>t</sub>>. Phần tử của C' <sub>k</sub> tương ứng với giao dịch t là <t.Tid, {c| c chứa trong t}>. Nếu một giao dịch không chứa bất kỳ tập ứng viên k\_itemset nào thì C' <sub>k</sub> sẽ không có một điểm vào nào cho giao dịch này. Do đó, số lượng điểm vào trong C' <sub>k</sub> có thể nhỏ hơn số giao dịch trong cơ sở dữ liệu, đặc biệt với k lớn. Hơn nữa, với các giá trị k khá lớn, mỗi điểm vào có thể nhỏ hơn giao dịch tương ứng vì một số ứng viên đã được chứa trong giao dịch. Tuy nhiên, với các giá trị k nhỏ, mỗi điểm vào có thể lớn hơn giao dịch tương ứng vì một điểm vào trong C' <sub>k</sub> bao gồm tất cả các ứng viên k-itemset được chứa trong giao dịch. Đã có nhiều thí nghiệm chứng minh thuật toán Apriori cần ít thời gian hơn thuật toán AprioriTID trong giai đoạn đầu nhưng mất nhiều thời gian cho các giai đoạn sau.

### **Thuật toán AprioriHybrid [6]–[16]**

Thuật toán AprioriHybrid kết hợp cả hai hướng tiếp cận trên. Thuật toán AprioriHybrid dựa vào ý tưởng “không nhất thiết sử dụng cùng một thuật toán cho tất cả các giai đoạn trên dữ liệu”. Thuật toán này sử dụng thuật toán Apriori ở các giai đoạn đầu và chuyển sang sử dụng AprioriTID cho các giai đoạn sau. Thuật toán AprioriHybrid được coi là tốt hơn thuật toán Apriori và thuật toán AprioriTID.

Ngoài ra còn có một số các giải thuật tựa Apriori:

### **Thuật toán DIC [6]–[16]**

Thuật toán DIC (Dynamic Itemset Counting) là một biến thể khác nữa của giải thuật Apriori. Giải thuật DIC làm giảm việc đếm và việc phát sinh các ứng viên. Bất kỳ ứng viên nào tới được ngưỡng  $\text{minsup}$ , thì giải thuật DIC bắt đầu phát sinh thêm các ứng viên dựa vào nó. Để thực hiện điều này giải thuật DIC dùng một prefix-tree (cây tiền tố). Ngược với cây băm (hashtable), mỗi nút (nút lá hoặc nút trong) của prefix-tree được gán một ứng viên xác định trong tập phổ biến. Cách sử dụng cũng ngược với cây băm, bắt cứ khi nào tới được một nút ta có thể khẳng định rằng tập item đã kết hợp với nút này trong giao tác đó. Hơn nữa, việc xác định độ hỗ trợ và phát sinh ứng viên khớp nhau sẽ làm giảm đi số lần duyệt cơ sở dữ liệu.

### **Thuật toán OCD [5]–[16]**

Thuật toán OCD (Offline Candidate Deteremination) được giới thiệu ở Manila vào năm 1994. Thuật toán này dùng các kết quả của phép phân tích tổ hợp thông tin thu được ở giai đoạn trước để loại bỏ đi các tập mục ứng viên không cần thiết. Nếu một tập  $Y \subseteq I$  là một tập không phổ biến thì cần quét ít nhất  $(1-s)$  giao dịch trong cơ sở dữ liệu,  $s$  là ngưỡng hỗ trợ. Do đó, nếu ngưỡng hỗ trợ nhỏ thì hầu như toàn bộ các giao dịch phải được quét.

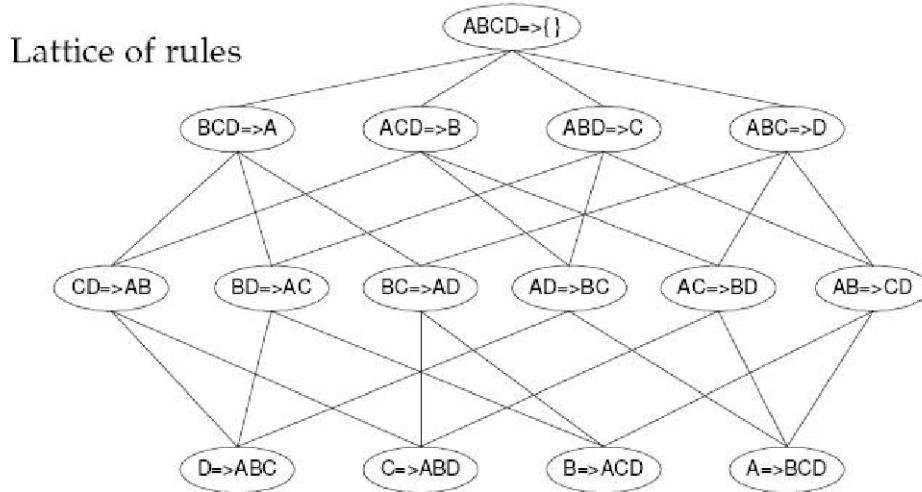
Ngoài các thuật toán khai phá luật kết hợp như thuật toán Apriori và các thuật toán thuộc họ Apriori còn có các thuật toán khác [16] như Partition, Sampling, CARMA (Continuous Association Rule Mining Algorithm), AIS, SETM, FP-Growth, Eclat (Equivalence CLass Transformation),...

### **2. 1. 4. 2. Sinh các luật kết hợp từ tập mục phổ biến (Pha 2)**

Việc phát hiện các tập mục phổ biến là rất tốn kém về mặt tính toán. Tuy nhiên, ngay khi tìm được tất cả các tập mục phổ biến ( $I \in L$ ), ta có thể sinh ra các luật kết hợp có thể có bằng các bước như sau:

- Tìm tất cả các tập con không rỗng  $a$ , của tập mục lớn  $I \in L$ .
- Với mỗi tập con  $a$  tìm được, ta xuất ra luật dạng  $a \Rightarrow (I - a)$  nếu tỷ số  $\text{Sup}(I)/\text{Sup}(a) \geq \text{minconf} (\%)$ .

Ví dụ:



Hình 2.3. Sinh luật từ tập mục phô biến

**Đầu vào:** Tập mục phô biến  $L_k$

**Đầu ra:** Tập luật thoả mãn độ tin cậy  $\geq \text{mincof}$  và độ hỗ trợ  $\geq \text{minsup}$

**Phương pháp:**

**forall**  $L_k, k \geq 2$  **do**

call genrules( $L_k, L_k$ );

**procedure** genrules( $L_k$ : large  $k$ -itemset,  $a_m$ : large  $m$ -itemset)

**begin**

$A = \{(m-1)\text{-itemset } a_{m-1} \mid a_{m-1} \subset a_m\}$

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

conf = sup( $L_k$ ) / sup( $a_{m-1}$ );

**if** (conf  $\geq \text{mincof}$ ) **then begin**

Xuất ra luật  $a_{m-1} \Rightarrow (L_k - a_{m-1})$ ;

**if** ( $m-1 > 1$ ) **then** call genrules( $L_k, a_{m-1}$ );

**end**

**end**

Thủ tục sinh luật nhanh: với mỗi tập mục phô biến  $l$ , ta có luật  $a \Rightarrow (l - a)$  có độ tin cậy nhỏ hơn minconf thì không cần xét luật  $a' \Rightarrow (l - a')$ , với  $\forall a' \subseteq a$ . Chẳng hạn, nếu  $ABC \Rightarrow D$  có độ tin cậy nhỏ hơn minconf thì không cần kiểm tra luật  $AB \Rightarrow CD$  vì  $AB \subset ABC$  nên  $\text{sup}(AB) \geq \text{sup}(ABC)$  và do đó  $\frac{\text{sup}(ABCD)}{\text{sup}(AB)} \leq \frac{\text{sup}(ABCD)}{\text{sup}(ABC)} < \text{minconf}$ .

Ngược lại, nếu luật  $(l - c) \Rightarrow c$  thỏa mãn minconf thì tất cả các luật  $(l - c') \Rightarrow c'$  cũng thỏa mãn minconf với  $\forall c' \subseteq c$ . Vì  $(l - c) \subseteq (l - c')$  nên suy ra  $\text{sup}(l - c) \geq \text{sup}(l - c')$  và  $\text{minconf} \leq \frac{\text{sup}(l)}{\text{sup}(l - c)} \leq \frac{\text{sup}(l)}{\text{sup}(l - c')}$ , do đó luật  $(l - c') \Rightarrow c'$  cũng thỏa mãn minconf.

Vì vậy, từ một tập phô biến  $l$ , đầu tiên, ta sinh tất cả các luật với 1-itemset trong mệnh đề kết quả. Sau đó sử dụng các mệnh đề kết quả và hàm *apriori\_gen()* để sinh các mệnh đề kết quả có thể của luật bao gồm 2-item, 3-item, ...[14].

#### 2. 1. 4. 3. Độ phức tạp của thuật toán khai phá luật kết hợp

Đánh giá độ phức tạp của thuật toán khi khai phá luật kết hợp là một bài toán khó. Nhiều kết quả phân tích cho thấy, với trường hợp cơ sở dữ liệu thưa, kích thước của giao dịch nhỏ thì độ phức tạp là tuyến tính với kích thước của cơ sở dữ liệu. Trường hợp còn lại thuộc lớp bài toán NP-đầy đủ [19]. Trong pha tìm tập mục phô biến, với  $m$  là số lượng tập mục, để tìm được tất cả các tập mục phô biến thì không gian tìm kiếm là  $2^m$ . Trong pha sinh luật kết hợp, với mỗi tập mục phô biến có kích cỡ  $k$  thì tạo ra  $2^k - 2$  khả năng sinh luật. Do đó, độ phức tạp của pha sinh luật là  $O(r * 2^k)$  với  $r$  là số lượng tập mục phô biến và  $1$  là tập mục phô biến dài nhất.

Ví dụ: Với giải thuật Apriori, việc tạo các tập mục ứng viên là lớn. Giả sử có  $10^4$  tập mục phô biến 1-itemset thì sẽ tạo ra  $10^7$  ứng viên 2-itemsets. Để khai phá tập mục phô biến có kích cỡ 100, ví dụ  $\{a_1, a_2, \dots, a_{100}\}$  cần tạo  $2^{100} \approx 10^{30}$  ứng viên. Số lần quét cơ sở dữ liệu là  $(n + 1)$  lần, trong đó  $n$  là độ dài của mẫu lớn nhất.

**Nhận xét:** Các giải thuật khai phá luật kết hợp tuần tự là khá đơn giản. Tuy nhiên, dữ liệu đang tăng rất nhanh về cả số chiều (số mục) và kích cỡ (số giao tác) mà các giải thuật tuần tự không đáp ứng được thay đổi đó, thời gian thực hiện thuật toán không phù hợp với thực tế. Vì thế, ta cần có các giải thuật khai phá luật kết hợp song song và phân tán để đáp ứng nhu cầu thực tế. Ngoài ra, quá trình tính toán song song được sự hỗ trợ rất lớn về cả nền tảng phần cứng và phần mềm.

#### 2. 2. Các thuật toán song song phát hiện luật kết hợp

Khai phá các luật kết hợp song song dựa trên ý tưởng của khai phá luật kết hợp, thực hiện song song hóa nhằm đáp ứng sự tăng lên nhanh chóng của dữ liệu và giảm thời gian thực hiện cho phù hợp với yêu cầu thực tế. Tuy nhiên, để thực hiện được các giải thuật song song tốt là điều không đơn giản. Thách thức có thể là quá trình đồng bộ hóa khi các bộ xử lý trao đổi với nhau, đảm bảo các chiến lược cân bằng tải, biểu diễn và kết hợp dữ liệu phù hợp, tối thiểu hóa việc đọc/ghi đĩa, ... Do đó, các giải thuật song song thiết kế cần phải đảm bảo các chiến lược cân bằng tải, nền tảng phần cứng, kiến trúc và cơ chế song song, ...

## 2. 2. 1. Thuật toán song song

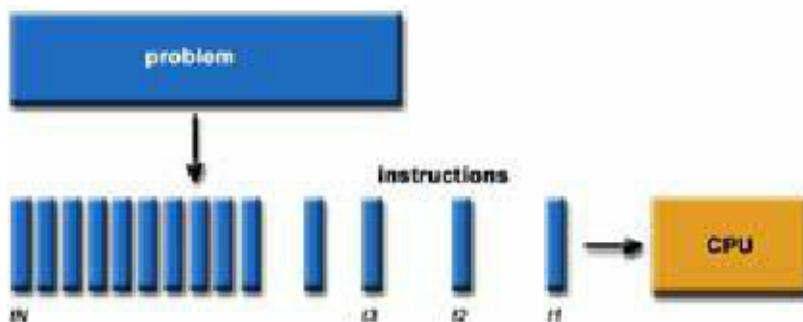
### 2. 2. 1. 1. Tính toán song song

Tính toán song song là một quá trình phát triển tiếp theo của tính toán tuần tự. Tính toán song song cho phép giả lập những gì thường xảy ra trong thế giới tự nhiên, rất nhiều sự kiện phức tạp, đan xen lẫn nhau, tác động lẫn nhau cùng xảy ra tại một thời điểm nằm trong một chuỗi trình tự.

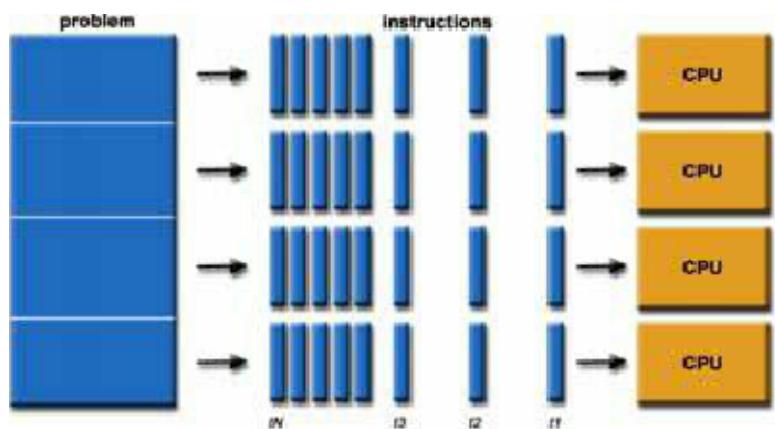
Các bài toán tính toán song song [1] thường có các đặc tính chung như sau: Cho phép chia nhỏ một công việc lớn thành nhiều phần việc nhỏ hơn và có thể giải quyết đồng thời. Tại một thời điểm, có thể thực thi nhiều chỉ thị chương trình, thời gian xử lý bài toán sẽ giảm xuống bởi nhiều tài nguyên tính toán được sử dụng.

Tính toán song song được áp dụng nhằm hai lý do chính: Tiết kiệm thời gian và giải quyết được bài toán lớn. Ngoài ra, còn có một số lý do khác như: tận dụng được các tài nguyên phi cục bộ (non-local), nếu máy tính được nối mạng, có thể sử dụng các tài nguyên tính toán trên mạng điện rộng và Internet, tiết kiệm chi phí bằng cách sử dụng nhiều tài nguyên tính toán giá rẻ thay thế cho việc sử dụng một siêu máy tính có giá thành cao, vượt qua được giới hạn về lượng bộ nhớ mà máy tính sử dụng vì nếu sử dụng nhiều máy tính khác nhau, chúng ta sẽ có lượng bộ nhớ không giới hạn.

