MỤC LỤC

PHẦN MỞ ĐẦU

Sự phát triển mạnh mẽ của công nghệ phần cứng đã tạo nên các máy tính có bộ xử lý tốc độ cao, bộ nhớ dung lượng lớn và cùng với điều đó, là sự phát triển không ngừng các hệ thống mạng viễn thông. Từ các kết quả đó, nhiều hệ thống thông tin phục vụ việc tự động hóa mọi hoạt động kinh doanh cũng như quản lý đã được triển khai với tốc độ tăng trưởng vượt bậc. Điều này đã tạo ra những dòng dữ liệu khổng lồ trở thành hiện tượng "bùng nổ thông tin" như nhiều người quan niệm. Nhiều hệ quản trị cơ sở dữ liệu mạnh với các công cụ phong phú và thuận tiện đã giúp con người khai thác có hiệu quả các nguồn tài nguyên dữ liệu lớn nói trên. Cùng với việc khối lượng dữ liệu được quản lý tăng không ngừng, các hệ thống thông tin cũng được chuyên môn hóa theo các lĩnh vực ứng dụng như sản xuất, tài chính, kinh doanh, y học,... Như vậy, bên cạnh chức năng khai thác dữ liệu có tính chất tác nghiệp, sự thành công trong kinh doanh không chỉ là năng suất của các hệ thông tin mà còn là tính linh hoạt và sẵn sàng đáp lại những nhu cầu trong thực tế, hay nói khác đi, người ta còn mong muốn các cơ sở dữ liệu cần đem lại tri thức từ dữ liệu hơn là chính bản thân dữ liệu. Để lấy được các thông tin mang tính tri thức trong khối dữ liệu khổng lồ như đã nói, cần thiết phải phát triển các kỹ thuật có khả năng hợp nhất các dữ liệu từ các hệ thống giao dịch khác nhau, chuyển đổi chúng thành một tập hợp các cơ sở dữ liệu ổn định, có chất lượng để sử dụng theo một số mục đích nào đó. Các kỹ thuật như vậy được gọi chung là các kỹ thuật *tạo kho dữ liệu* và môi trường các dữ liệu nhận được sau khi áp dụng các kỹ thuật nói trên được gọi là các *kho dữ liệu*.

Các kho dữ liệu có thể giúp khai thác thông tin bằng các công cụ truy vấn và báo cáo, cũng như được sử dụng để hỗ trợ việc phân tích trực tuyến, kiểm định các giả thuyết. Tuy nhiên, nếu chỉ có các kho dữ liệu thì chưa thể có được tri thức.

Chúng không có khả năng đưa ra các giả thuyết. Nếu dữ liệu được phân tích một cách thông minh thì chúng sẽ là nguồn tài nguyên vô cùng quý giá. Từ các dữ liệu sẵn có, nhu cầu tìm ra những thông tin tiềm ẩn có giá trị (những tài nguyên quý giá) chưa được phát hiện, những xu hướng phát triển và những yếu tố tác động lên chúng là một điều hết sức cần thiết. Tiến hành công việc như vậy chính là thực hiện quá trình phát hiện tri thức trong cơ sở dữ liệu (Knowledge Discovery in Databases - KDD) mà trong đó kỹ thuật khai phá dữ liệu (data mining) cho phép phát hiện được các tri thức tiềm ẩn.

Nếu phát hiện tri thức là toàn bộ quá trình rút ra tri thức hữu ích từ cơ sở dữ liệu thì khai phá dữ liệu là giai đoạn chính của quá trình này [7]. Giai đoạn khai phá dữ liệu được thực hiện sau các khâu tinh lọc và tiền xử lý dữ liệu, nhằm tìm ra các mẫu, các xu hướng có ý nghĩa từ các tập dữ liệu được hi vọng là sẽ thích hợp với nhiệm vụ khai phá. Chỉ các mẫu, các xu hướng được xem là đáng quan tâm (xét theo một phương diện nào đó) mới được coi là tri thức, và tri thức là có ích khi nó có thể giúp đạt được mục đích của hệ thống hoặc người dùng. Người ta đã sử dụng các kỹ thuật và các khái niệm của các lĩnh vực đã được nghiên cứu từ trước như học máy, nhận dạng, thống kê, hồi quy, xếp loại, phân nhóm, các mô hình đồ thị, mạng Bayes... để khai phá các khối dữ liệu của kho dữ liệu nhằm phát hiện ra các mẫu mới, các tương quan mới, các xu hướng có ý nghĩa.

Một trong các nội dung cơ bản nhất trong khai phá dữ liệu và rất phổ biến 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 đến sự xuất hiện của một (hoặc một tập) thuộc tính khác như thế nào. Điều đó có thể được diễn giải như sau. Cho một lược đồ $R = \{A_1, A_2,..., A_p\}$ các thuộc tính với miền giá trị $\{0, 1\}$ và một quan hệ r trên R, một luật kết hợp trên r được mô tả dưới dạng $X \rightarrow Y$ với $X \subseteq R$ và $Y \in R \setminus X$. Về mặt trực giác, có thể phát

biểu ý nghĩa của luật là: nếu một bản ghi của bảng r có giá trị 1 tại mỗi thuộc tính thuộc X thì giá trị của thuộc tính Y cũng là 1 trong bản ghi đó.

Cho $W \subseteq R$, đặt $s(W, r)$ là tần số xuất hiện của W trong r được tính bằng tỉ lệ của các hàng trong r có giá trị 1 tại mỗi cột thuộc W. Tần số xuất hiện, còn gọi là độ hỗ trợ của luật $X \rightarrow Y$ trong r được định nghĩa là $s(X \cup \{Y\}, r)$, độ tin cậy của luật là $s(X \cup \{Y\}, r)/s(X, r)$. Ở đây X có thể gồm nhiều thuộc tính, B là giá trị không cố định, và ta thấy không gian tìm kiếm có kích thước tăng theo hàm mũ của số các thuộc tính ở đầu vào. Nhiệm vụ của việc phát hiện các luật kết hợp là phải tìm tất cả các luật $X \rightarrow Y$ sao cho độ hỗ trợ của luật không nhỏ hơn ngưỡng σ cho trước và độ tin cậy của luật không nhỏ hơn ngưỡng α cho trước. Từ một cơ sở dữ liệu ta có thể tìm ra hàng nghìn, thậm chí hàng trăm nghìn các luật kết hợp.

Do việc phát hiện luật kết hợp đòi hỏi lượng tính toán và truy xuất dữ liệu lớn, cùng với sự phân tán của dữ liệu, đặc biệt trên các cơ sở dữ liệu trực tuyến, một giải pháp tự nhiên được nghĩ đến là áp dụng tính toán song song, bởi các máy tính song song vốn có khả năng thực hiện nhanh lượng tính toán lớn và xử lý tốt lượng dữ liệu lớn [4, 10, 15, 17]. Các thuật toán phát hiện luật kết hợp có thể được song song hóa theo nhiều cách khác nhau: chúng ta có thể tìm kiếm độc lập, song song hóa hoặc lặp lại một thuật toán tuần tự. Để chọn được chiến lược phù hợp, chúng ta cần dựa trên các độ đo về tính phức tạp và chi phí cho lập trình song song với mỗi chiến lược.

Vấn đề dư thừa dữ liệu hoặc dữ liệu không đầy đủ trong hệ thông tin có thể được khắc phục bằng cách sử dụng khái niệm tập thô do Pawlak đưa ra [14, 1]. Tập thô cho phép chia bảng quyết định thành các thuộc tính điều kiện và thuộc tính quyết định, trong đó thông tin tương ứng với các thuộc tính quyết định tuỳ thuộc vào thông tin tương ứng với các thuộc tính điều kiện, phù hợp với cách biểu diễn các luật kết hợp. Việc nghiên cứu luật kết hợp thông qua cách tiếp cận tập thô đã được

Tetsuya Murai, Yoshiharu Sato đề xuất trong [12]. Hệ thông tin được phân hoạch thành tập các tập cơ bản, mà giá trị của tập thô trong mỗi tập cơ bản là giống nhau, từ đó phần tử đại diện cho mỗi tập cơ bản được chọn ra, ta có được rút gọn của bảng quyết định để giảm bớt khối lượng thông tin điều kiện dư thừa có trong bảng quyết định. Mối quan hệ của luật kết hợp trong các hệ thông tin con S_i với luật kết hợp trong hệ thông tin hợp thành $S = \cup \{S_i\}$ được tìm hiểu để tìm ra điều kiện cho tính khả tách của hệ thông tin, từ đó có thể phát hiện song song luật kết hợp dựa trên phân tán theo dữ liệu.

Luận văn với đề tài "***Luật kết hợp theo tiếp cận lý thuyết tập thô và khai phá dữ liệu song song***" khảo sát lĩnh vực phát hiện tri thức trong cơ sở dữ liệu, trong đó tập trung vào các nội dung phát hiện luật kết hợp theo cách tiếp cận của tập thô. Mô hình song song phát hiện luật kết hợp cũng được xem xét với việc phân tích một số thuật toán song song phát hiện luật kết hợp.

Phương pháp nghiên cứu chính yếu của luận văn là khảo sát các bài báo khoa học được xuất bản trong một vài năm gần đây từ đó đưa ra được một số ý tưởng nhằm cải tiến thuật toán.

Nội dung của bản luận văn này gồm có Phần mở đầu, ba chương và Phần kết luận. Cuối mỗi chương của bản luận văn có phần kết luận chương trình bày tóm tắt những nội dung chính yếu trong nội dung của chương.

Chương một giới thiệu một số nội dung cơ bản về khai phá dữ liệu và phát hiện tri thức trong cơ sở dữ liệu (mục 1.1), các hệ thống đa xử lý và tính toán song song (mục 1.2.1); và các chiến lược và mô hình chi phí của khai phá dữ liệu song song (mục 1.2.2, 1.2.3). Một số nội dung trong chương này được trích dẫn từ các tài liệu [2], [7], [9]. Đây là những kiến thức nền tảng làm cơ sở để cho nội dung các chương sau và việc thiết lập các thuật toán.

Chương hai của bản luận văn trình bày về khái niệm và một số công nghệ phát hiện luật kết hợp (mục 2.1); lý thuyết tập thô và vấn đề khai phá dữ liệu theo cách tiếp cận tập thô (mục 2.1). Một thuật toán tìm tập tối ưu các luật và thuật toán cải tiến của nó được trình bày (mục 2.2.2, thuật toán 2.1, 2.2) cùng với độ phức tạp về thời gian tính toán. Hai thuật toán này được dùng làm cơ sở đề xuất ra mô hình song song tương ứng trong chương 3.

Chương thứ ba trình bày tóm tắt một số thuật toán phát hiện song song luật kết hợp trên các nền phần cứng khác nhau và so sánh chúng (mục 3.2). Qua khảo sát một bài toán hệ thông tin của Sở Y tế Hà Nội [3], luận văn cũng đề xuất một mô hình phát hiện song song luật kết hợp theo cách tiếp cận tập thô, trong đó cơ sở dữ liệu được trình bày dưới dạng một bảng quyết định, và việc song song hóa được thực hiện trên các bước dữ liệu (mục 3.3).

Phần kết luận đưa ra một số nội dung liên quan đến phương hướng nghiên cứu phát triển nội dung của luận văn này: phát triển mô hình phát hiện luật kết hợp và thử nghiệm trên hệ thống tính toán song song thực sự.

Nội dung cơ bản của bản luận văn đã được trình bày tại xê-mi-na khoa học tại bộ môn Các Hệ thống Thông tin, Khoa Công nghệ, Đại học Quốc gia Hà Nội.

Luận văn này được thực hiện dưới sự hướng dẫn khoa học của TS. Hà Quang Thụy. Tôi xin bày tỏ lòng biết ơn sâu sắc tới Thầy đã có những chỉ dẫn tận tình quý báu giúp tôi có thể hoàn thành bản luận văn. Tôi xin chân thành cảm ơn các thầy giáo và bạn bè trong bộ môn Các Hệ thống Thông tin đã có những góp ý hữu ích trong quá trình thực hiện bản luận văn. Tôi cũng xin cảm ơn các thầy cô giáo trong khoa, cán bộ thuộc phòng Khoa học và Đào tạo, Khoa Công nghệ, đã tạo điều kiện thuận lợi giúp đỡ tôi trong quá trình học tập và nghiên cứu tại Khoa. Tôi vô cùng cảm ơn những người thân trong gia đình và bạn bè đã luôn động viên khích lệ để tôi có thể hoàn thành bản luận văn này.

CHƯƠNG I. TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU VÀ KHAI PHÁ DỮ LIỆU SONG SONG

I.1. KHAI PHÁ DỮ LIỆU VÀ PHÁT HIỆN TRI THỨC TRONG CƠ SỞ DỮ LIỆU

I.1.1. Sơ bộ về khai phá dữ liệu và phát hiện tri thức trong cơ sở dữ liệu

Phát hiện tri thức trong cơ sở dữ liệu là quá trình khám phá những tri thức có ích từ một lượng lớn dữ liệu được lưu trong các cơ sở dữ liệu. Do các dữ kiện dạng điện tử được thu thập và tích lũy ngày càng nhiều, do nhu cầu chuyển các dữ liệu đó thành các thông tin và tri thức có ích cho các ứng dụng rộng rãi như phân tích thị trường, quản trị doanh nghiệp, hỗ trợ quyết định ngày càng tăng, cho nên lĩnh vực phát hiện tri thức đã ngày càng được quan tâm trong ngành công nghiệp thông tin trong những năm gần đây [7].

Các cơ sở dữ liệu được xây dựng với mục đích quản lý, tập hợp các dữ liệu có tổ chức và theo đó, một kết quả tự nhiên là con người có được một khối lượng dữ liệu rất lớn. Nhiều dữ liệu nghĩa là có thể có nhiều thông tin. Các chuyên gia được đào tạo về phân tích hỗ trợ quyết định đã phân tích những dữ liệu đó và phát hiện ra thông tin dưới dạng các mẫu và các quy luật tiềm ẩn sau quan hệ giữa các thuộc tính khác nhau trong dữ liệu. Việc này giúp cho các doanh nghiệp thấy được kết quả của các hoạt động trước đây và định hướng cho các hoạt động sắp tới. Tuy nhiên, lượng dữ liệu sẵn có đã trở nên quá lớn để có thể dễ dàng phát hiện được các thông tin như vậy.

Một ứng dụng khác của phát hiện tri thức là cung cấp các hỗ trợ quyết định tác nghiệp [9]. Không như cách tiếp cận hỗ trợ quyết định theo chu kỳ, trong đó thời gian từ thời điểm phát hiện ra thông tin tới thời điểm dùng các thông tin đó trong quá trình ra quyết định có thể mất nhiều tuần hoặc nhiều tháng (chúng thường được dùng để hỗ trợ quyết định dài hạn cho doanh nghiệp), hỗ trợ quyết định tác nghiệp

của phát hiện tri thức có thể diễn ra trong vài phút và được dùng để cung cấp hỗ trợ quyết định ngắn hạn hoặc tức thì trong một tập rất ít các trường hợp, thậm chí trong một trường hợp. Có được các hỗ trợ như vậy do phát hiện tri thức đã cung cấp các kỹ thuật, công cụ đặc thù thao tác tới dữ liệu.