Sau đây là hình ảnh mô tả về tính toán tuần tự và tính toán song song.



Hình 2. 4. Tính toán tuần tự



Hình 2. 5. Tính toán song song

### 2. 2. 1. 2. Nguyên lý thiết kế thuật toán song song

Khi nói đến xử lý song song là phải xét cả kiến trúc máy tính lẫn các thuật toán song song. Những thuật toán, trong đó có một số thao tác có thể thực hiện đồng thời được gọi là thuật toán song song. Tổng quát hơn, thuật toán song song là một tập các tiến trình hoặc các tác vụ có thể thực hiện đồng thời và có thể trao đổi dữ liệu với nhau để kết hợp cùng giải một bài toán đặt ra. Thuật toán song song có thể xem như là một tập hợp các đơn thể độc lập, một số trong số chúng có thể thực hiện tương tranh trên máy tính song song [1].

Có năm nguyên lý chính trong thiết kế thuật toán song song:

1. *Các nguyên lý lập lịch*: Giảm tối thiểu các bộ xử lý sử dụng trong thuật toán sao cho thời gian tính toán là không tăng (xét theo khía cạnh độ phức tạp).
2. *Nguyên lý hình ống*: Nguyên lý này được áp dụng khi bài toán xuất hiện một dãy các thao tác  $\{T_1, T_2, \dots, T_n\}$ , trong đó  $T_{i+1}$  thực hiện sau khi  $T_i$  kết thúc.
3. *Nguyên lý chia để trị*: Chia bài toán thành những phần nhỏ hơn tương đối độc lập với nhau và giải quyết chúng một cách song song.
4. *Nguyên lý đồ thị phụ thuộc dữ liệu*: Phân tích mối quan hệ dữ liệu trong tính toán để xây dựng đồ thị phụ thuộc dữ liệu và xây dựng thuật toán song song.
5. *Nguyên lý điều kiện tranh đua*: Nếu hai tiến trình cùng muốn truy cập vào cùng một mục dữ liệu chia sẻ thì cũng phải tương tranh với nhau, nghĩa là chúng có thể cản trở lẫn nhau.

Ngoài những nguyên lý nêu trên, khi thiết kế thuật toán song song còn một số điểm cần quan tâm:

- ❖ Hiệu quả thực hiện của thuật toán song song có thể rất khác nhau và yếu tố quan trọng nhất ảnh hưởng tới độ phức tạp tính toán là cấu hình tông liên kết mạng.
- ❖ Thuật toán song song phải được thiết kế dựa trên những kiến trúc về kiến trúc máy tính, ngôn ngữ lập trình song song và các phương pháp tính toán.

### 2. 2. 1. 3. Các cách tiếp cận trong thiết kế thuật toán song song

Có ba cách tiếp cận để thiết kế thuật toán song song là [1]:

1. Thực hiện song song hóa những thuật toán tuần tự, biến đổi những cấu trúc tuần tự để tận dụng được những khả năng song song tự nhiên của tất cả các thành phần trong hệ thống xử lý.
2. Thiết kế những thuật toán song song mới phù hợp với kiến trúc song song.
3. Xây dựng những thuật toán song song từ những thuật toán song song đã được xây dựng cho phù hợp với cấu hình tông và môi trường song song thực tế.

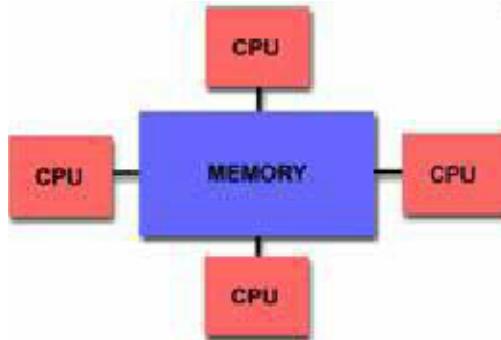
Như vậy, cách làm khá thông dụng là biến đổi các thuật toán tuần tự về song song, hay chuyển từ một dạng song song về dạng song song phù hợp hơn sao cho vẫn bảo toàn tính tương đương trong tính toán.

#### **2. 2. 1. 4. Kiến trúc bộ nhớ của máy tính song song**

##### **2. 2. 1. 4. 1. Bộ nhớ chia sẻ (Shared Memory)**

Các bộ xử lý có thể hoạt động độc lập nhưng truy nhập chung bộ nhớ. Các bộ xử lý khác có khả năng nhìn thấy các thay đổi trong bộ nhớ do một bộ xử lý tác động. Các máy tính có bộ nhớ chia sẻ có thể được chia thành 2 loại chính: *UMA* (Uniform Memory Access) và *NUMA* (Non Uniform Memory Access).

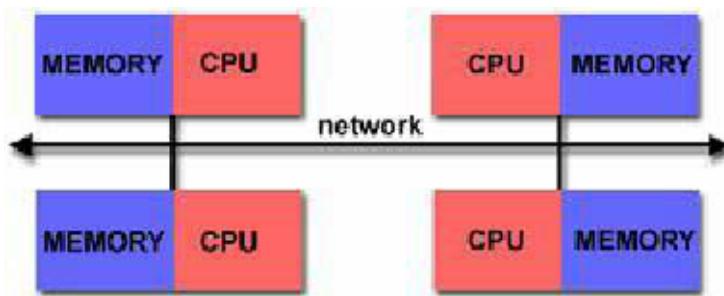
- ❖ Mô hình đa bộ xử lý truy xuất bộ nhớ đồng nhất (Uniform Memory Access (UMA) multi processor) -bộ nhớ chia sẻ tập trung.
- ❖ Mô hình đa bộ xử lý truy xuất bộ nhớ không đồng nhất (Non Uniform Memory Access (NUMA) multi processor) -bộ nhớ chia sẻ phân tán.



Hình 2. 6. Kiến trúc bộ nhớ chia sẻ

##### **2. 2. 1. 4. 2. Bộ nhớ phân tán (Distributed Memory)**

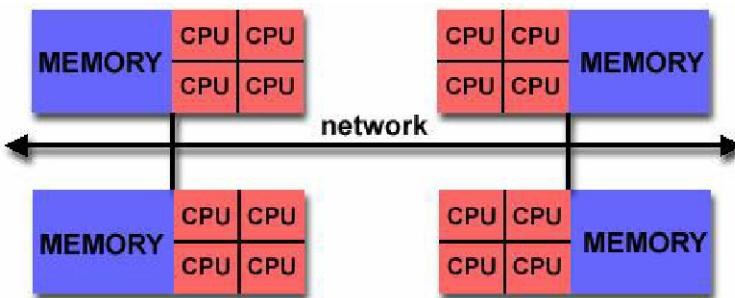
Các bộ xử lý có bộ nhớ riêng. Các hệ thống bộ nhớ phân tán đều đòi hỏi phải được kết nối mạng với nhau để nối kết các bộ nhớ liên bộ xử lý.



Hình 2. 7. Kiến trúc bộ nhớ phân tán

##### **2. 2. 1. 4. 3. Bộ nhớ lai (Hybrid Distributed-Shared Memory)**

Các máy tính lớn nhất và nhanh nhất ngày nay đều dùng cả 2 loại kiến trúc bộ nhớ phân tán và bộ nhớ chia sẻ.



Hình 2. 8. Kiến trúc bộ nhớ lai

## 2. 2. 1. 5. Mô hình song song

Có hai hướng tiếp cận chính: Mô hình song song dữ liệu và Mô hình song song thao tác

### 2. 2. 1. 5. 1. Mô hình song song dữ liệu

Mô hình song song dữ liệu thực thi thao tác giống nhau hay thực thi chỉ thị lệnh trên nhiều tập con dữ liệu cùng một thời điểm. Tất cả các bộ xử lý thực hiện chương trình giống nhau. Tuy nhiên, đối với chương trình này ta có thể sử dụng cấu trúc điều khiển if - then - else để chỉ định lệnh nào được thực thi bởi bộ xử lý nào, nghĩa là một số phần chương trình chỉ được thực hiện trên một hoặc vài bộ xử lý. Trong mô hình song song dữ liệu, dữ liệu cần phải phân chia thành các tập con dữ liệu, để tăng tốc đạt được bằng cách giảm khối lượng dữ liệu cần được xử lý trên mỗi bộ xử lý.

Các thuật toán được thiết kế dựa vào mô hình song song dữ liệu dễ dàng thực thi và năng suất, ít phụ thuộc vào kiến trúc máy tính song song. Tuy nhiên, mô hình song song dữ liệu cũng gặp khó khăn trong việc cân bằng tải công việc do sự chênh lệch dữ liệu.

### 2. 2. 1. 5. 2. Mô hình song song thao tác

Đối với mô hình song song thao tác, mỗi bộ xử lý thực thi tập chỉ thị khác nhau. Các chương trình phối hợp với nhau để hoàn thành cùng một mục tiêu, ý tưởng của mô hình song song thao tác là giảm độ phức tạp thao tác bằng cách chia thao tác thành các thao tác nhỏ hơn để thực thi và tập dữ liệu thực hiện trong mỗi chương trình không nhất thiết giống nhau.

## 2. 2. 2. Khai phá các luật kết hợp song song

### 2. 2. 2. 1 Các thuật toán song song phát hiện hiện tập mục phổ biến

Trong các thuật toán trình bày ở phần tiếp theo, sẽ sử dụng một số ký hiệu được mô tả như trong bảng.

I	Tập các mục phân biệt trong cơ sở dữ liệu giao dịch D
D <sub>1</sub> , D <sub>2</sub> , ..., D <sub>p</sub>	Các phân hoạch cơ sở dữ liệu D, p là số các bộ xử lý

minsup	Độ hỗ trợ tối thiểu
L	Tập các mục phổ biến

Bảng 2.2. Ký hiệu dùng trong các thuật toán song song

### 2.2.2.1.1. Thuật toán Count Distribution

Thuật toán Count Distribution [Agrawal 1996] sử dụng kiến trúc không chia sẻ, mỗi bộ xử lý có một bộ nhớ chính và một bộ nhớ phụ riêng. Các bộ xử lý được kết nối với nhau bởi một mạng truyền thông và có thể truyền tin cho nhau bằng phương pháp truyền thông điệp. Dựa vào 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 và số đếm hỗ trợ của các tập mục là không đầy đủ. Các số đếm hỗ trợ cục bộ được tính bởi các bộ xử lý trên các phân hoạch dữ liệu của nó. Số đếm hỗ trợ tổng thể được thiết lập thông qua mô hình truyền thông MPI.

**Nội dung của thuật toán:**

**Dữ liệu vào:** I,  $minsup$ ,  $D_1, D_2, \dots, D_p$

**Dữ liệu ra:** L

**Phương pháp:**

$C_1 = I;$

**for** ( $k=1; C_k \neq \emptyset; k++$ ) **do begin**

// bước 1: tính các số đếm hỗ trợ cục bộ

$\text{count}(C_k, D_i);$  // bộ xử lý cục bộ thứ i

// bước 2: trao đổi các số đếm hỗ trợ với các bộ xử lý khác để

// thu được các số đếm hỗ trợ tổng thể trong D

**forall** tập mục  $X \in C_k$  **do begin**

$$X.\text{count} = \sum_{j=1}^p X_j.\text{count};$$

**end**

// bước 3: Xác định các tập mục phổ biến và sinh các tập mục

// ứng viên  $C_{k+1}$

$L_k = \{c \in C_k | c.\text{count} \geq minsup * |D_1 \cup D_2 \cup \dots \cup D_p|\};$

$C_{k+1} = \text{apriori\_gen}(L_k);$

**end**

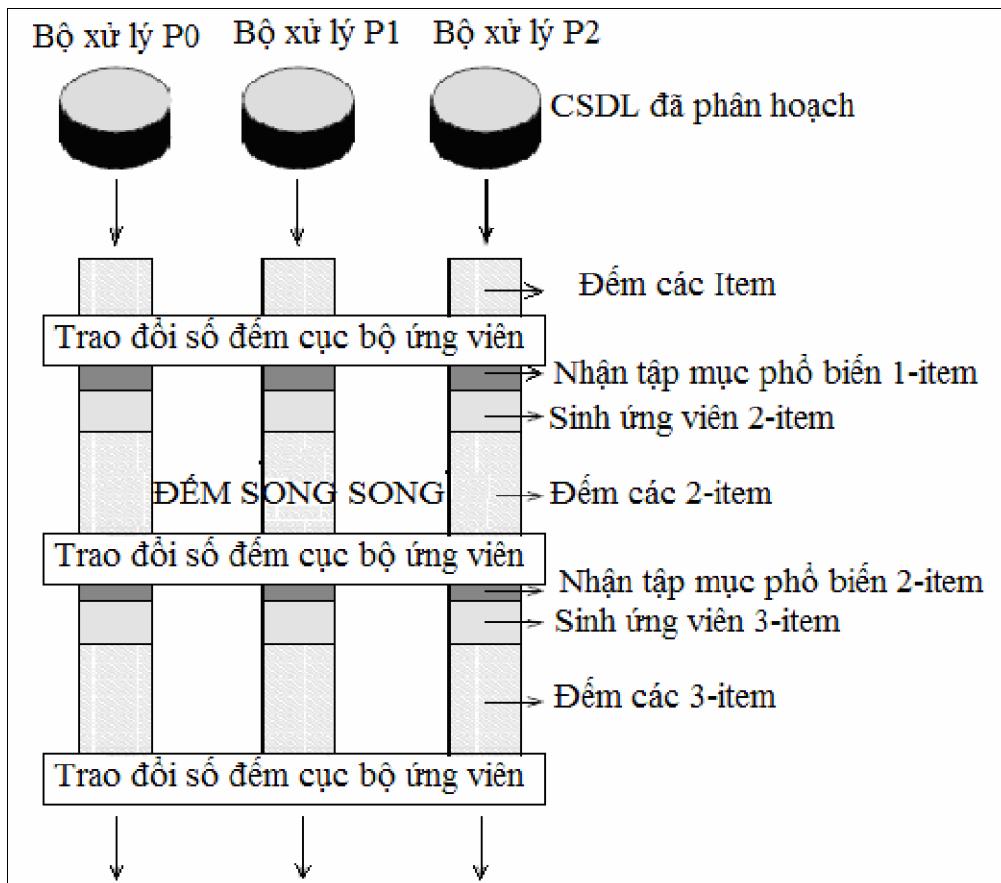
**return**  $L = L_1 \cup L_2 \cup \dots \cup L_k;$

Thuật toán Count Distribution thực hiện như sau:

Cơ sở dữ liệu D được phân hoạch thành  $\{D_1, D_2, \dots, D_p\}$  và phân bổ lần lượt cho các bộ xử lý  $P_i$  ( $1 \leq i \leq p$ ). Thuật toán thực hiện gồm 3 bước:

- Bước 1: Mỗi bộ xử lý  $P_i$  quét phân hoạch cơ sở dữ liệu cục bộ  $D_i$  để tính các số đếm hỗ trợ cục bộ cho các tập mục ứng viên  $C_k$ .
- Bước 2: Mỗi bộ xử lý  $P_i$  trao đổi các số đếm hỗ trợ cục bộ của các tập mục ứng viên để tính các số đếm hỗ trợ tổng thể của tất cả các tập mục ứng viên trong cơ sở dữ liệu D bằng cách sử dụng mô hình truyền thông điệp MPI.
- Bước 3: Các tập mục phổ biến tổng thể  $L_k$  được xác định dựa vào ngưỡng hỗ trợ  $minsup$  và các tập mục ứng viên  $C_{k+1}$  được sinh ra từ  $L_k$  bằng cách áp dụng thuật toán *apriori\_gen()* trên mỗi bộ xử lý một cách độc lập.

Thuật toán lặp lại bước 1 → 3 cho đến lúc không còn tập mục ứng viên nào sinh ra.



Hình 2. 9. Giải thuật Count Distribution

Ví dụ: Cho cơ sở dữ liệu giao dịch D, độ hỗ trợ tối thiểu  $minsup = 33\%$ . Tìm tập mục phổ biến.

Cơ sở dữ liệu D		Tập mục phổ biến (minsup = 33%)					
TID	Items	Tập mục phổ biến: độ hỗ trợ (support)					
1	f d b e	1	a:67% b:50% c:33% d:50% e:83% f:67%				
2	f e b	2	ac:33% ad:33% ae:50% af:33% bd:33% be:33% bf:33% ce:33% cf:33% de:33% ef:67%				
3	a d b	3	ace:33% acf:33% aef:33% bef:33% cef:33%				
4	a e f c	4	acef:33%				
5	a d e						
6	a c f e						

Hình 2. 10. Cơ sở dữ liệu D và các tập mục phổ biến

Thuật toán Count Distribution thực hiện theo thứ tự như hình 2. 11 với ví dụ như trong hình 2. 10. Phân hoạch cơ sở dữ liệu giao dịch D cho 3 bộ xử lý P<sub>0</sub>, P<sub>1</sub> và P<sub>2</sub>, mỗi bộ xử lý có 2 giao dịch.

Trong ví dụ trên, để khai phá tất cả các tập mục phổ biến thì thuật toán Count Distribution cần quét cơ sở dữ liệu nhiều lần. Mỗi bộ xử lý chỉ quét cơ sở dữ liệu cục bộ và đếm các tập mục ứng viên cục bộ đã được phân hoạch cho nó, việc này được xử lý song song. Tổng hợp các số đếm cục bộ cho ta số đếm tổng thể, được dùng để tính độ hỗ trợ của tập mục đó. Việc sinh ra và đếm các tập mục ứng viên được dựa trên các thủ tục giống như Apriori.

(a) Phân hoạch cơ sở dữ liệu			(b) Khai phá 1-itemset phổ biến			(c) Khai phá 2-itemset phổ biến		
TID	Items	Bộ xử lý			Số đếm	Số đếm	Số đếm	
		P0	P1	P2				
1	f d b e				a	0	2	2
2	f e b	P0			b	2	1	0
3	a d b				c	0	1	1
4	a e f c	P1			d	1	1	1
5	a d e				e	2	1	2
6	a c f e	P2			f	2	1	1

(d) Khai phá 3-itemset phổ biến			(e) Khai phá 4-itemset phổ biến			
3-itemset	Số đếm cục bộ		4-itemset	Số đếm cục bộ	Số đếm	
	P0	P1		P0	P1	
ace	0	1	1	0	1	2
acf	0	1	1	0	1	2
ade	0	0	1	0	0	1
aef	0	1	1	0	1	2
bde	1	0	0	1	0	1
bef	2	0	0	0	0	2
cef	0	1	1	0	1	2

Hình 2. 11. Tìm tập mục phổ biến theo thuật toán song song Count Distribution

### 2. 2. 2. 1. 2. Thuật toán Data Distribution

Trong thuật toán Data Distribution, cơ sở dữ liệu D được phân hoạch thành {D<sub>1</sub>, D<sub>2</sub>, ..., D<sub>p</sub>} nên mỗi bộ xử lý làm việc với một tập dữ liệu không đầy đủ, vì vậy việc trao đổi dữ liệu giữa các bộ xử lý là rất cần thiết. Ngoài ra, các tập mục ứng viên

cũng được phân hoạch và phân bố cho tất cả các bộ xử lý, mỗi bộ xử lý làm việc với tập mục ứng viên  $C^i$  khác nhau.

**Nội dung của thuật toán:**

**Dữ liệu vào:** I, minsup,  $D_1, D_2, \dots, D_p$

**Dữ liệu ra:** L

**Phương pháp:**

$C^i = 1/p * |I|;$

**for** ( $k=1; C^i_k \neq \emptyset; k++$ ) **do begin**

// bước 1: tính các số đếm hỗ trợ cục bộ

count( $C^i_k, D_i$ ); // bộ xử lý thứ i

// bước 2: truyền các phân hoạch cơ sở dữ liệu cục bộ đến các

// bộ xử lý khác, nhận các phân hoạch cơ sở dữ liệu cục bộ từ

// các bộ xử lý truyền đến, quét  $D_j$  ( $l \leq i \leq p, j \neq i$ ) để tính số đếm

// hỗ trợ tổng thể

broadcast( $D_i$ );

**for** ( $j=1; (j \leq p \text{ and } j \neq i); j++$ ) **do begin**

receive( $D_j$ ); // từ bộ xử lý j

count( $C^i_k, D_j$ );

**end**

// bước 3: xác định các tập mục phổ biến  $L^i_k$  từ  $C^i_k$ ,

// trao đổi  $L^i_k$  với các bộ xử lý để thu được tập mục phổ biến  $L_k$ .

// sinh tập ứng viên  $C_{k+1}$

// phân hoạch  $C_{k+1}$  và phân bổ cho tất cả bộ xử lý

$L^i_k = \{c | c \in C^i_k \text{ and } c.\text{count} \geq \text{minsup} * |D_1 \cup D_2 \cup \dots \cup D_p|\};$

—  
=   
—

$C_{k+1} = \text{apriori\_gen}(L_k);$

$C^i_{k+1} = 1/p * |C_{k+1}|; // \text{phân hoạch lại các tập mục ứng viên}$

**end**

**return**  $L = L_1 \cup L_2 \cup \dots \cup L_k;$

Thuật toán Data Distribution thực hiện như sau:

Cơ sở dữ liệu D được phân hoạch  $\{D_1 \cup D_2 \cup \dots \cup D_p\}$  và phân bổ lần lượt cho các bộ xử lý  $P_i$  ( $1 \leq i \leq p$ ). Thuật toán có 3 bước cơ bản như sau:

- Bước 1: Mỗi bộ xử lý quét phân hoạch cơ sở dữ liệu cục bộ để tính các số đếm hỗ trợ cục bộ của các tập mục ứng viên được phân bổ cho nó.
- Bước 2: Mỗi bộ xử lý truyền phân hoạch cơ sở dữ liệu của nó đến các bộ xử lý khác và nhận các phân hoạch cơ sở dữ liệu từ các bộ xử lý khác truyền đến. Tiếp theo, quét các phân hoạch cơ sở dữ liệu nhận được để tính các số đếm hỗ trợ tổng thể của các tập mục ứng viên trong cơ sở dữ liệu D.
- Bước 3: Mỗi bộ xử lý sẽ xác định các tập mục phổ biến từ phân hoạch tập mục ứng viên của nó và trao đổi với các bộ xử lý khác để nhận được tất cả các tập mục phổ biến  $L_k$ , sau đó sinh tập mục ứng viên  $C_{k+1}$  từ  $L_k$ , phân hoạch  $C_{k+1}$  và phân bổ các phân hoạch ứng viên đó cho tất cả các bộ xử lý.

Thuật toán lặp lại bước 1  $\rightarrow$  3 cho đến lúc không còn tập mục ứng viên nào được sinh ra.

*Phương pháp phân bổ dữ liệu cục bộ của thuật toán:* Thuật toán thông báo đến  $(p - 1)$  hàm nhận không đồng bộ MPI để nhận dữ liệu từ các bộ xử lý. Nếu dữ liệu là tập mục được nhận từ bộ xử lý khác, trước hết bộ xử lý sẽ xử lý dữ liệu nhận được trước khi xử lý dữ liệu cục bộ của chính nó. Công việc này sẽ tránh tắc nghẽn đường truyền hoặc bộ đệm. Nếu không có dữ liệu nào được nhận, bộ xử lý sẽ đọc một tập mục từ dữ liệu cục bộ và cập nhật số đếm hỗ trợ cho tập mục ứng viên. Nếu tất cả dữ liệu đều được cập nhật tập mục ứng viên cục bộ, công việc tiếp theo sẽ tìm tập mục ứng viên cho xử lý kế tiếp.

Ví dụ: Ta có cơ sở dữ liệu giao dịch D như hình 2. 10, với độ hỗ trợ tối thiểu  $minsup = 33\%$ . Phân hoạch cơ sở dữ liệu giao dịch D cho 3 bộ xử lý  $P_0$ ,  $P_1$  và  $P_2$ , mỗi bộ xử lý có 2 giao dịch và k chỉ số vòng lặp để tìm các tập mục phổ biến. Thuật toán thực hiện theo thứ tự như sau:

TID	Item	Bộ xử lý
1	f d b e	P0
2	f e b	
3	a d b	P1
4	a e f c	
5	a d e	P2
6	a c f e	

(1) Phân hoạch cơ sở dữ liệu D

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
a	0	c	1	e	2
b	2	d	1	f	1

(2) Bước 1 với k = 1

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
a	4	c	2	e	5
b	3	d	3	f	4

(3) Bước 2 với k = 1

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
ab	0	bc	0	ce	1
ac	0	bd	1	cf	1
ad	0	be	0	de	1
ae	0	bf	0	df	0
af	0	cd	0	ef	1

(4) Bước 3 với k = 1 và bước 1 với k = 2

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
ab	1	be	0	ce	2
ac	2	bd	2	cf	2
ad	2	be	2	de	2
ae	3	bf	2	df	1
af	2	ed	0	ef	4

(5) Bước 2 với k = 2

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
ace	0	aef	1	bef	0
acf	0	bde	0	cef	1
ade	0				

(6) Bước 3 với k = 2 và bước 1 với k = 3

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
ace	2	aef	2	bef	2
acf	2	bde	1	cef	2
ade	1				

(7) Bước 2 với k = 3

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
acef	0	∅	∅	∅	∅

(8) Bước 3 với k = 3 và bước 1 với k = 4

Item	Số đêm	Item	Số đêm	Item	Số đêm
	P0		P1		P2
acef	2	∅	∅	∅	∅

(9) Bước 2 với k = 4 và bước 3 với k = 4

Trả về L, thuật toán kết thúc

Hình 2. 12. Tìm tập mục phổ biến theo thuật toán song song Data Distribution

### 2. 2. 2. 1. 3. Thuật toán song song Eclat

Thuật toán song song Eclat [18] dùng phương pháp nhóm các tập mục phổ biến có liên quan với nhau bằng cách sử dụng lược đồ phần chia lớp tương đương, với mỗi lớp tương đương chứa tập các tập mục ứng viên quan hệ tương đương với nhau. Phương pháp này sử dụng kỹ thuật tổ chức cơ sở dữ liệu theo chiều dọc để nhóm các giao dịch liên quan với nhau:

- Phân lớp tương đương: Gọi  $L_k$  là tập các itemset phổ biến, giả sử  $L_k$  được sắp xếp theo thứ tự từ điển. Có thể phân hoạch các tập mục trong  $L_k$  thành các lớp tương đương: Nếu các phần tử trong  $L_k$  có  $k-1$  thành viên đầu tiên giống nhau thì chúng thuộc cùng một lớp. Ký hiệu lớp tương đương chứa a là  $S_a = \{a\}$ . Trong phạm vi một lớp, ta sinh  $k$ -itemset ứng viên bằng cách kết nối tất cả  $\binom{|S_i|}{2} = |S_i|(|S_i| - 1) / 2$  cặp tiền tố là định danh của lớp. Trong đó  $|S_i|$  là số phần tử của lớp có định danh là i. Các  $k$ -itemset ứng viên được sinh ra từ các lớp khác nhau sẽ độc lập với nhau.
- Tổ chức cơ sở dữ liệu: Thuật toán Eclat sử dụng cách tổ chức dữ liệu theo

chiều dọc. Với cách tổ chức này, một cơ sở dữ liệu gồm danh sách các mục và mỗi mục xác định một danh sách các định danh của giao dịch có chứa mục đó, ký hiệu tid-List. Những ưu điểm của cách tổ chức dữ liệu theo chiều dọc như sau:

- Nếu tid-List đã được sắp xếp theo thứ tự tăng dần thì độ hỗ trợ của k-itemset ứng viên có thể đã được tính toán bởi phép lấy giao các tid-List của hai (k-1)-subset bất kỳ.
- Các danh sách của định danh của giao dịch tid-List chứa tất cả các thông tin liên quan về một tập mục. Vì vậy, khi tính độ hỗ trợ cho một tập mục không cần phải quét toàn bộ cơ sở dữ liệu.

		Theo chiều ngang							Theo chiều dọc				
		ITEMS							ITEMS				
		A	B	C	D	E			A	B	C	D	E
TIDS	T1	1	0	0	1	1	TIDS	T1	1	0	0	1	1
	T2	1	1	0	0	0		T2	1	1	0	0	0
	T3	0	0	1	1	1		T3	0	0	1	1	1
	T4	1	1	0	1	1		T4	1	1	0	1	1

Hình 2. 13. Tổ chức dữ liệu theo chiều ngang và theo chiều dọc

Trong hình vẽ trên thì tid-List của A là  $T(A) = \{1; 2; 4\}$  và tid-List của B là  $T(B) = \{2; 4\}$ . Lấy  $T(A) \cap T(B)$  sẽ cho  $T(AB)$ . Khi đó tid-list của AB là  $T(AB) = \{2; 4\}$ . Ta có thể tính được độ hỗ trợ bằng cách đếm số phần tử trong tid-List, nếu phần tử tid-List lớn hơn hoặc bằng độ hỗ trợ tối thiểu thì chèn AB vào  $L_2$ .

### Nội dung thuật toán song song Eclat

**begin**

*/\* Giai đoạn khởi tạo \*/*

- Duyệt qua các phân hoạch cơ sở dữ liệu cục bộ.
- Tính toán số đếm hỗ trợ cục bộ cho tất cả 2-itemset.
- Xây dựng số đếm hỗ trợ tổng thể cho các tập mục chứa  $L_2$ .

*/\* Giai đoạn biến đổi \*/*

- Phân hoạch  $L_2$  thành các lớp tương đương.
- Lập lịch  $L_2$  trên tập các bộ xử lý P.
- Tổ chức phân hoạch cơ sở dữ liệu cục bộ theo chiều dọc.
- Truyền các tid-List có liên quan tới các bộ xử lý khác.
- Nhận các tid-List từ các bộ xử lý khác =  $L_2$  cục bộ.

*/\* Giai đoạn đồng thời \*/*

- forparallel mỗi lớp tương đương  $E_2$  trong  $L_2$  cục bộ

    Compute\_Frequent( $E_2$ )

*/\* Giai đoạn rút gọn \*/*

- Tổng hợp các kết quả và đưa ra các luật kết hợp.