Trong quá trình phát hiện tri thức, một số kiểu phân tích khác nhau có thể được dùng để phát hiện được các mẫu và quy luật từ dữ liệu đã có sẵn, trong một tình huống được đặt ra của doanh nghiệp, sau đó thông tin có thể được lưu lại như một mô hình toán học trừu tượng của dữ liệu vốn có, được coi như một mô hình phát hiện tri thức. Sau khi đã tạo được mô hình phát hiện tri thức, dữ liệu mới có thể được kiểm tra trong mô hình để xem liệu nó có phù hợp với mẫu và quy luật mong muốn không. Từ thông tin này, có thể có các hành động để cải thiện kết quả trong một tình huống được doanh nghiệp đặt ra.

Một định nghĩa khác về phát hiện tri thức là quá trình nhằm xác định ra các mẫu có giá trị, mới, có tiềm năng sử dụng và dễ hiểu từ dữ liệu [7]. Các nội dung sau đây hình thức hóa định nghĩa này. Nếu coi dữ liệu là một tập các sự kiện F thì mẫu là một biểu thức E trong ngôn ngữ L mô tả các sự kiện trong một tập con F_E của F, biểu thức này phải đơn giản hơn là việc liệt kê tất cả các sự kiện trong F. Các tính chất có giá trị, có tiềm năng sử dụng, dễ hiểu của mẫu lần lượt được đo bằng các hàm C, U, S; các hàm này ánh xạ các biểu thức trong ngôn ngữ L vào các không gian đo có thứ tự toàn phần hay thứ tự bộ phận M_C, M_U, M_S.

Các mẫu thu được là ***mới*** nếu có các thay đổi trong dữ liệu khi so sánh giá trị hiện tại với giá trị cũ hoặc giá trị dự đoán, hoặc cho thấy các giá trị mới tìm được liên quan thế nào với các giá trị cũ, ký hiệu tính mới mẻ của mẫu là $N(E, F)$, nó có thể là một hàm logic hoặc một phép đo về mức độ mới hoặc không ngờ tới của mẫu. Một khái niệm quan trọng khác là ***tính thú vị***, thường được coi là độ đo tổng thể giá trị của mẫu, tính thú vị có thể được đo bằng một hàm I trong không gian độ đo

M_I: $i = I(E, F, C, N, U, S)$. Mẫu $E \in L$ được gọi là tri thức nếu với ngưỡng i do người dùng định nghĩa, ta có $I(E, F, C, N, U, S) > i$.

Nhìn chung, quá trình phát hiện tri thức là một chuỗi nối tiếp và lặp lại các bước sau:

- làm sạch dữ liệu: xử lý các dữ liệu có lỗi, bị nhiễu, thiếu dữ liệu hoặc dữ liệu không thích hợp;
- tích hợp dữ liệu: các nguồn dữ liệu bị lặp lại, không đồng nhất có thể được tích hợp làm một;
- lựa chọn dữ liệu: lấy ra các dữ liệu liên quan tới công việc phân tích;
- biến đổi dữ liệu: dữ liệu được biến đổi hoặc củng cố dưới các dạng thích hợp để khai phá bằng cách thực hiện các thao tác tóm tắt hay tập hợp.
- khai phá dữ liệu: quá trình cốt yếu để áp dụng các phương pháp thông minh nhằm tách ra các mẫu dữ liệu;
- đánh giá mẫu: xác định các mẫu thực sự thú vị biểu diễn tri thức dựa trên một số độ đo tính thú vị;
- biểu diễn tri thức: dùng các kỹ thuật biểu diễn tri thức và trực quan hóa để đưa ra tri thức mới khai phá được cho người dùng.

Từ việc sẵn có các hệ cơ sở dữ liệu quan hệ và các kho dữ liệu, bốn bước đầu tiên: làm sạch dữ liệu, tích hợp dữ liệu, lựa chọn dữ liệu và biến đổi dữ liệu có thể được thực hiện bằng cách xây dựng các kho dữ liệu và thực hiện một số phép xử lý phân tích trực tuyến (OLAP) trên kho dữ liệu đó. Đôi khi các bước khai phá dữ liệu, đánh giá mẫu và biểu diễn tri thức được kết hợp vào làm một quá trình (thường là lặp lại), được gọi là khai phá dữ liệu. Việc khai phá dữ liệu này được tiến hành trên tập dữ liệu có hi vọng là sẽ thích hợp với nhiệm vụ khai phá để có được các mẫu thú vị, chứ không phải trên toàn bộ dữ liệu trong thời gian đủ dài để có các mẫu không thực sự có ích như khái niệm trong thống kê trước đây.

I.1.2. Nội dung của khai phá dữ liệu

I.1.2.1 Các nhiệm vụ chính của khai phá dữ liệu

Công việc khai phá dữ liệu có thể chia làm hai loại: khai phá dữ liệu mô tả và khai phá dữ liệu dự đoán [2, 7]. Loại thứ nhất mô tả dữ liệu một cách ngắn gọn, tóm tắt và trình bày các tính chất chung đáng quan tâm của dữ liệu. Loại thứ hai xây dựng một hoặc một tập các mô hình, thực hiện các phép suy luận trên dữ liệu sẵn có và dự đoán hành vi của các tập dữ liệu mới.

Các mục tiêu mô tả và dự đoán đạt được thông qua các công việc khai phá dữ liệu chính sau đây:

- **Phân lớp** là việc học một hàm ánh xạ một mẫu dữ liệu vào một trong số các lớp đã xác định. Quá trình này phân tích một tập dữ liệu huấn luyện (tức là một tập các đối tượng mà ta đã biết tên lớp của nó) và xây dựng một mô hình cho mỗi lớp dựa trên các đặc tính trong dữ liệu. Một cây quyết định hoặc một tập các luật phân lớp được tạo ra từ quá trình phân lớp đó, nó có thể được dùng để hiểu rõ hơn mỗi lớp trong cơ sở dữ liệu và để phân loại dữ liệu trong tương lai.

Ví dụ, người ta có thể phân loại các bệnh và giúp dự đoán bệnh dựa trên các triệu chứng của bệnh nhân. Phân lớp được dùng trong việc phân nhóm khách hàng, mô hình hóa doanh nghiệp và phân tích tín dụng...

- **Hồi quy** là việc học một hàm ánh xạ từ một mẫu dữ liệu sang một biến dự đoán có giá trị thực. Có rất nhiều các ứng dụng khai phá dữ liệu với nhiệm vụ hồi quy, ví dụ như đánh giá khả năng tử vong của bệnh nhân dựa trên các kết quả xét nghiệm chẩn đoán, dự đoán nhu cầu tiêu thụ một sản phẩm mới bằng một hàm chi tiêu quảng cáo.

- **Phân nhóm (đoạn)** là việc mô tả chung để tìm ra các tập xác định các nhóm để mô tả dữ liệu. Các nhóm có thể tách rời hoặc phân cấp hoặc gối lên nhau, tức là

một dữ liệu có thể vừa thuộc nhóm này, vừa thuộc nhóm khác. Các ứng dụng khai phá dữ liệu có nhiệm vụ phân nhóm như phát hiện tập khách hàng có phản ứng giống nhau trong cơ sở dữ liệu tiếp thị, xác định các loại quang phổ từ các phương pháp đo tia hồng ngoại.