**end**

## Giải thích nội dung thuật toán

\* *Giai đoạn khởi tạo:* Bao gồm việc tính toán tất cả các 2-itemset phô biến trong cơ sở dữ liệu cần khai phá. Trong giai đoạn khởi tạo ta không cần tính số đếm hỗ trợ của các 1-itemset vì việc xác định số đếm hỗ trợ của 2-itemset có thể đạt được chỉ trong một lần duyệt cơ sở dữ liệu. Để tính toán cho các 2-itemset, trên mỗi bộ xử lý sử dụng một mảng cục bộ và tiến hành chỉ số hóa các mục trong cơ sở dữ liệu theo cả hai chiều. Mặt khác, mỗi bộ xử lý tính số đếm hỗ trợ cục bộ cho các 2-itemset và thực hiện phép lấy tổng rút gọn của tất cả các bộ xử lý để xây dựng các số đếm hỗ trợ tổng thể.

Khi kết thúc giai đoạn khởi tạo, tất cả các bộ xử lý đều có những số đếm hỗ trợ tổng thể của tất cả các 2-itemset phô biến  $L_2$  trong cơ sở dữ liệu.

\* *Giai đoạn biến đổi:* Thực hiện gồm các bước như sau:

- Bước 1: Đầu tiên  $L_2$  được phân hoạch thành các lớp tương đương, tiếp theo các lớp tương này sẽ được gán cho các bộ xử lý sao cho cân bằng nhau.
- Bước 2: Cơ sở dữ liệu đã được biến đổi định dạng theo chiều ngang thành chiều dọc và được phân phối lại. Vì vậy, trong bộ nhớ cục bộ của mỗi bộ xử lý, các tid-List của tất cả các 2-itemset trong một lớp tương đương bất kỳ được gán cho nó.

## Phương pháp lập lịch phân lớp tương đương

Trước tiên, ta phân hoạch  $L_2$  thành các lớp tương đương bằng cách sử dụng tiền tố chung. Tiếp theo, phân chia cho mỗi bộ xử lý một lớp tương đương và mỗi lớp tương đương được gán một trọng số dựa vào số các phần tử trong lớp. Vì phải khảo sát tất cả các cặp trong bước lặp tiếp theo cho nên ta gán trọng số  $\frac{1}{m}$  cho một lớp, với  $m$  là số các phần tử của lớp tương đương tương ứng.

Sắp xếp các lớp dựa theo các trọng số và lần lượt gán mỗi lớp cho mỗi bộ xử lý đã nạp ít nhất, nghĩa là bộ xử lý đó có trọng số toàn phần của các lớp nhỏ nhất.

Nếu ước lượng tốt được số các tập mục phô biến mà có thể nhận được từ một lớp tương đương thì có thể sử dụng ước lượng này làm một trọng số. Trong phạm vi một lớp, cũng có thể lấy độ hỗ trợ trung bình của các tập mục làm trọng số.

## Biến đổi cơ sở dữ liệu theo chiều dọc

Sau khi phân hoạch các lớp tương đương cân bằng giữa các bộ xử lý, ta biến đổi cơ sở dữ liệu cục bộ từ định dạng theo chiều ngang sang chiều dọc. Điều này có thể thực hiện được trong hai bước.

*Bước 1:* Mỗi bộ xử lý duyệt cơ sở dữ liệu cục bộ của nó và xây dựng các tid-List cục bộ cho tất cả các 2-itemset.

*Bước 2:* Mỗi bộ xử lý cần xây dựng các tid-List toàn cục cho các tập mục trong các lớp tương đương của nó. Vì vậy, nó phải gửi các tid-List này cho các bộ xử lý khác và nhận các tid-List từ các bộ xử lý khác gửi đến.

\* *Giai đoạn đồng thời:* Kết thúc giai đoạn biến đổi, các cơ sở dữ liệu đã được phân bố lại, do đó tất cả các tid-List của tất cả các 2-itemset trong các lớp tương đương cục bộ của nó là đã thường trú trên đĩa cục bộ. Mỗi bộ xử lý có thể tính toán tất cả các tập mục phổ biến một cách độc lập. Nó đọc trực tiếp từ bộ nhớ cục bộ các tid-List của các 2-itemset, sau đó sinh ra tất cả các tập mục phổ biến có thể trước khi chuyển đến lớp tiếp theo, bước này bao gồm việc quét phân hoạch cơ sở dữ liệu cục bộ đã được biến đổi một lần.

Trong phạm vi mỗi lớp tương đương cần khảo sát tất cả các cặp 2-itemset và thực hiện lấy giao các tid-List tương ứng. Nếu số các phần tử của tid-List lớn hơn hoặc bằng độ hỗ trợ tối thiểu thì tập mục mới được bổ sung vào  $L_3$  thành lập các lớp tương đương dựa trên các tiền tố chung độ dài bằng 2. Quá trình này được lặp cho đến khi không còn k-itemset phổ biến nào được tìm thấy.

Thủ tục được thực hiện như sau:

```

begin Compute_Frequent( $E_{k-1}$ )
  for tất cả các itemset  $I_1$  và  $I_2$  trong  $E_{k-1}$ 
    if  $((I_1.\text{tidList} \cap I_2.\text{tidList}) \geq \text{minsup})$  then
      Bổ sung  $(I_1 \cup I_2)$  là  $L_k$ ;
      Phân hoạch  $L_k$  thành các lớp tương đương;
      for parallel mỗi lớp tương đương  $E_k$  trong  $L_k$ 
        Compute_Frequent( $E_k$ );
    end
```

\* *Giai đoạn rút gọn:* Tại thời điểm cuối cùng của giai đoạn đồng thời, ta trích rút tất cả các kết quả từ mỗi bộ xử lý và đưa ra kết quả.

### Quá trình thực hiện các bước truyền thông khác nhau của thuật toán

\* *Giai đoạn khởi tạo:* Khi đã thu được các số đếm hỗ trợ cục bộ của tất cả các 2-itemset, ta cần thực hiện một phép lấy tổng rút gọn để tính số đếm tổng thể.

\* *Giai đoạn biến đổi:* Mỗi bộ xử lý quét phân hoạch cơ sở dữ liệu cục bộ của nó lần thứ hai và xây dựng các tid-List theo chiều dọc đối với tất cả các 2-itemset phổ biến  $L_2$ .

Quá trình biến đổi được hoàn thành qua hai bước như sau:

*Bước 1:* Biến đổi tid-List cục bộ

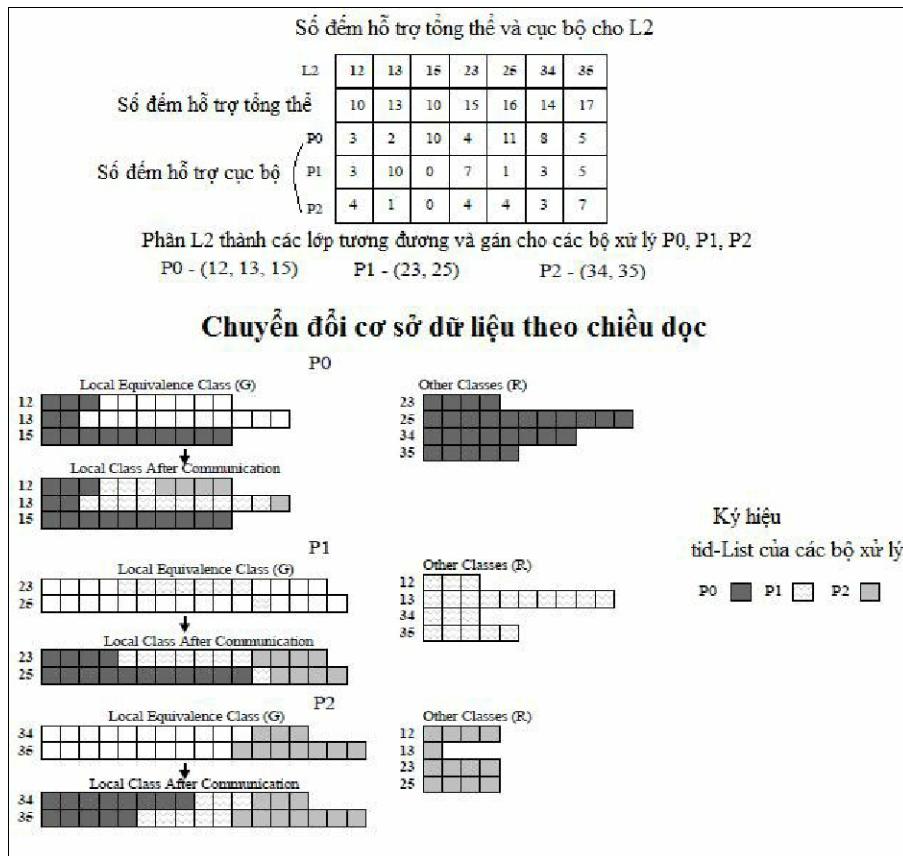
Trước tiên, chia L<sub>2</sub> thành hai nhóm. Các tập mục thuộc các lớp tương đương mà được gán cho bộ xử lý cục bộ, ký hiệu là G. Các tập mục còn lại thuộc các lớp tương đương khác, ký hiệu là R. Với mỗi bộ xử lý P<sub>i</sub>, bộ nhớ sẽ dành ra một vùng nhớ có kích thước:

$$\sum_{g \in G} global\_count(g) + \sum_{r \in R} partial\_count(r, P_i)$$

Với g ∈ G, r ∈ R: tập các mục

partial\_count(r, P<sub>i</sub>): Số đếm hỗ trợ cục bộ của tập mục r trên bộ xử lý P<sub>i</sub>.

Tiếp theo, mỗi bộ xử lý thực hiện việc biến đổi và ghi tid-List của các phần tử của G vào các khoảng trống thích hợp. Các phần tử của R được để trống.

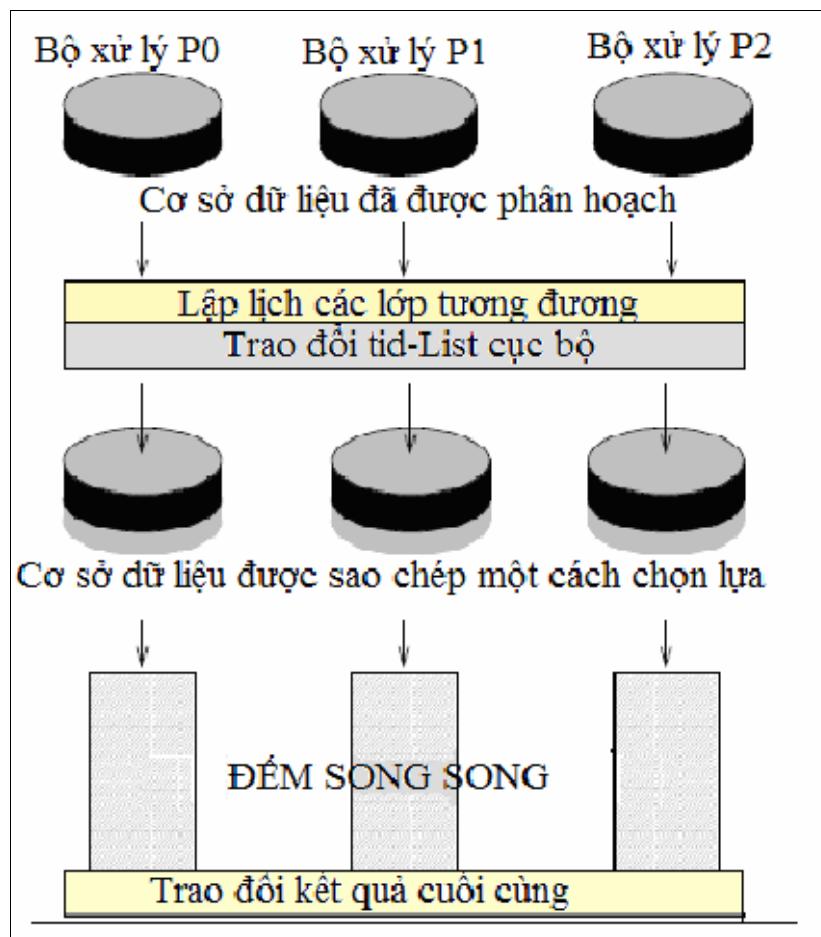


Hình 2. 14. Chuyển đổi dữ liệu

#### Bước 2: Truyền tid-List

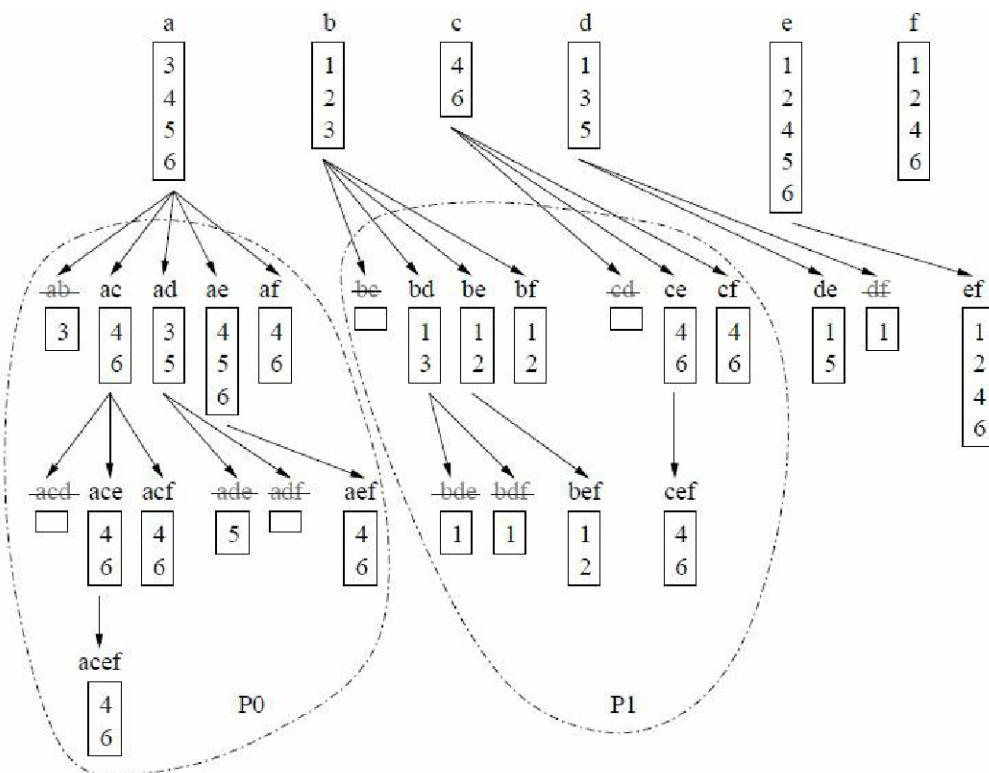
Khi việc biến đổi cơ sở dữ liệu cục bộ hình thành, chúng ta cần phải nhận các tid-List của tất cả các 2-itemset trong G từ các bộ xử lý khác truyền đến và truyền các tid-List của R đến các bộ xử lý khác. Các tid-List đến được sao chép vào các khoảng trống thích hợp. Do các vùng giao dịch là phân biệt và tăng đều, các tid-List cuối cùng của mỗi 2-itemset đã xuất hiện theo thứ tự từ điển. Các tid-List đã được viết ra ngoài

đĩa, còn trong R thì bị loại bỏ. Để truyền các tid-List cục bộ qua kênh bộ nhớ, chúng ta sử dụng lợi thế của việc truyền thông điệp nhanh ở mức người dùng. Mỗi bộ xử lý sẽ xác định kích thước bộ đệm cho một vùng truyền, một định dạng và dùng chung một định danh. Việc truyền thông được tiến hành theo cách khóa luân phiên các giai đoạn ghi và đọc. Trong giai đoạn ghi, mỗi bộ xử lý ghi các tid-List của các tập mục trong P vào cùng truyền của nó cho đến khi đạt đến giới hạn không gian bộ đệm. Tại thời điểm này, nó đi và giai đoạn đọc, lần lượt quét vùng nhận của mỗi bộ xử lý và đặt các tid-List của G vào các khoảng trống thích hợp. Lúc vùng đọc đã được quét xong, nó đi vào giai đoạn ghi. Quá trình này được lặp lại cho đến khi nhận được tất cả các tid-List bộ phận. Tại thời điểm cuối của giai đoạn này, cơ sở dữ liệu được định dạng theo chiều dọc.