- **Tóm tắt** là phương pháp tìm kiếm một mô tả cô đọng cho một tập con dữ liệu. Ví dụ như việc lập bảng các độ lệch chuẩn và trung bình cho tất cả các trường. Các kỹ thuật tóm tắt thường được áp dụng cho các phân tích dữ liệu tương tác có tính thăm dò và tạo báo cáo tự động.

- **Mô hình hoá phụ thuộc** bao gồm việc tìm kiếm một mô hình mô tả sự phụ thuộc đáng kể giữa các biến. Các mô hình phụ thuộc tồn tại dưới hai mức: mức cấu trúc của mô hình xác định những biến nào là phụ thuộc cục bộ với nhau, và mức định lượng của một mô hình xác định độ mạnh của sự phụ thuộc theo một thước đo nào đó.

- **Phát hiện sự thay đổi và chệch hướng** khai thác những thay đổi đáng kể nhất trong dữ liệu từ các giá trị chuẩn hoặc được đo trước đó.

Các nhiệm vụ khác nhau này đòi hỏi số lượng và dạng thông tin khác nhau nên chúng thường ảnh hưởng đến việc thiết kế và chọn thuật toán khai phá dữ liệu khác nhau.

I.1.2.2 Các thành phần của thuật toán khai phá dữ liệu

Ba thành phần chủ yếu trong một thuật toán khai phá dữ liệu là *biểu diễn mô hình*, *đánh giá mô hình* và *phương pháp tìm kiếm*.

Biểu diễn mô hình là việc xây dựng ngôn ngữ L để miêu tả các mẫu có thể phát hiện được. Nếu sự mô tả này bị giới hạn quá thì sẽ không xây dựng được mô hình chính xác cho dữ liệu, vì thế người phân tích dữ liệu phải hiểu đầy đủ các khả năng tiêu biểu của phương pháp được dùng. Ngoài ra người thiết kế thuật toán cũng

cần chỉ rõ giả thiết mô tả nào được tạo bởi thuật toán nào. Mô hình có khả năng miêu tả quá mạnh sẽ làm tăng nguy cơ dữ liệu huấn luyện quá phù hợp, dẫn đến việc giảm độ chính xác dự đoán các dữ liệu chưa biết, thêm vào đó nó còn làm cho việc tìm kiếm trở nên phức tạp và việc giải thích mô hình khó hơn.

Đánh giá mô hình xem xét một mẫu có đáp ứng được các tiêu chuẩn của quá trình phát hiện tri thức hay không. Việc đánh giá độ chính xác dự đoán dựa trên đánh giá chéo, đánh giá chất lượng mô tả liên quan đến độ chính xác dự đoán, tính mới mẻ, tính hữu ích và dễ hiểu của mô hình. Cả hai tiêu chuẩn thống kê và logic có thể được dùng để đánh giá mô hình.

Phương pháp tìm kiếm bao gồm hai thành phần là tìm kiếm tham số và tìm kiếm mô hình. Thuật toán phải tìm ra các tham số để tối ưu hoá các tiêu chuẩn đánh giá mô hình với các dữ liệu quan sát được và một cách miêu tả mô hình đã định. Trong tìm kiếm mô hình, miêu tả mô hình được thay đổi để xét một họ các mô hình mới. Với mỗi cách biểu diễn mô hình, phương pháp tìm kiếm tham số được áp dụng để để đánh giá chất lượng mô hình. Các phương pháp tìm kiếm mô hình thường sử dụng các kỹ thuật tìm kiếm phỏng đoán do kích thước lớn của không gian các mô hình thường cản trở việc tìm kiếm toàn diện.

I.1.3. Các phương pháp khai phá dữ liệu phổ biến và việc lựa chọn phương pháp

Có rất nhiều các phương pháp khai phá dữ liệu, mỗi phương pháp có đặc điểm riêng về biểu diễn mô hình, đánh giá mô hình và cách tìm kiếm, phù hợp với với một lớp các bài toán với các dạng dữ liệu và miền dữ liệu nhất định. Dưới đây là một số phương pháp phổ biến thường được dùng [9]:

- Phương pháp quy nạp
- Cây quyết định và luật
- Phát hiện luật kết hợp
- Các phương pháp phân lớp và quy hồi phi tuyến

- Phân nhóm và phân đoạn
- Các phương pháp dựa trên mẫu
- Khai phá dữ liệu văn bản
- Mạng nơ-ron
- Thuật toán di truyền.
- Mô hình phụ thuộc dựa trên đồ thị xác suất.
- Mô hình học quan hệ

Các thuật toán khai phá dữ liệu tự động vẫn mới chỉ ở giai đoạn phát triển ban đầu. Người ta vẫn chưa đưa ra được một tiêu chuẩn nào trong việc quyết định sử dụng phương pháp nào và trong trường hợp nào thì có hiệu quả.

Hầu hết các kỹ thuật khai phá dữ liệu đều mới đối với lĩnh vực kinh doanh. Hơn nữa lại có rất nhiều kỹ thuật, mỗi kỹ thuật được sử dụng cho nhiều bài toán khác nhau. Mỗi phương pháp đều có điểm mạnh và điểm yếu của nó, nhưng hầu hết các điểm yếu đều có thể khắc phục được, vì vậy cần tìm cách áp dụng mỗi kỹ thuật một cách thật đơn giản, dễ sử dụng để không cảm thấy những phức tạp vốn có của kỹ thuật đó.

Để so sánh các kỹ thuật cần phải có một tập lớn các quy tắc và các phương pháp thực nghiệm tốt. Thường thì quy tắc này không được sử dụng khi đánh giá các kỹ thuật mới nhất. Vì vậy mà những yêu cầu cải thiện độ chính xác không phải lúc nào cũng thực hiện được.

Nhiều công ty đã đưa ra những sản phẩm sử dụng kết hợp nhiều kỹ thuật khai phá dữ liệu khác nhau với hy vọng nhiều kỹ thuật thì sẽ tốt hơn. Nhưng thực tế cho thấy nhiều kỹ thuật chỉ thêm nhiều rắc rối và gây khó khăn cho việc so sánh giữa các phương pháp và các sản phẩm. Theo nhiều đánh giá cho thấy khi đã hiểu được các kỹ thuật và nghiên cứu tính giống nhau giữa chúng, người ta thấy rằng nhiều kỹ thuật lúc đầu thì có vẻ khác nhau nhưng thực chất khi hiểu ra được các kỹ thuật này thì thấy chúng hoàn toàn giống nhau. Tuy nhiên, đánh giá này cũng chỉ để tham khảo vì cho đến nay, khai phá dữ liệu vẫn còn là kỹ thuật mới chứa nhiều tiềm năng mà người ta vẫn chưa khai thác hết.

I.1.4 Ưu thế của khai phá dữ liệu

Khai phá dữ liệu thực chất không có gì mới mà hoàn toàn dựa trên các phương pháp cơ bản đã biết. Vậy khai phá dữ liệu có gì khác so với các phương pháp đó và tại sao khai phá dữ liệu lại có ưu thế hơn hẳn chúng? Các phân tích sau đây sẽ giải đáp những câu hỏi này [2].

Học máy (Machine Learning)

Tuy phương pháp học máy đã được cải tiến để nó có thể phù hợp với mục đích khai phá dữ liệu nhưng sự khác biệt giữa thiết kế, các đặc điểm của cơ sở dữ liệu đã làm nó trở nên không phù hợp với mục đích này mặc dù cho đến nay phần lớn các phương pháp khai phá dữ liệu vẫn dựa trên nền tảng cơ sở của phương pháp học máy.