Hình 2. 15. Thuật toán song song Eclat

Ví dụ: Ta có cơ sở dữ liệu giao dịch D như hình 2. 10, với độ hỗ trợ tối thiểu  $minsup = 33\%$ . Thuật toán song song Eclat thực hiện khai phá tập mục phổ biến như sau:



Hình 2. 16. Khai phá tập mục phổ biến sử dụng thuật toán song song Eclat

Trong ví dụ trên, thuật toán song song Eclat khai phá tập mục phổ biến của mỗi lớp bằng cách lấy giao các danh sách định danh giao dịch (tid-List). Tập mục phổ biến 2-itemset được chia thành 5 lớp tương đương và được gán cho 2 bộ xử lý. Bộ xử lý  $P_0$  khai phá lớp tương đương  $\{ac, ad, ae, af\}$ , bộ xử lý  $P_1$  khai phá hai lớp tương đương  $\{bd, be, bf\}$  và  $\{ce, cf\}$ . Các lớp  $\{de\}$  và  $\{ef\}$  không cần khai phá thêm nữa. Hai bộ xử lý  $P_0$  và  $P_1$  được xử lý song song, không có sự phụ thuộc dữ liệu giữa hai bộ xử lý này. Ví dụ: để khai phá tập mục “ace” thì bộ xử lý  $P_0$  cần kết hợp hai tập mục “ac” và “ae” bằng cách lấy giao “46” và “456” để nhận được kết quả tid-List “46”. Cùng lúc đó, bộ xử lý  $P_1$  có thể khai phá độc lập tập mục “bef” từ “be” và “bf”.

Không như các giải thuật song song dựa trên thuật toán Apriori là phải quét cơ sở dữ liệu nhiều lần (số lần phụ thuộc vào độ dài của tập mục phổ biến lớn nhất). Các giải thuật song song dựa trên giải thuật Eclat quét cơ sở dữ liệu ít hơn, giảm thiểu chi phí vào ra. Hơn nữa, sự độc lập giữa các bộ xử lý làm giảm chi phí truyền thông và đồng bộ. Để có cơ chế song song tốt thì số bộ xử lý nên nhỏ hơn số lớp tương đương. Mặt khác, chúng ta cũng cần sử dụng các chiến lược hiệu quả để đảm bảo cân bằng tải.

#### 2. 2. 2. 1. 4. Thuật toán song song FP-Growth

Thuật toán FP-Growth dựa vào thuật toán FP-Tree tuần tự [10] – [22], thuật toán xây dựng một số FP-Tree cục bộ trong môi trường bộ nhớ phân tán và sử dụng mô hình “Chủ-Tớ”. Thuật toán dựa vào 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 thuật toán.

Khai phá tập mục song song của thuật toán có hai nhiệm vụ chính sau đây:

#### - **Xây dựng song song FP-Tree:**

Trong giai đoạn đầu của thuật toán xây dựng các FP-Tree đồng thời trên mỗi bộ xử lý. Chúng ta chia cơ sở dữ liệu giao dịch D cho P bộ xử lý và chắc chắn mỗi bộ xử lý có N/P giao dịch ( $D_{NP}$ ), với N và P lần lượt là tổng số giao dịch trong cơ sở dữ liệu D và số các bộ xử lý. Công việc phân hoạch cơ sở dữ liệu D cho P xử lý được thực hiện ngẫu nhiên. Khi kết thúc việc phân hoạch dữ liệu, công việc tiếp theo là xác định 1-itemset phổ biến ( $F_{1\text{-itemset}}$ ) trước khi xây dựng FP-Tree cục bộ. Mỗi bộ xử lý tính toán số đếm hỗ trợ ( $f_{local}(i)$ ) của mỗi mục i bằng cách quét phân hoạch cơ sở dữ liệu cục bộ  $D_{NP}$ , tất cả các bộ xử lý chuyển số đếm hỗ trợ  $f_{local}(i)$  cục bộ đến bộ xử lý “Chủ”. Bộ xử lý “Chủ” tập hợp các mục và kết hợp chúng lại để sinh số đếm hỗ trợ tổng thể ( $f_{global}(i)$ ). Sau đó, các mục có độ hỗ trợ nhỏ hơn ngưỡng hỗ trợ tối thiểu  $minsup$  được lược bỏ. Tập các 1-itemset phổ biến thu được sẽ được truyền cho tất cả các bộ xử lý trong nhóm truyền thông.

Giai đoạn tiếp theo là xây dựng các FP-Tree cục bộ, mỗi bộ xử lý quét cơ sở dữ liệu cục bộ  $D_{NP}$  của nó, sau đó chèn các tập mục phổ biến vào trong cây FP-Tree. Công việc xây dựng FP-Tree bởi mỗi bộ xử lý với cơ sở dữ liệu cục bộ của nó giống như trong thuật toán tuần tự [13].

#### **Khai phá song song và sinh tập mục phổ biến**

Trong giai đoạn đầu, chúng ta xét toàn bộ FP-Tree và tạo ra các mẫu điều kiện cơ sở.

Giai đoạn tiếp theo, chúng ta tập hợp các mẫu điều kiện cơ sở từ các bộ xử lý để xây dựng FP-Tree điều kiện cơ sở (CFPT) cho mỗi mục phổ biến.

Giai đoạn cuối cùng, thực thi việc khai phá bằng cách xây dựng đệ qui các mẫu điều kiện cơ sở và các CFPTs cho đến khi sinh tất cả cá tập mục phổ biến.

*Xây dựng các mẫu điều kiện cơ sở:* Mỗi bộ xử lý sẽ thăm bảng header (1-itemset phổ biến cục bộ) của nó theo hướng từ trên xuống và tạo ra các mẫu điều kiện cơ sở cho mỗi mục phổ biến. Công việc thiết lập các mẫu điều kiện cơ sở bằng cách toàn bộ các nút trong FP-Tree cục bộ từ trên xuống như trong thuật toán tuần tự [13].

*Xây dựng FP-Tree điều kiện cơ sở:* Khi tìm được các mẫu điều kiện cơ sở, các FP-Tree điều kiện được xây dựng bằng cách hợp nhất các mẫu này. Phương pháp hợp nhất được trình bày trong [10]–[13]. Với mỗi mục phổ biến, các mẫu điều kiện cơ sở được hợp nhất sao cho các số đếm hỗ trợ của các mục giống nhau được tăng lên để tính được số đếm hỗ trợ tổng thể. Nếu số đếm hỗ trợ tổng thể của một mục nhỏ hơn ngưỡng hỗ trợ tối thiểu thì mục đó sẽ được lược bỏ khỏi FP-Tree điều kiện.

Khi thực hiện sinh các FP-Tree điều kiện, chúng ta sử dụng mô hình “Chủ-Tớ”. Khi bộ xử lý Chủ chuyển các mục cần khai báo cho bộ xử lý Tớ, các bộ xử lý Tớ sinh các FP-Tree điều kiện cho các mục đó. Khi các bộ xử lý Tớ hoàn thành việc sinh FP-Tree điều kiện, nó sẽ chuyển một mã thông báo đến bộ xử lý Chủ yêu cầu chuyển mục kế tiếp. Nhiệm vụ của bộ xử lý Chủ là thực hiện yêu cầu của bộ xử lý Tớ và nó trả lời bằng cách chuyển mục kế tiếp đến bộ xử lý Tớ đó. Khi bộ xử lý Tớ nhận mục kế tiếp từ bộ xử lý Chủ chuyển đến nó sẽ bắt đầu sinh CFPT cho mục này. Chi phí cho việc truyền thông trong thuật toán này tương đối thấp bởi vì mỗi bộ xử lý chỉ chuyển một mã thông báo. Ngoài ra, khối lượng công việc là công bằng giữa các bộ xử lý trong nhóm bởi vì mỗi khi một bộ xử lý Tớ nào đó hoàn thành công việc nó sẽ nhận trực tiếp một công việc khác.

#### *Sinh các tập mục phô biến:*

Bằng cách xây dựng lần lượt các mẫu điều kiện cơ sở và các cây điều kiện FP-Tree bởi mỗi bộ xử lý. Khi một nhánh của FP-Tree điều kiện được xây dựng, chúng ta thu được tập hợp các mục khả năng như trong thuật toán FP-Growth tuần tự [13]. Mô hình song song được áp dụng là mô hình “Chủ-Tớ”. Khi bộ xử lý Chủ chuyển các mục cơ sở cần khai phá đến các bộ xử lý Tớ, các bộ xử lý Tớ thực hiện nhiệm vụ khai phá cho các mục này và sinh các tập mục phô biến.

Với mô hình này, khi một bộ xử lý Tớ hoàn thành nhiệm vụ, ngay lập tức nó nhận được nhiệm vụ, điều này làm cho các bộ xử lý bận cho đến khi kết thúc quá trình khai phá. Việc cân đối khối lượng công việc xảy ra trong thời gian thực thi, mô hình “Chủ-Tớ” được duy trì cho đến khi tất cả các tập mục phô biến được sinh ra ứng với mỗi mục phô biến trong  $F_{1\text{-itemset}}$ . Sau đó, tất cả các bộ xử lý Tớ chuyển các tập mục phô biến mà nó sinh ra đến bộ xử lý Chủ và giai đoạn khai phá kết thúc. Với mỗi mục  $i \in F_{1\text{-itemset}}$ , các tập mục phô biến được sinh đệ qui bởi các mẫu điều kiện cơ sở và các FP-Tree điều kiện.

#### **Thuật toán FP-Growth [10]**

**Dữ liệu vào:** Các phân hoạch cơ sở dữ liệu  $D_{NP}$  cho mỗi bộ xử lý và  $\text{minsup}$ .

**Dữ liệu ra:** Tập các mục phô biến.

**Phương pháp:**

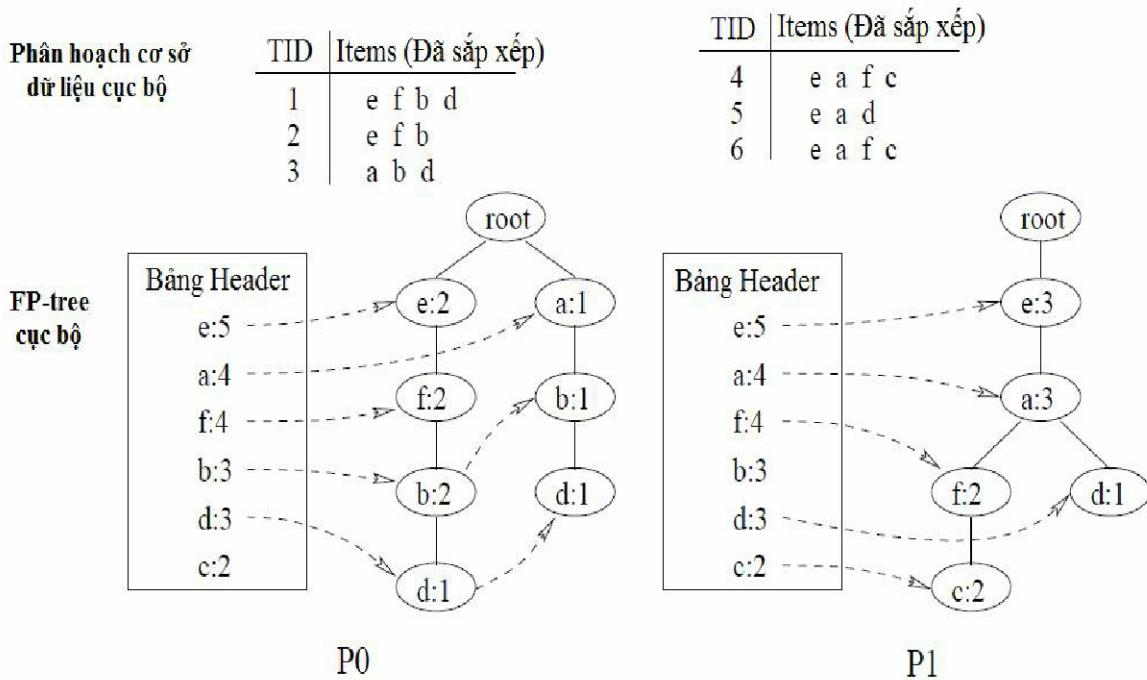
- 1) Quét cơ sở dữ liệu cục bộ  $D_{NP}$ ;
- 2) Tính số đếm hỗ trợ cục bộ  $f_{local}(i)$  của mỗi mục  $i$ ;
- 3) if (Bộ xử lý Chủ) then
  - 4) for (mỗi Bộ xử lý Tớ) do
  - 5) Nhận  $f_{local}(i)$ ;
  - 6) Gán  $F_{1\text{-itemset}} = \{i | \sum f_{local}(i) \geq \text{minsup}, \forall i\}$  và truyền  $F_{1\text{-itemset}}$  đến các bộ xử lý Tớ;
  - 7) else Chuyển  $f_{local}(i)$ ,  $\forall i$  đến Bộ xử lý Chủ và nhận 1-itemset phô biến  $F_{1\text{-itemset}}$ ;

- 8) Xây dựng FP-Tree cục bộ  $FPT_{local}$  của các mục trong  $F_{1-itemset}$  bằng cách quét  $D_{NP}$  cục bộ;
- 9) Duyệt toàn bộ  $FPT_{local}$  và sinh ra các mẫu điều kiện cơ sở và truyền đến tất cả các Bộ xử lý;
- 10) if (Bộ xử lý Chủ) then begin
  - 11) for (mỗi mục phô biến  $i \in F_{1-itemset}$ ) do // *Lập lịch tạo ra các CFPTs*  
Nhận yêu cầu bộ xử lý Tớ và chuyển mục  $i$ ;
  - 12) for (mỗi mục phô biến  $i \in F_{1-itemset}$ ) do // *Lập lịch cho việc khai phá*  
Nhận yêu cầu bộ xử lý Tớ và chuyển mục  $i$  cần khai phá;
  - 13) for (mỗi Bộ xử lý Tớ) do  
Tập hợp tập mục phô biến và xuất hết các tập mục phô biến;
  - 14) else do // *hợp nhất các mẫu điều kiện cơ sở*
  - 15) Yêu cầu mục  $i$  tiếp theo và sinh FP-Tree điều kiện  $CFPT_i$  cho mục  $i$ ;
  - 16) until (hết các mục phô biến);
  - 17) Truyền CFPTs đến tất cả bộ xử lý (trừ Bộ xử lý Chủ) và Nhận CFPTs;
  - 18) do  
Yêu cầu mục  $i$  tiếp theo và gọi FP-Growth-OneItem ( $CFPTs, null, i$ );
  - 19) until (hết các mục phô biến);
  - 20) Chuyển các tập mục phô biến đến Bộ xử lý Chủ;

### Thủ tục con FP-Growth-OneItem (Tree, $\alpha$ , $i$ )

#### Phương pháp:

- 1) if (Tree chứa đường dẫn đơn) and ( $i \neq null$ ) then
  - 2) Sinh tập mục có độ hỗ trợ  $\geq minsup$  đối với mỗi tổ hợp các nút trong đường dẫn
  - 3) else if ( $i \neq null$ ) then
    - 4) Sinh tập mục  $\beta = i \cup \alpha$  và Xây dựng các mẫu điều kiện cơ sở của  $\beta$  và  $CFPT_\beta$
    - 5) else for mỗi  $i$  trong bảng header của Cây
      - 6) Sinh tập mục  $\beta = i \cup \alpha$  và Xây dựng các mẫu điều kiện cơ sở của  $\beta$  và  $CFPT_\beta$
      - 7) if  $CFPT_\beta \neq \emptyset$  then FP-Growth-OneItem ( $CFPT_\beta, \beta, null$ );



Hình 2. 17. Cấu trúc FP-tree cục bộ được xây dựng từ các phân hoạch cơ sở dữ liệu

Cấu trúc của FP-tree cục bộ được xây dựng từ các phân hoạch cơ sở dữ liệu cục bộ cho 2 bộ xử lý. Trong các giao dịch và bảng header thì các items được sắp xếp theo tần suất giảm dần. Hình 2. 17 chỉ ra cấu trúc song song của các FP-tree cục bộ trên hai bộ xử lý của ví dụ đã đưa ra trong hình 2. 10.