Trong các hệ quản trị cơ sở dữ liệu, một cơ sở dữ liệu là một tập hợp dữ liệu được tích hợp một cách logic, được lưu trong một hay nhiều tệp và được tổ chức để lưu trữ, sửa đổi và lấy thông tin một cách hiệu quả và dễ dàng. Trong học máy, thuật ngữ cơ sở dữ liệu chủ yếu đề cập tới một tập các mẫu (*instance* hay *example*) được lưu trong một tệp. Các mẫu thường là các vector thuộc tính có độ dài cố định, thông tin về tên thuộc tính và dãy giá trị của chúng đôi khi cũng được lưu lại như trong từ điển dữ liệu. Một thuật toán học còn sử dụng tập dữ liệu và các thông tin kèm theo tập dữ liệu đó làm đầu vào và đầu ra biểu thị kết quả cuả việc học.

Với so sánh cơ sở dữ liệu thông thường và cơ sở dữ liệu trong học máy như trên, có thể thấy là học máy có khả năng được áp dụng cho cơ sở dữ liệu, bởi vì không phải học trên tập các mẫu mà học trên tệp các bản ghi của cơ sở dữ liệu. Tuy nhiên, phát hiện tri thức trong cơ sở dữ liệu làm tăng thêm các khó khăn vốn đã là điển hình trong học máy và đã vượt quá khả năng của học máy. Trong thực tế, cơ sở dữ liệu thường động, không đầy đủ, bị nhiễu và lớn hơn nhiều so với các tập dữ liệu học máy điển hình. Các yếu tố này làm cho hầu hết các thuật toán học máy trở nên không hiệu quả trong hầu hết các trường hợp. Vì vậy trong khai phá dữ liệu, cần tập trung rất nhiều công sức vào việc vượt qua những vấn đề này trong CSDL.

Phương pháp hệ chuyên gia

Các hệ chuyên gia cố gắng nắm bắt các tri thức thích hợp với một bài toán nào đó. Các kỹ thuật thu thập giúp cho việc lấy tri thức từ các chuyên gia con người. Mỗi phương pháp đó là một cách suy diễn các luật từ các ví dụ và giải pháp đối với bài toán chuyên gia đưa ra. Phương pháp này khác với khai phá dữ liệu ở chỗ các ví dụ của chuyên gia thường ở mức chất lượng cao hơn rất nhiều so với các dữ liệu trong cơ sở dữ liệu, và chúng thường chỉ bao quát được các trường hợp quan trọng. Hơn nữa, các chuyên gia sẽ xác nhận tính giá trị và hữu dụng của các mẫu phát hiện được. Cũng như với các công cụ quản trị cơ sở dữ liệu, ở các phương pháp này đòi hỏi có sự tham gia của con người trong việc phát hiện tri thức.

Phát kiến khoa học

Khai phá dữ liệu rất khác với phát kiến khoa học ở chỗ những khai phá trong cơ sở dữ liệu ít có chủ tâm và có điều khiển hơn. Các dữ liệu khoa học có từ thực nghiệm nhằm loại bỏ một số tác động của các tham số để nhấn mạnh độ biến thiên của một hay một số tham số đích. Tuy nhiên, các cơ sở dữ liệu thương mại thường ghi lại một số lượng thừa thông tin về các dự án của họ để đạt được một số mục đích về mặt tổ chức. Sự dư thừa này có thể là hiển hiện hay ẩn chứa trong các mối quan hệ dữ liệu. Hơn nữa, các nhà khoa học có thể tạo lại các thí nghiệm và có thể tìm ra rằng các thiết kế ban đầu không thích hợp. Trong khi đó, các nhà quản lý cơ sở dữ liệu hầu như không thể xa xỉ đi thiết kế lại các trường dữ liệu và thu thập lại dữ liệu.

Phương pháp thống kê

Mặc dù các phương pháp thống kê cung cấp một nền tảng lý thuyết vững chắc cho các bài toán phân tích dữ liệu nhưng chỉ có tiếp cận thống kê thuần tuý thôi chưa đủ. Thứ nhất, các phương pháp thống kê chuẩn không phù hợp đối với các kiểu dữ liệu có cấu trúc trong rất nhiều cơ sở dữ liệu. Thứ hai, các phương pháp thống kê hoàn toàn bị dữ liệu điều khiển, nó không sử dụng tri thức sẵn có về lĩnh vực. Thứ ba, các kết quả của phân tích thống kê có thể sẽ rất nhiều và khó có thể làm rõ được. Cuối cùng, các phương pháp thống kê cần có sự hướng dẫn của người dùng để xác định phân tích dữ liệu như thế nào và ở đâu.

Sự khác nhau cơ bản giữa khai phá dữ liệu và thống kê là ở chỗ khai phá dữ liệu là một phương tiện được dùng bởi người dùng cuối chứ không phải là các nhà thống kê. Khai phá dữ liệu tự động hóa quá trình thống kê một cách hiệu quả, vì vậy làm nhẹ bớt công việc của người dùng cuối, tạo ra một công cụ dễ sử dụng hơn. Như vậy, nhờ có khai phá dữ liệu, việc dự đoán và kiểm tra rất vất vả trước đây có thể được đưa lên máy tính, được tính, dự đoán và kiểm tra một cách tự động.

I.1.5. Một số thách thức trong ứng dụng và nghiên cứu kỹ thuật khai phá dữ liệu

Việc nghiên cứu và ứng dụng các kỹ thuật khai phá dữ liệu còn gặp nhiều khó khăn, các khó khăn này không phải là không thể giải quyết, song chúng cần được tìm hiểu để có thể phát triển tốt hơn. Những khó khăn điển hình được trình bày dưới đây.

Các vấn đề về cơ sở dữ liệu

Đầu vào chủ yếu của một hệ thống phát hiện tri thức là các dữ liệu thô trong cơ sở dữ liệu. Những vấn đề khó khăn phát sinh trong khai phá dữ liệu chính từ nguyên nhân là dữ liệu trong thực tế thường động, không đầy đủ, lớn và bị nhiễu. Trong những trường hợp khác, người ta không biết cơ sở dữ liệu có chứa các thông tin cần thiết cho việc khai thác hay không và làm thế nào để giải quyết sự dư thừa thông tin không thích hợp này.

- ***Dữ liệu lớn:*** Cho đến nay, các cơ sở dữ liệu với hàng trăm trường và bảng, hàng triệu bản ghi và với kích thước gigabyte đã là chuyện bình thường. Hiện nay đã bắt đầu xuất hiện các cơ sở dữ liệu có kích thước tới tetrabyte. Các phương pháp giải quyết hiện nay là đưa ra một ngưỡng cho cơ sở dữ liệu, lấy mẫu, các phương pháp xấp xỉ, xử lý song song.
- ***Kích thước lớn:*** Không chỉ có số lượng bản ghi mà số các trường trong cơ sở dữ liệu cũng nhiều, vì vậy mà kích thước của bài toán trở nên lớn hơn. Một tập dữ liệu có kích thước lớn sẽ làm tăng không gian tìm kiếm. Hơn nữa, nó cũng làm tăng khả năng một thuật toán khai phá dữ liệu có thể tìm thấy các

mẫu giả. Biện pháp khắc phục là làm giảm kích thước tác động của bài toán và sử dụng các tri thức biết trước để xác định các biến không phù hợp.

- ***Dữ liệu động:*** Đặc điểm cơ bản của hầu hết các cơ sở dữ liệu là nội dung của chúng thay đổi liên tục, dữ liệu có thể thay đổi theo thời gian và việc khai phá dữ liệu bị ảnh hưởng bởi thời điểm quan sát dữ liệu. Việc thay đổi dữ liệu nhanh chóng có thể làm cho các mẫu khai thác được trước đó mất giá trị. Hơn nữa, các biến trong cơ sở dữ liệu của ứng dụng đã cho cũng có thể bị thay đổi, bị xóa hoặc là tăng lên theo thời gian. Vấn đề này được giải quyết bằng các giải pháp nâng cấp các mẫu và coi những thay đổi như là cơ hội để khai thác bằng cách sử dụng nó để tìm kiếm các mẫu bị thay đổi.
- ***Các trường hợp không phù hợp:*** Một đặc điểm quan trọng khác là tính không thích hợp của dữ liệu, nghĩa là mục dữ liệu trở thành không thích hợp với trọng tâm hiện tại của việc khai thác. Một khía cạnh khác đôi khi cũng liên quan đến tính phù hợp là sự có giá trị của một thuộc tính đối với một tập con của cơ sở dữ liệu.
- ***Các giá trị bị thiếu:*** Sự có mặt hay vắng mặt của giá trị các thuộc tính dữ liệu phù hợp có thể ảnh hưởng đến việc khai phá dữ liệu. Trong hệ thống tương tác, sự thiếu vắng dữ liệu quan trọng có thể dẫn tới yêu cầu cho giá trị của nó hoặc kiểm tra để xác định giá trị của nó. Hoặc cũng có thể sự vắng mặt của dữ liệu được coi như một điều kiện, thuộc tính bị mất có thể được coi như một giá trị trung gian và là giá trị không biết.
- ***Các trường bị thiếu:*** Một quan sát không đầy đủ cơ sở dữ liệu có thể làm cho dữ liệu có các giá trị bị xem như có lỗi. Việc quan sát cơ sở dữ liệu phải phát hiện được toàn bộ các thuộc tính có thể dùng để thuật toán khai phá dữ liệu có thể áp dụng để giải quyết bài toán. Giả sử ta có các thuộc tính để phân biệt các tình huống đáng quan tâm. Nếu chúng không làm được điều đó thì có nghĩa là đã có lỗi trong dữ liệu. Đây cũng là vấn đề thường xảy ra trong cơ sở dữ liệu kinh doanh. Các thuộc tính quan trọng có thể sẽ bị thiếu dữ liệu không được chuẩn bị cho việc khai phá dữ liệu.

- ***Độ nhiễu và không chắc chắn:*** Đối với các thuộc tính đã thích hợp, độ nghiêm trọng của lỗi phụ thuộc vào kiểu dữ liệu của các giá trị được phép. Các giá trị của các thuộc tính khác nhau có thể là các số thực, số nguyên, chuỗi, và có thể thuộc vào tập các giá trị định danh. Các giá trị định danh này có thể sắp xếp theo thứ tự bộ phận hoặc đầy đủ, thậm chí có thể có cấu trúc ngữ nghĩa.

  Một yếu tố khác của độ không chắc chắn là tính kế thừa hoặc độ chính xác mà dữ liệu cần có, nói cách khác là độ nhiễu của dữ liệu. Dựa trên việc tính toán trên các phép đo và phân tích có ưu tiên, mô hình thống kê mô tả tính ngẫu nhiên được tạo ra và được sử dụng để định nghĩa độ mong muốn và độ dung sai của dữ liệu. Thường thì các mô hình thống kê được áp dụng theo cách đặc biệt để xác định một cách chủ quan các thuộc tính để đạt được các thống kê và đánh giá khả năng chấp nhận của các giá trị thuộc tính. Đặc biệt là với các kiểu dữ liệu số, sự đúng đắn của dữ liệu có thể là một yếu tố trong việc khai phá. Ví dụ như trong việc đo nhiệt độ cơ thể, ta thường cho phép chênh lệch 0.1 độ. Nhưng việc phân tích theo xu hướng nhạy cảm nhiệt độ của cơ thể lại yêu cầu độ chính xác cao hơn. Để một hệ thống khai thác có thể liên hệ đến xu hướng này để chuẩn đoán thì lại cần có một độ nhiễu trong dữ liệu đầu vào.

- ***Mối quan hệ phức tạp giữa các trường:*** Các thuộc tính hoặc các giá trị có cấu trúc phân cấp, các mối quan hệ giữa các thuộc tính và các phương tiện phức tạp để diễn tả tri thức về nội dung của cơ sở dữ liệu yêu cầu các thuật toán phải có khả năng sử dụng một cách hiệu quả các thông tin này. Ban đầu, kỹ thuật khai phá dữ liệu chỉ được phát triển cho các bản ghi có giá trị thuộc tính đơn giản. Tuy nhiên, ngày nay người ta đang tìm cách phát triển các kỹ thuật nhằm rút ra mối quan hệ giữa các biến này.

Một số vấn đề khác

- ***"Quá phù hợp":*** Khi một thuật toán tìm kiếm các tham số tốt nhất cho một mô hình nào đó sử dụng một tập dữ liệu hữu hạn, nó có thể sẽ bị tình trạng "quá độ" dữ liệu (nghĩa là tìm kiếm quá mức cần thiết gây ra hiện tượng chỉ phù hợp với các dữ liệu đó mà không có khả năng đáp ứng cho các dữ liệu lạ), làm cho mô hình hoạt động rất kém đối với các dữ liệu thử. Các giải pháp khắc phục bao gồm đánh giá chéo (cross-validation), thực hiện theo nguyên tắc nào đó hoặc sử dụng các biện pháp thống kê khác.
- ***Khả năng biểu đạt của mẫu:*** Trong rất nhiều ứng dụng, điều quan trọng là những điều khai thác được phải càng dễ hiểu với con người càng tốt. Vì vậy, các giải pháp thường bao gồm việc diễn tả dưới dạng đồ họa, xây dựng cấu trúc luật với các đồ thị có hướng, biểu diễn bằng ngôn ngữ tự nhiên và các kỹ thuật khác nhằm biểu diễn các tri thức và dữ liệu.
- ***Sự tương tác với người sử dụng và các tri thức sẵn có:*** Rất nhiều công cụ và phương pháp khai phá dữ liệu không thực sự tương tác với người dùng và không dễ dàng kết hợp cùng với các tri thức đã biết trước đó. Việc sử dụng tri thức miền là rất quan trọng trong khai phá dữ liệu. Đã có nhiều biện pháp nhằm khắc phục vấn đề này như sử dụng cơ sở dữ liệu suy diễn để phát hiện tri thức, những tri thức này sau đó được sử dụng để hướng dẫn cho việc tìm kiếm khai phá dữ liệu hoặc sử dụng phân bố và xác suất dữ liệu trước đó như một dạng mã hóa tri thức có sẵn.

I.2. KHAI PHÁ DỮ LIỆU SONG SONG

Hầu hết các thuật toán khai phá dữ liệu đều đòi hỏi số lượng tính toán lớn, chúng cần có khả năng mở rộng được để xử lý lượng dữ liệu lớn hơn và phức tạp hơn [7]. Một số cơ sở dữ liệu vốn liên quan tới Internet và do đó đã sẵn có tính phân tán. Tính toán song song vốn có thể xử lý tốt lượng tính toán lớn trên lượng lớn dữ liệu, vì thế nó là một công cụ tốt để khai phá dữ liệu [8].