Bộ xử lý	P0				P1	
Item	e:5	a:4	f:4	b:3	d:3	c:2
Mẫu điều kiện cơ sở		(e:3)	(e:2) (e:2, a:2)	(e:2, f:2) (a:1)	(e:1, f:1, b:1) (a:1, b:1) (e:1, a:1)	(e:2, a:2, f:2)
FP-tree điều kiện						
Tập mục phổ biến được khai phá	e:5	a:4 ea:3	f:4 ef:4 af:2 eaf:2	b:3 eb:2 fb:2 efb:2	d:3 ed:2 ad:2 bd:2	c:2 ac:2 ec:2 fc:2 afc:2 efc:2 aec:2 aefc:2

Hình 2. 18. Khai phá tập mục phổ biến sử dụng thuật toán song song FP-Growth

Hình 2. 18 Khai phá song song tập mục phổ biến từ mẫu điều kiện cơ sở và FP-tree điều kiện. Trong hình vè cho ta thấy các mẫu điều kiện cơ sở và FP-tree điều kiện của tất cả các tập mục, được gán cho hai bộ xử lý. Bộ xử lý P<sub>0</sub> tính FP-tree điều kiện cho tập mục a, f và b. Bộ xử lý P<sub>1</sub> tính FP-tree điều kiện cho tập mục d, c. Tập mục e

có mẫu điều kiện cơ sở rỗng, nên không cần khai phá thêm. Các tập mục phô biến được suy ra từ FP-tree điều kiện.

### 2. 2. 2. 2. Thuật toán sinh luật song song

Bao gồm việc phân hoạch tập tất cả các tập mục phô biến cho tất cả các bộ xử lý. Mỗi bộ xử lý sinh các luật dựa và phân hoạch của nó bằng cách sử dụng thuật toán sinh luật kết hợp. Vì số các luật được sinh ra từ một tập mục bị ảnh hưởng bởi kích thước tập mục, vì vậy khi phân hoạch tập tất cả các tập mục phô biến  $L = \{L_1, L_2, \dots, L_k\}$  cho  $P$  bộ xử lý thì một bộ xử lý  $i$  có một phân hoạch  $L^i = \{L_1^i, L_2^i, \dots, L_k^i\}$  (với  $1 \leq i \leq P$ ) sao cho kích thước của các  $L_1^i, L_2^i, \dots, L_k^i$  trên các bộ xử lý phải gần bằng nhau. Để tính độ tin cậy của luật, mỗi bộ xử lý cần phải kiểm tra độ hỗ trợ của một tập mục, do đó mỗi bộ xử lý cần phải xét tất cả các tập mục phô biến trước khi bắt đầu sinh luật.

### 2. 2. 2. 3. Đánh giá việc thực hiện của các thuật toán song song

#### 2. 2. 2. 3. 1. Cách đánh giá thuật toán song song

Đánh giá thuật toán tuần tự có thể căn cứ chủ yếu vào thời gian thực hiện tính theo hàm kích cỡ dữ liệu vào (input). Hàm này được gọi là độ phức tạp tính toán thời gian  $f(n)$  của thuật toán và được ký hiệu là  $O(f(n))$ . Mọi cách hình thức,  $O()$  được định nghĩa như sau:

Một thuật toán có độ phức tạp tính toán  $f(x) = O(g(x)) \Leftrightarrow$  Tồn tại số  $C$  dương và số nguyên  $x_0$  sao cho  $0 \leq f(x) \leq C * g(x)$ , với mọi số lượng dữ liệu vào  $x \geq x_0$ ,  $O(1)$  ký hiệu cho một hằng số bất kỳ.

Ngoài ra, độ phức tạp tính toán của thuật toán song song còn phụ thuộc vào kiến trúc máy tính song song và số lượng bộ xử lý được phép sử dụng trong hệ thống và do vậy phụ thuộc và thời gian trao đổi dữ liệu giữa các bộ xử lý.

Độ phức tạp thời gian là thước đo quan trọng nhất đánh giá mức độ hiệu quả của thuật toán song song. Giả sử rằng mô hình tính toán của chúng ta có  $p$  bộ xử lý; dẫn đến mức độ song song có giới hạn; ngược lại, không bị giới hạn khi số lượng bộ xử lý không bị chặn.

Mức độ hiệu quả của thuật toán được thể hiện ở mức độ song song của thuật toán là số lượng cực đại các phép toán độc lập có thể thực hiện đồng thời ở mỗi thời điểm thực hiện của thuật toán.

Ký hiệu  $p(w)$  là độ song song của thuật toán, thì thuật toán đạt hiệu quả để giải toán có kích cỡ  $w$  là thuật toán chỉ cần sử dụng nhiều nhất  $p(w)$  bộ xử lý.

Độ phức tạp thời gian của thuật toán song song sử dụng  $p$  bộ xử lý để giải một bài toán có kích cỡ  $n$  là hàm  $f(n, p)$  xác định thời gian cực đại trôi qua giữa thời điểm bắt đầu thực hiện thuật toán bởi một bộ xử lý và thời điểm kết thúc của các bộ xử lý đối với bộ dữ liệu vào bất kỳ.

Có hai loại thao tác khác nhau trong các thuật toán song song:

Các phép toán cơ sở như: +, -, \*, /, AND, OR, ...

Các phép toán truyền dữ liệu trên các kênh truyền.

Vì độ phức tạp thời gian của thuật toán song song được xác định bởi số các phép toán cơ sở và số các bước truyền tải dữ liệu giữa các bộ xử lý với nhau. Nên từ đó suy ra, độ phức tạp thời gian của thuật toán song song không chỉ phụ thuộc vào mô hình tính toán mà còn phụ thuộc vào số bộ xử lý được sử dụng.

Có ba cách định nghĩa [1] liên quan đến độ phức tạp của giải thuật song song là:

**Định nghĩa 1:** Một thuật toán song song có độ phức tạp tính toán  $O(t)$  với  $p$  bộ xử lý khi nó thực hiện nhiều nhất là  $O(t * p)$  phép toán cơ sở (Định lý Brent).

**Định nghĩa 2:** Một thuật toán song song có độ phức tạp tính toán  $O(t)$  sử dụng nhiều bộ xử lý để thực hiện  $O(e)$  phép toán cơ sở khi cài đặt với  $p$  bộ xử lý thì sẽ có độ phức tạp thời gian là  $O([e/p] + t)$ .

**Định nghĩa 3:** Một thuật toán song song có độ phức tạp tính toán  $O(t)$  với  $p$  bộ xử lý có thể cài đặt với  $[p/f]$  bộ xử lý ( $1 \leq f \leq p$ ) thì sẽ có độ phức tạp thời gian là  $O(f*t)$ .

Định nghĩa 2 chỉ ra rằng khi số bộ xử lý được sử dụng giảm xuống một lượng nhất định thì thuật toán vẫn tiếp tục làm việc nhưng thời gian thực hiện sẽ tăng lên.

Định nghĩa 3 khẳng định rằng có cách để cài đặt thuật toán song song khi số lượng bộ xử lý được sử dụng bị giảm xuống.

Ngoài ra, trong đánh giá thuật toán song song chúng ta còn cần phải xét tới độ tăng tốc và hiệu suất của nó.

### Chi phí và chi phí tối ưu của giải thuật

Chi phí tính toán song song có thể định nghĩa:

$$\text{Chi phí} = (\text{thời gian thực thi}) * (\text{tổng số các bộ xử lý được sử dụng})$$

Chi phí tính toán tuần tự đơn giản là thời gian xử lý của nó, ts. Chi phí tính toán song song là  $tp * p$ . Thuật toán song song tối ưu về chi phí là một trong những thuật toán mà chi phí giải quyết bài toán tương đương với thời gian thực hiện trên một hệ thống bộ xử lý đơn.

$$\text{Chi phí} = tp * p = k * ts$$

Ở đây  $k$  là hằng số,  $p$  là số lượng bộ xử lý được sử dụng. Phân tích độ phức tạp thời gian, chúng ta có thể nói rằng thuật toán song song là tối ưu về chi phí nếu:

$$\text{Độ phức tạp thời gian song song} * \text{số bộ xử lý} = \text{độ phức tạp thời gian tuần tự}$$

Nói chung, các ứng dụng song song phức tạp hơn rất nhiều so với các ứng dụng

tuần tự tương ứng. Không chỉ có nhiều luồng chỉ thị thực thi tại cùng một thời điểm mà còn có nhiều luồng dữ liệu giữa chúng.

### **2. 2. 2. 3. 2. So sánh việc thực hiện của các thuật toán**

Để đánh giá, so sánh việc thực hiện của thuật toán ta phải dựa vào việc thực thi của thuật toán trên cơ sở dữ liệu khác nhau. Ta nhận thấy rằng thuật toán FP-Growth thực hiện nhanh nhất, tiếp đến là thuật toán Eclat, rồi đến thuật toán Count Distribution, thuật toán Data Distribution. Vấn đề xếp thứ hạng này chỉ mang tính chất tương đối vì mỗi thuật toán đều có những ưu điểm và nhược điểm riêng của nó.

Các thuật toán song song như thuật toán Count Distribution, Data Distribution được cài đặt dựa trên cơ sở của thuật toán Apriori được sử dụng phổ biến. Trong số các thuật toán khai phá các luật kết hợp song song thì các luật kết hợp song song có thể được sinh ra trực tiếp dựa vào cách thức khai phá tập mục, vì khi các tập mục ứng viên được sinh ra thì tất cả thông tin của tập con đều được tính toán. Tốc độ thực hiện của thuật toán nói trên tỷ lệ với số lượng các giao dịch nhưng có thể gặp khó khăn trong việc xử lý quá nhiều mục hoặc nhiều mẫu khi cơ sở dữ liệu quá lớn. Chẳng hạn, với thuật Count Distribution nếu số các tập mục ứng viên hoặc số các tập mục phổ biến tăng lên vượt quá giới hạn lưu giữ trong bộ nhớ chính của mỗi bộ xử lý thì thuật toán có thể hoạt động không tốt cho dù có bổ sung thêm nhiều bộ xử lý.

Thời gian thực thi của các thuật toán có thể bị kéo dài ra do thủ tục tính toán tập mục chậm và do tùy thuộc vào số lượng các giao dịch mà thuật toán tìm kiếm lặp lại nhiều lần vô số tập mục.

## **2. 3. Kết luận chương 2**

Khai phá luật kết hợp trong cơ sở dữ liệu có thể chia thành hai bài toán con: tìm tất cả các tập mục phổ biến từ cơ sở dữ liệu và sinh ra các luật từ các tập mục phổ biến. Nội dung của chương đã trình bày một cách tổng quan về luật kết hợp, các định nghĩa, tính chất liên quan đến luật kết hợp như độ hỗ trợ, độ tin cậy, tập mục phổ biến và phát biểu bài toán khai phá luật kết hợp. Tiếp theo, nội dung chương trình bày một số thuật toán cơ bản để phát hiện tập mục phổ biến và phát hiện luật kết hợp từ các tập mục phổ biến đó làm cơ sở cho các nghiên cứu như cải tiến thuật toán, thuật toán song song, ... Cuối cùng nội dung của chương trình bày một số thuật toán khai phá luật kết hợp song song. Đây chính là cơ sở lý thuyết để từ đó chúng ta đi sâu tìm hiểu, cài đặt thử nghiệm một số thuật toán song song trong khai phá dữ liệu đó trong chương 3.

### CHƯƠNG 3

## CÀI ĐẶT THUẬT TOÁN KHAI PHÁ CÁC LUẬT KẾT HỢP SONG SONG TRONG KHAI PHÁ DỮ LIỆU

### 3. 1. Cài đặt thuật toán khai phá các luật kết hợp song song

Các thuật toán khai phá luật kết hợp song song đã được trình bày chi tiết trong chương 2 gồm các thuật toán Count Distribution, Data Distribution, Eclat, FP-Growth, ... Ngoài ra, còn có rất nhiều thuật toán khai phá các luật kết hợp song song khác như Candidate Distribution, Bitmaps, ... đã được nghiên cứu và triển khai.

Để tiến hành cài đặt thuật toán ứng dụng cho bài toán khai phá dữ liệu, trong phần này tôi tiến hành cài đặt thử nghiệm hai thuật toán song song đã trình bày đó là Count Distribution và Eclat. Ngoài ra, tôi có cài đặt thêm thuật toán Apriori như là một cách kiểm chứng giúp ta đánh giá được hiệu quả của các thuật toán song song.

#### 3. 1. 1. Môi trường cài đặt chương trình thử nghiệm

Cài đặt và sử dụng MPI.NET [9], một thư viện .NET cho phép tạo ra các ứng dụng song song hiệu suất cao có thể được triển khai trên các máy trạm đa luồng và theo cụm. MPI.NET cung cấp cách thức truy cập trong C# và tất cả các ngôn ngữ .NET khác thông qua truyền thông điệp MPI (Message Passing Interface). MPI là một chuẩn cho các chương trình truyền thông điệp với nhau một cách độc lập, có khả năng thực thi các chương trình song song trên các cụm máy tính và trên các siêu máy tính. Phần lớn các chương trình MPI viết cho mô hình song song đơn chương trình, đa luồng dữ liệu (Single Program, Multiple Data (SPMD)). MPI hỗ trợ các mô hình SPMD bằng cách cho phép người dùng dễ dàng khởi động cùng chương trình trên nhiều máy khác nhau (các nút) với một lệnh. Khi tạo ban đầu thì mỗi bộ xử lý có một hạng (rank) duy nhất (hạng là kiểu integer và được gán từ 0 đến  $P - 1$ , với  $P$  là số bộ xử lý). Các bộ xử lý trao đổi với nhau thông qua truyền thông điệp nhờ hạng của chúng.