I.2.1. Các hệ thống tính toán song song

Có hai cách tăng tốc độ phần cứng máy tính: về mặt công nghệ có thể sử dụng các thiết bị tốc độ cao, về mặt cấu trúc có thể tạo các cấu trúc mới hoặc ghép các hệ sẵn có với nhau theo những cách sáng tạo. Các bộ đa xử lý cung cấp một lựa chọn tốt để nâng cao hiệu suất các hệ thống máy tính bằng cách ghép nối một số bộ xử lý có giá thành thấp. Một hệ thống đa xử lý gồm các bộ vi xử lý chuẩn sẵn có thể đạt tỉ lệ chi phí/hiệu suất tốt hơn một bộ đơn xử lý tốc độ cao dựa trên công nghệ lai. Giá tương đối cao của các hệ đa xử lý có thể được bù đắp bằng cách khai thác chúng như các máy chủ tính toán trong các hệ phân tán. Đa xử lý được dùng để tăng lưu lượng hệ thống bằng cách thực hiện song song một số tiến trình khác nhau của người sử dụng trên các bộ xử lý khác nhau, và tăng tốc ứng dụng bằng cách thực hiện song song một số phần của ứng dụng. Các phần của ứng dụng được song song hóa cần được đồng bộ hóa và trao đổi dữ liệu sau khi hoàn thành mỗi bước tính toán. Sự đồng bộ hóa và truyền thông giữa các bộ xử lý nói chung sẽ làm giảm một phần tốc độ do nó tạm dừng một số tính toán và sử dụng băng thông kết nối hệ thống. Một trong những thách thức về thiết kế hệ đa xử lý là làm sao giảm thiểu các tương tác giữa các bộ xử lý và có một cơ chế hiệu quả để thực hiện việc này khi cần thiết.

I.2.1.1. Tính chất và phân loại các hệ đa xử lý

Các hệ đa xử lý có các ưu điểm chính như: khả năng tính toán và hiệu suất cao, khả năng kháng lỗi, khả năng thích nghi, có thể phát triển theo mô-đun, chuyên môn hoá các chức năng và có tỉ lệ chi phí/ hiệu suất thấp. Với các ưu điểm này, các hệ đa xử lý có thể được thiết kế theo mô-đun và tự điều chỉnh một cách linh hoạt nhằm có được hệ thống tối ưu cho các mục đích cụ thể [10].

Các hệ thống tính toán song song gồm bốn loại chính: xử lý đơn lệnh đơn dữ liệu (SISD), xử lý đơn lệnh đa dữ liệu (SIMD), xử lý đa lệnh đơn dữ liệu (MISD) và

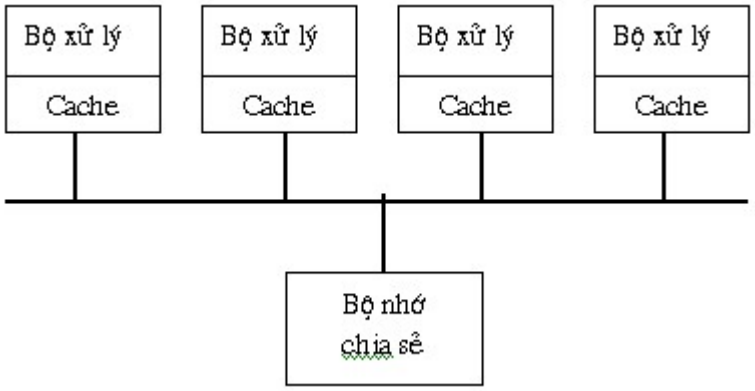
xử lý đa lệnh đa dữ liệu (MIMD). Các bộ đa xử lý thuộc về loại cuối cùng, chúng lại có thể được chia thành các bộ đa xử lý ghép chặt, trong đó tất cả các bộ xử lý đều được truy cập tới bộ nhớ chia sẻ toàn cục, và các bộ đa xử lý ghép lỏng, trong đó các bộ xử lý có bộ nhớ riêng, không có bộ nhớ toàn cục được chia sẻ. Cũng tồn tại các hệ lai trong đó các bộ xử lý có bộ nhớ riêng nhưng vẫn được truy cập vào bộ nhớ chung, thậm chí cả vào bộ nhớ riêng của các bộ xử lý khác. Trong các hệ ghép chặt thì bộ nhớ toàn cục chính là cơ sở truyền thông và đồng bộ hóa giữa các bộ xử lý; trong các hệ ghép lỏng, sự truyền thông này được thực hiện bởi cơ chế truyền thông báo. Độ trễ lớn trong các đường truyền thông ngày nay đã được giảm bớt nhờ sự phát triển của sợi quang học và các công nghệ mạng tốc độ cao.

I.2.1.2. Các cách kết nối trong hệ đa xử lý

Các kiểu cấu trúc cơ bản thường gặp của các hệ đa xử lý gồm các hệ hướng đường truyền, các hệ kết nối chéo, cấu trúc siêu khối, và các hệ dựa trên chuyển mạch nhiều tầng [11].

- *Các hệ hướng đường truyền* là một trong những cách tạo hệ đa xử lý đơn giản nhất bằng cách dùng một đường truyền chung để nối các bộ xử lý và các bộ nhớ. Trong sơ đồ, này các bộ xử lý có thể có hoặc không có các bộ nhớ riêng, các thiết bị nhập/xuất có thể được gắn với các bộ xử lý hoặc với đường truyền chung, và bản thân bộ nhớ chia sẻ cũng thường được tạo dưới dạng một số vùng nhớ gắn liền với đường truyền chung. Đường truyền chia sẻ và bộ nhớ chia sẻ là hai yếu tố chính của hệ hướng đường truyền. Có thể dùng các bộ nhớ truy cập nhanh (cache) để giảm tải cho đường truyền chung, chúng có thể được gắn với bộ nhớ chung hoặc với mỗi bộ xử lý. Cách thứ hai thường được dùng hơn, tuy nhiên việc duy trì tính nhất quán giữa các bản sao vật lý của dữ liệu được lưu trong cache thường làm tăng lưu lượng đường truyền, làm giảm ích lợi của việc sử dụng cache riêng cho mỗi bộ xử lý. Khó khăn này sẽ được giảm bớt nếu dùng đường truyền rộng hoặc dùng giao thức phân

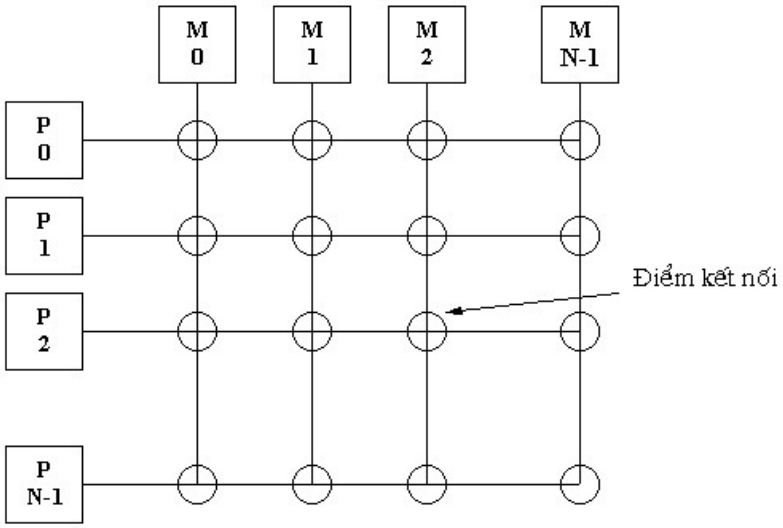
tách yêu cầu/đáp ứng để các yêu cầu và đáp ứng về bộ nhớ được xử lý như những nhiệm vụ khác nhau, sau khi một bộ xử lý yêu cầu một vùng nhớ, đường truyền có thể được dành cho các bộ xử lý khác dùng trong thời gian bộ nhớ tìm và tập hợp đủ các thành phần liên quan. Tuy khả năng mở rộng của các hệ đa xử lý hướng đường truyền bị hạn chế do sự cạnh tranh của đường truyền chia sẻ và bộ nhớ chia sẻ, cách tiếp cận này vẫn được sử dụng rộng rãi trong nhiều ứng dụng thương mại do cách thực hiện đơn giản của nó.