\* **Cài đặt:** Phát triển các chương trình song song bằng cách sử dụng MPI.NET, ta sẽ cần một số công cụ khác. Ta không cần phải có một cluster Windows hoặc thậm chí một máy trạm đa lõi/đa bộ xử lý để phát triển các chương trình song song: bất kỳ máy tính để bàn có thể chạy Windows XP có thể được dùng để phát triển các chương trình MPI.NET.

- Microsoft Visual Studio 2005
- Microsoft's Message Passing Interface (MS-MPI): Thực tế, có nhiều cách khác nhau để có được MS-MPI: Microsoft HPC SDK hoặc Microsoft Compute Cluster Pack SDK. Và Microsoft HPC Server 2008 hoặc Microsoft Compute Cluster Server.
- Bộ cài Windows

**Cài đặt MPI.NET trên một cụm (Cluster):** Để xây dựng chương trình MPI.NET cần cài MPI.NET software development kit (MPI.NET SDK) trên các máy trạm.

\* **Cấu hình phần cứng máy tính cài đặt chương trình:** Intel (R) core (TM) Duo P8700 2.53GHz. Bộ nhớ RAM 3.0 GB, ổ đĩa cứng dung lượng 320 GB.

### 3. 1. 2. Mô tả dữ liệu của bài toán

Tại các Ngân hàng hiện nay nhu cầu sử dụng thẻ của khách hàng trong các giao dịch là rất lớn và đem lại rất nhiều tiện lợi cho người dùng trong việc thanh toán trong nước cũng như ngoài nước. Thẻ được chấp nhận như một phương thức thanh toán không dùng tiền mặt tại các điểm thanh toán có chấp nhận thẻ. Theo số liệu của Ngân hàng Nhà nước, tỷ trọng thanh toán bằng thẻ hiện chiếm tỷ trọng ngày càng lớn trong tổng số món giao dịch của các phương tiện thanh toán không dùng tiền mặt. Tốc độ tăng trưởng bình quân của lượng thẻ phát hành ra lưu thông những năm gần đây khoảng 150-300%/năm. Hiện cả nước có khoảng 9.000 ATM, hơn 28.000 điểm chấp nhận thẻ và hơn 17 triệu thẻ đang lưu hành với 176 thương hiệu thẻ do 41 tổ chức cung ứng dịch vụ thanh toán phát hành [23].

Theo chức năng thì thẻ thường chia làm 02 loại thẻ chính: Thẻ Debit (thẻ ghi nợ) và thẻ Credit (thẻ tín dụng).

- Thẻ ghi nợ là thẻ mà chủ thẻ sẽ đóng tiền vào để chi tiêu hoặc rút tiền trên số dư tài khoản tiền gửi của chính khách hàng, một số ngân hàng cho phép dùng tài khoản thẻ ghi nợ nhưng cho phép thâu chi nếu tài khoản thẻ đó là tài khoản thẻ trả lương. Với thẻ ghi nợ, khách hàng không những rút tiền mặt hoặc thanh toán tại các điểm chấp nhận thẻ của Ngân hàng mà còn có thể tận hưởng tiện nghi mua sắm tại bất kỳ điểm chấp nhận thẻ nào trên thế giới và qua Internet. Ngoài ra, thẻ ghi nợ còn nhiều tiện ích khác: Tiền gửi trong thẻ vẫn được hưởng lãi, Hiệu quả trong quản lý và kiểm soát chi tiêu cá nhân, Không còn sợ rủi ro như khi mang theo nhiều tiền mặt, Chuyển khoản an toàn và nhanh chóng, ...
- Thẻ tín dụng (thẻ ghi có) là loại thẻ chi tiêu trước trả tiền sau. Nghĩa là chủ thẻ sẽ vay tiền của ngân hàng để thanh toán hoặc rút tiền mặt trong hạn mức tín dụng nhất định được cấp bởi ngân hàng. Hạn mức tín dụng là số tiền tối đa chủ thẻ được chi tiêu trong một khoảng thời gian nào đó. Với loại thẻ này chủ thẻ sẽ phải chịu tiền lãi trên số tiền mà họ đã chi tiêu. Với thẻ ghi có tùy thuộc mức thu nhập và tình hình tài chính của khách hàng ngân hàng sẽ cấp cho khách hàng hạn mức tín dụng (1 tháng, 45 ngày hay hơn). Khách hàng có thể rút số tiền được ngân hàng cấp đó trong thời hạn nhất định và buộc phải thanh toán khi đáo hạn. Nếu quá hạn mức tín dụng mà chưa thanh toán kịp ngân hàng sẽ tính lãi suất cao.

Ở Việt Nam thẻ thông dụng và phổ biến nhất là thẻ ATM (Automatic Teller Machine) – Loại thẻ rút tiền từ Máy rút tiền tự động hay máy giao dịch tự động, là một thiết bị ngân hàng giao dịch tự động với khách hàng, thực hiện việc nhận dạng khách hàng thông qua thẻ ATM (thẻ ghi nợ, thẻ tín dụng) hay các thiết bị tương thích, và giúp khách hàng kiểm tra tài khoản, rút tiền mặt, chuyển khoản, thanh toán tiền hàng hóa dịch vụ, thầu chi tài khoản, ...

Sản phẩm thẻ là nguồn thu mang tính chất chiến lược của Ngân hàng. Ngân hàng muốn tìm hiểu về các sản phẩm thẻ mà khách hàng sử dụng được kết hợp với các sản phẩm khác như thẻ nào nhằm phát triển các sản phẩm/ dịch vụ của mình tốt hơn nữa nhằm giữ được các khách hàng cũ và tiếp cận thêm với nhiều khách hàng mới. Việc sử dụng luật kết hợp trong trường hợp này là khá tự nhiên và nó hỗ trợ rất lớn trong việc dự báo, phân tích và đưa ra quyết định.

Tuy nhiên, Khai phá luật kết hợp trong lĩnh vực Ngân hàng có khá nhiều thách thức:

- Các tham số đầu vào lấy từ dữ liệu trong Ngân hàng thường khá lớn. Các tham số này có từ nhiều nguồn (bản thân Ngân hàng hoặc các tổ chức tài chính khác, ...). Lựa chọn tham số để xây dựng mô hình khai phá dữ liệu là không hề đơn giản.
- Các tri thức có được nhờ luật kết hợp cần phải được kiểm chứng bởi các chuyên gia kinh tế. Các chuyên gia sẽ là người quyết định giữ lại hay loại bỏ luật.
- Số lượng luật sinh ra là rất lớn nên khó có thể quan sát hết được, vì thế cần có các công cụ hỗ trợ khác nữa, ...

\* Cơ sở dữ liệu vào:

Tập dữ liệu đầu vào là danh sách các sản phẩm tương ứng khách hàng sử dụng. Điều đó có nghĩa là một khách hàng có thể sử dụng cùng lúc nhiều dịch vụ.

\* Cơ sở dữ liệu ra:

Luật kết hợp tìm được với độ hỗ trợ và độ tin cậy của luật kết hợp.

Ví dụ: {TheTinDung} --> {ATM}. Độ hỗ trợ = 31.63%, Độ tin cậy = 81.58%.

Kết quả luật trên có nghĩa là 81.58% khách hàng sử dụng thẻ tín dụng thì thường sử dụng kèm ATM, 31.63% khách hàng sử dụng cả hai dịch vụ trên.

**Nhận xét:** Theo định hướng tìm luật kết hợp giữa các sản phẩm thẻ và các dịch vụ khác được ngân hàng cung cấp thì khi chạy chương trình ta thay đổi giá trị của độ hỗ trợ, độ tin cậy ta có nhận xét như sau:

- Nếu chọn giá trị độ hỗ trợ nhỏ và độ tin cậy nhỏ thì số luật sinh ra là nhiều. Ví dụ: Độ hỗ trợ là 0.05 và độ tin cậy là 0.1 thì số luật là 374 luật. Việc có nhiều

luật được sinh ra nên ta khó quan sát hết các luật, tuy nhiên nhờ có nó ta tìm được các luật hiếm nhưng có giá trị.

- Nếu chọn luật có độ hỗ trợ và độ tin cậy lớn thì số luật là quá ít. Ví dụ: Độ hỗ trợ là 0.2 và độ tin cậy là 0.6 thì số luật là 20 luật.
- Thông thường ta nên chọn độ hỗ trợ nhỏ và độ tin cậy khá lớn. Ví dụ: Độ hỗ trợ là 0.1 và độ tin cậy là 0.5 thì số luật là 80 luật. Việc chọn này vừa đảm bảo không phải tìm quá nhiều luật, vừa đảm bảo ít bỏ sót luật có giá trị.

Với dữ liệu thu thập và phân tích được ta có thể khai thác được một số luật nhằm giúp cho ngân hàng đưa ra các quyết định phục vụ cho hoạt động kinh doanh của họ.

**Với thẻ tín dụng:** Từ thẻ tín dụng ta có một số luật sau:

{TheTinDung} --> {TheGhiNo}. Độ hỗ trợ = 19.39%, Độ tin cậy = 50.00%

{TheTinDung} --> {ATM}. Độ hỗ trợ = 31.63%, Độ tin cậy = 81.58%

{TheTinDung} --> {DichVuKhac}. Độ hỗ trợ = 12.22%, Độ tin cậy = 34.14%

{TheTinDung} --> {VayTien}. Độ hỗ trợ = 22.45%, Độ tin cậy = 57.89%

{TheTinDung} --> {TietKiem}. Độ hỗ trợ = 24.49%, Độ tin cậy = 63.16%

{TheTinDung} --> {ATM, TheGhiNo}. Độ hỗ trợ = 19.39%, Độ tin cậy = 50.00%

{TheTinDung} --> {TheGhiNo, VayTien}. Độ hỗ trợ = 16.33%, Độ tin cậy = 42.11%

{TheTinDung} --> {ATM, VayTien}. Độ hỗ trợ = 22.45%, Độ tin cậy = 57.89%

{TheTinDung} --> {ATM, TietKiem}. Độ hỗ trợ = 23.47%, Độ tin cậy = 60.53%

{TheTinDung} --> {TietKiem, VayTien}. Độ hỗ trợ = 14.29%, Độ tin cậy = 36.84%

{TheTinDung} --> {ATM, TheGhiNo, VayTien}. Độ hỗ trợ = 16.33%, Độ tin cậy = 42.11%

{TheTinDung} --> {ATM, TietKiem, VayTien}. Độ hỗ trợ = 14.29%, Độ tin cậy = 36.84%

{TheTinDung} --> {Pos}. Độ hỗ trợ = 11.22%, Độ tin cậy = 28.95%

{TheTinDung} --> {ChuyenTien}. Độ hỗ trợ = 10.20%, Độ tin cậy = 26.32%

Ta thấy, khách hàng sử dụng thẻ tín dụng có xu hướng sử dụng ATM rất lớn (độ hỗ trợ = 31.63%, độ tin cậy = 81.58%). Việc sử dụng thẻ tín dụng cùng với thẻ ghi nợ là không đáng kể, trong thực tế thì lượng khách hàng sử dụng cả hai dịch vụ này chiếm tỷ lệ nhỏ vì xét cho đến cùng bản chất của hai dịch vụ này là khác nhau, tuy nhiên thực tế trong quá trình sử dụng thẻ thì ranh giới giữa hai loại thẻ này đang ngày càng mờ nhòe vì ngân hàng ngày càng có nhiều dịch vụ áp dụng cho thẻ. Một điểm đáng chú ý là khách hàng sử dụng thẻ tín dụng thì thường cũng có xu hướng vay tiền (độ hỗ trợ = 22.45%, độ tin cậy = 57.89%) hay gửi tiết kiệm (độ hỗ trợ = 24.49%, độ tin cậy = 63.16%). Phải chăng từ luật này thì ngân hàng có thể áp dụng để đưa vào thực tế đó là có chế độ đặc biệt khuyến khích khách hàng gửi/ vay tiền ngân hàng dễ dàng mở thẻ tín dụng và ngược lại. Tuy nhiên ta thấy khách hàng sử dụng Thẻ tín dụng

cùng với dịch vụ khác là khá ít (độ hỗ trợ = 12.22%, độ tin cậy = 34.14%). Một trong những lý do dẫn đến điều này có thể là do các dịch vụ khác mà ngân hàng cung cấp còn chưa phong phú. Ngân hàng nên tăng cường tốt hơn nữa các dịch vụ của mình để thu hút thêm khách hàng và tăng cường khả năng cạnh tranh với các ngân hàng khác,...

**Với thẻ ghi nợ:** Từ thẻ ghi nợ ta có một số luật sau:

{TheGhiNo} --> {VayTien}. Độ hỗ trợ = 20.41%, Độ tin cậy = 35.71%  
 {TheGhiNo} --> {TheTinDung}. Độ hỗ trợ = 19.39%, Độ tin cậy = 33.93%  
 {TheGhiNo} --> {Pos}. Độ hỗ trợ = 22.45%, Độ tin cậy = 39.29%  
 {TheGhiNo} --> {DichVuKhac}. Độ hỗ trợ = 24.49%, Độ tin cậy = 42.86%  
 {TheGhiNo} --> {ChuyenTien}. Độ hỗ trợ = 25.51%, Độ tin cậy = 44.64%  
 {TheGhiNo} --> {DichVuKhac}. Độ hỗ trợ = 24.49%, Độ tin cậy = 42.86%  
 {TheGhiNo} --> {ATM, VayTien}. Độ hỗ trợ = 20.41%, Độ tin cậy = 35.71%  
 {TheGhiNo} --> {ATM, TheTinDung}. Độ hỗ trợ = 19.39%, Độ tin cậy = 33.93%  
 {DichVuKhac} --> {TheGhiNo}. Độ hỗ trợ = 24.49%, Độ tin cậy = 57.14%

Cũng giống như Thẻ tín dụng thì Khách hàng sử dụng thẻ ghi nợ có xu hướng sử dụng ATM rất lớn (độ hỗ trợ = 39.80%, độ tin cậy = 69.64%). So với thẻ tín dụng thì khách hàng sử dụng thẻ ghi nợ cùng với các dịch vụ khác là khá tốt (độ hỗ trợ = 24.49%, độ tin cậy = 42.86%). Ngoài ra khách hàng còn thường sử dụng Thẻ ghi nợ cùng với các dịch vụ như: vay tiền, pos, chuyển tiền khá nhiều.

Hiện nay, số lượng khách hàng sử dụng thẻ ghi nợ khá lớn, do đó ngân hàng cần có chính sách để giữ chân được các khách hàng cũ đồng thời có thể mở rộng hoạt động thẻ của mình cùng với các sản phẩm khác.

Khi phân tích các luật ta thấy cũng có những luật khá thú vị, ví dụ:

{TheGhiNo, TheTinDung} --> {ATM} có độ hỗ trợ = 19.39%, độ tin cậy = 100.00%. Ở luật trên ta thấy việc sử dụng thẻ ghi nợ, thẻ thanh toán, ATM chiếm tỷ lệ nhỏ 19.39% tuy nhiên khi sử dụng thẻ ghi nợ, thẻ tín dụng thì xu hướng dùng ATM lên tới 100%.

Ở cả hai loại thẻ trên thì việc khách hàng sử dụng kết hợp thẻ với ATM là rất lớn. Do đó, các ngân hàng cần không ngừng hoàn thiện và tăng cường cho hệ thống ATM để tạo thuận lợi cho khách hàng. Mặc dù việc sử dụng thẻ cùng với các dịch vụ khác (e-banking, đầu tư chứng khoán, ...) còn ít nhưng trong tương lai thì ngân hàng cũng nên đầu tư để mở rộng thị phần này, bởi xét cho cùng thì đây cũng là xu thế tất yếu. Ngoài ra ta thấy, khách hàng sử dụng các dịch vụ của Ngân hàng còn thiếu sự kết hợp các sản phẩm/ dịch vụ với nhau, chưa tận dụng hết các tiện ích gia tăng mà các dịch vụ mang lại. Việc sử dụng ATM cùng với các dịch vụ khác còn quá ít so với nhu cầu thực sự của khách hàng.

Về phía Ngân hàng có một số đề suất:

- Hoàn thiện nâng cao chất lượng và tiếp tục phát triển các sản phẩm/ dịch vụ giá trị gia tăng phục vụ tốt hơn nhu cầu của khách: dịch vụ nạp tiền Topup, dịch vụ thu hộ cước phí Smart Bill, dịch vụ thương mại điện tử Smart Ecom, ...
- Mở rộng kết nối hệ thống ATM/ POS, ...để khách hàng có thể dễ dàng sử dụng các dịch vụ đi kèm.
- Hoàn thiện và thương mại hóa sản phẩm dịch vụ cung cấp.
- Triển khai các kế hoạch truyền thông, kế hoạch marketing sản phẩm/ dịch vụ. Ngân hàng nên triển khai các dịch vụ cung cấp hướng tới từng đối tượng khách hàng cụ thể. Thực tế cho thấy mỗi khi có những đợt khuyến mại thì lượng khách đăng ký sử dụng thẻ tăng nhiều, nhưng vẫn có tình trạng khách đăng ký dùng thẻ nhưng sau lại không dùng. Nên chẳng ngân hàng nên hướng dịch vụ vào từng đối tượng người dùng hơn nữa. Ví dụ như phát triển thẻ cho các cơ quan, doanh nghiệp,...
- Kết hợp với hệ thống tài chính, hệ thống siêu thị, khách sạn, hàng không, ...để tăng cường khả năng sử dụng thẻ.

Chạy chương trình và phân tích kết quả thì ta có một số kết luận:

**Ưu điểm:** Luật kết hợp mà ta khai thác được có thể xem như một tư liệu quý giúp cho những người làm ngân hàng có những quyết định hợp lý. Mặc dù không phải luật nào sinh ra cũng là có giá trị, nhưng nếu tìm được một vài luật tốt cũng có thể đã mang lại nhiều lợi ích về kinh tế.

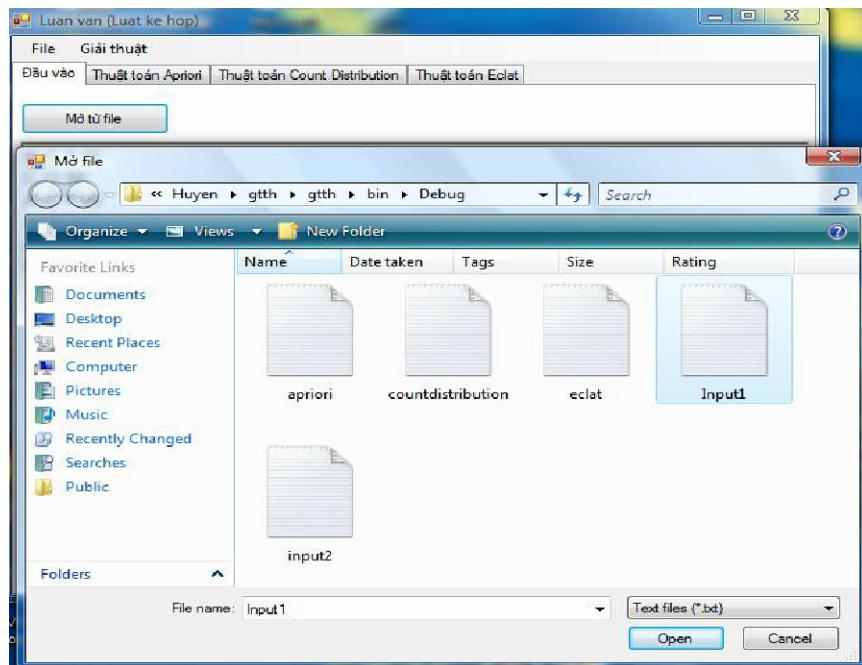
**Nhược điểm:**

- Số thuộc tính và dữ liệu để khai phá còn ít. Trong thực tế thì ngay cả từng sản phẩm thẻ tín dụng hay ghi nợ lại được chia ra làm rất nhiều các sản phẩm khác nữa, ngoài ra ngân hàng còn có nhiều dịch vụ mà ta vẫn chưa phân tích hết được,...
- Luật kết hợp tìm được còn nhiều, chưa tập trung, khó khăn cho người khai thác luật tìm được.

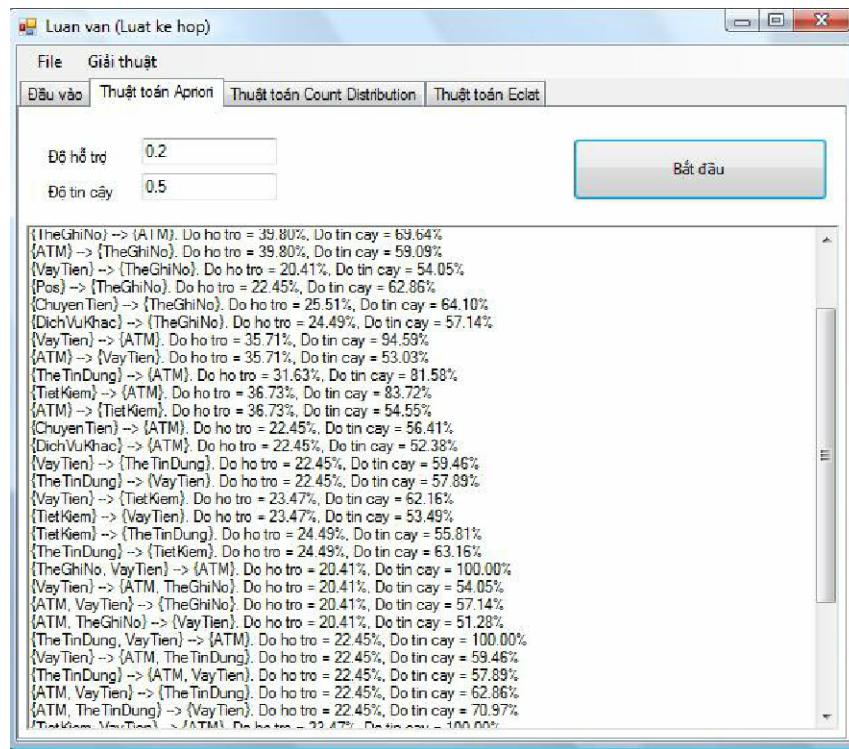
**Hướng phát triển trong tương lai:**

- Mở rộng số thuộc tính, dữ liệu khai phá để tìm được nhiều luật có ý nghĩa hơn.
- Nghiên cứu, cải tiến chương trình để có thể áp dụng được với dữ liệu ngày càng gia tăng trong ngân hàng.
- Kết quả thể hiện của chương trình là các luật, cần phát triển thêm những phân tích khác để chương trình có tính ứng dụng cao hơn.

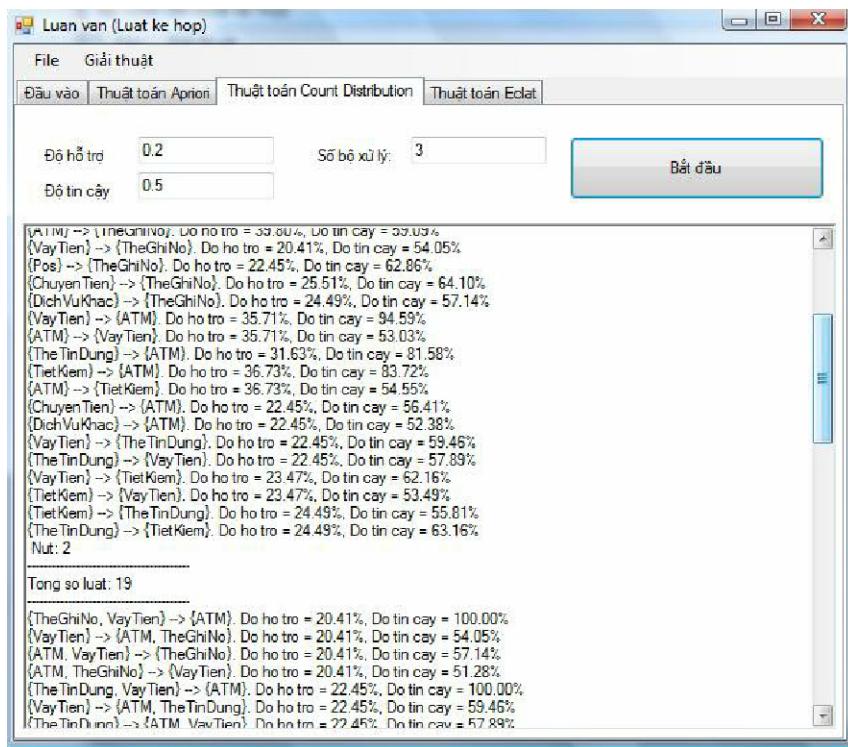
### 3. 1. 3. Giao diện chương trình



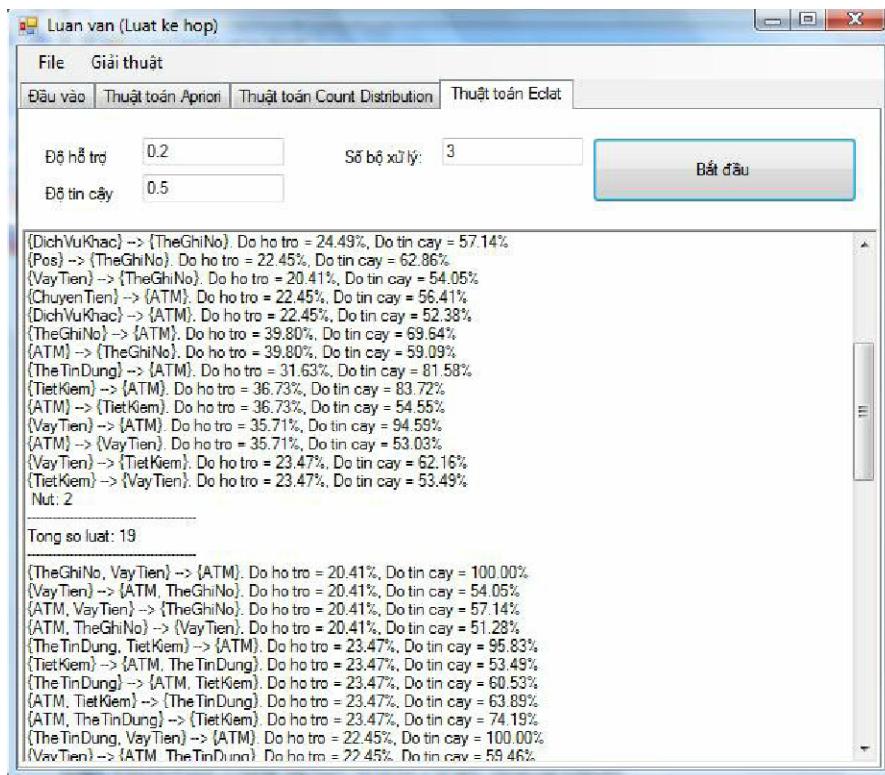
Hình 3.1. Giao diện nhập dữ liệu đầu vào



Hình 3. 2. Giao diện thực hiện theo thuật toán Apriori



Hình 3. 3. Giao diện thực hiện theo thuật toán song song Count Distribution



Hình 3. 4. Giao diện thực hiện theo thuật toán song song Eclat

### 3. 2. Đánh giá kết quả

#### 3. 2. 1. Phương pháp đánh giá các chương trình song song

##### 3. 2. 1. 1. Đánh giá thời gian thực hiện song song

Để đánh giá được độ phức tạp tính toán của các thuật toán song song, ngoài việc xác định số bước tính toán chúng ta còn cần đánh giá thời gian truyền thông của các tiến trình. Trong một hệ thống truyền thông điệp, thời gian truyền thông điệp cũng được xem xét trong thời gian thực hiện của thuật toán. Thời gian thực hiện song song, ký hiệu là  $tp$  gồm hai phần:  $tcomp$  là thời gian tính toán và  $tcomm$  là thời gian truyền thông dữ liệu. Như vậy chúng ta có:

$$tp = tcomp + tcomm$$

Thời gian tính toán  $tcomp$  được xác định giống như thuật toán tuần tự, bằng cách đếm số lượng các bước tính toán. Khi có nhiều hơn một tiến trình được thực hiện đồng thời thì chỉ cần tính thời gian thực hiện của tiến trình phức tạp nhất (thực hiện lâu nhất). Thông thường, tất cả các tiến trình thực hiện cùng một thao tác tính toán, nên chúng ta chỉ cần đếm một cách đơn giản số lượng các bước tính toán của một tiến trình. Trong trường hợp khác, chúng ta sẽ tìm số lượng các bước tính toán lớn nhất của những tiến trình thực hiện trong cùng một thời gian.

Để thuận tiện chúng ta bỏ qua thời gian tính toán trong thành phần phân chia thời gian truyền thông điệp. Khi đó thời gian cho tính toán là:

$$tcomp = tcomp1 + tcomp2 + tcomp3 + \dots$$

##### 3. 2. 1. 2. Thời gian truyền thông

Thời gian truyền thông phụ thuộc vào: số lượng các thông điệp, kích thước của mỗi thông điệp, cấu hình kết nối mạng đường truyền và cách thức truyền tải thông điệp,... Công thức ước lượng thời gian truyền thông được xác định là:

$$tcomm = tstartup + n * tdata$$

Trong đó:

- +  $tstartup$  là thời gian khởi động (thời gian tối thiểu) cần để truyền một thông báo không có dữ liệu. Bao gồm cả thời gian đóng gói thông điệp ở nơi gửi và thời gian mở gói thông điệp ở nơi nhận. Để đơn giản chúng ta giả thiết thời gian này là hằng số.

- +  $tdata$  là thời gian cần để gửi một từ (word) dữ liệu (hay một mục dữ liệu) từ nơi gửi tới nơi nhận, được giả thiết là một hằng số và có  $n$  (word) là số từ dữ liệu được trao đổi trong hệ thống. Tốc độ truyền được đo bằng bit/ giây.

Thời gian truyền thông cuối cùng  $tcomm$  sẽ là tổng của thời gian truyền thông của tất cả các thông điệp tuần tự từ một tiến trình.

##### 3. 2. 1. 3. Tỉ lệ tính toán/ truyền thông

Thông thường, truyền thông rất hao tốn (mất nhiều thời gian), nếu cả tính toán và truyền thông cùng độ phức tạp thì tăng n sẽ không chắc cải thiện được sự thực thi.