Cấu trúc hệ đa xử lý hướng đường truyền

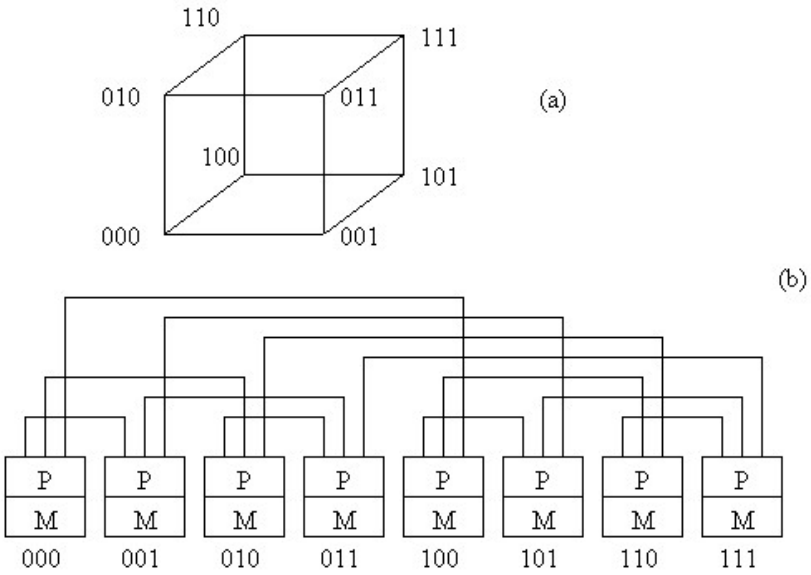
- *Các hệ kết nối chéo* cho phép N bộ xử lý đồng thời truy cập vào N bộ nhớ với điều kiện tại mỗi thời điểm, mỗi bộ xử lý truy cập vào một bộ nhớ khác nhau, bản thân hệ này không có tính cạnh tranh và là cách đối lập với hệ thống hướng đường truyền. Sự chậm trễ giữa một bộ xử lý và một bộ nhớ chỉ xảy ra ở điểm kết nối, trong trường hợp các bộ xử lý không có bộ nhớ riêng thì đây là hệ thống đa xử lý truy cập bộ nhớ không đổi. Sự xếp đặt dữ liệu một cách hợp lý sẽ tránh được tranh chấp có thể xảy ra khi nhiều hơn một bộ xử lý cùng cố truy cập một bộ nhớ, song nó cũng không tránh được tranh chấp khi có nhiều bộ xử lý cùng cố truy cập một vị trí trong cùng một bộ nhớ. Vì thế sơ đồ này cho phép song song hóa mức cao giữa các công việc không liên quan, tuy nhiên sự đồng bộ hóa giữa các tiến trình hoặc các bộ xử lý trên bộ nhớ chia sẻ sẽ dễ gây ra tranh chấp về bộ nhớ. Do mô hình này cần N^2

điểm kết nối để nối *N* bộ nhớ với *N* bộ xử lý, độ phức tạp sẽ tăng gấp 4 theo số phần tử, làm hạn chế khả năng mở rộng của sơ đồ này.



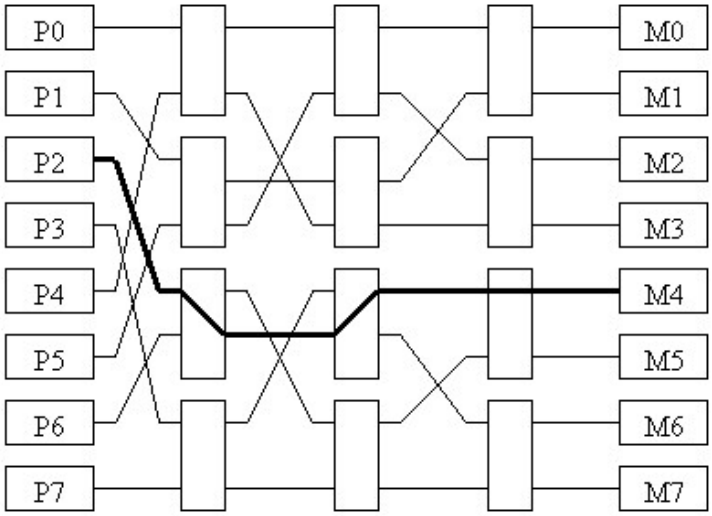
Kết nối chéo

- *Hệ thống siêu khối* giải quyết được bài toán về khả năng mở rộng và giá thành bằng các liên kết có độ phức tạp tăng theo hàm lô-ga-rit của số nút *N* và khoảng cách tối đa giữa các nút là log_2N. Cấu trúc này là đệ quy theo nghĩa các siêu khối nhiều chiều chứa các siêu khối ít chiều hơn như những tập con của mình. Hệ thống này sử dụng các bộ nhớ riêng cho mỗi bộ xử lý và truyền thông báo để truyền thông và đồng bộ hóa giữa các bộ xử lý. Ngoài chức năng xử lý, mỗi nút còn thực hiện các giao thức truyền thông, nó cũng định tuyến và chuyển tiếp các thông báo để tạo đường truyền thông gián tiếp giữa các nút từ xa kết nối trực tiếp với nó. Các thiết bị nhập/xuất có thể được gắn cục bộ vào mỗi nút, để tăng băng thông nhập/xuất người ta có thể dùng một số nút nhập/xuất chuyên dụng như các luồng và kho chứa dữ liệu nhập/xuất cho một nhóm các nút.



Siêu khối 8 nút: (a) mô hình 3 chiều
(b) các kết nối giữa các bộ xử lý

- *Các hệ thống dựa trên bộ chuyển mạch nhiều tầng* để kết nối các bộ xử lý và các bộ nhớ. Kiểu tổng quát nhất của cách tiếp cận này cung cấp các liên kết giữa *N* đầu vào và *N* đầu ra, nó có log_2N tầng, mỗi tầng gồm *N* liên kết tới *N/2* hộp trao đổi. Mạng chuyển mạch có thể nối bất kỳ đầu vào với đầu ra nào bằng cách tạo các kết nối thích hợp tại mỗi tầng. Việc định tuyến trong mạng này thường là cố định và được thực hiện nhờ các thẻ địa chỉ đích mà bên gửi gửi kèm với yêu cầu kết nối. Chuyển mạch nhiều tầng có thể đồng thời kết nối tất cả các đầu vào và tất cả các đầu ra với điều kiện không có hai bộ xử lý nào đồng thời cố truy cập cùng một mô-đun bộ nhớ; nếu không thì tranh chấp tại cả mô-đun bộ nhớ và trong mạng chuyển mạch sẽ xảy ra và gây tắc nghẽn. Để tránh điều này xảy ra, người ta thêm phần cứng phụ trợ vào bản thân hệ thống chuyển mạch; hệ thống có thêm khả năng xử lý tại các điểm chuyển mạch được gọi là mạng kết hợp.



Mạng chuyển mạch nhiều tầng và đường nối P2-M4

I.2.2. Các chiến lược khai phá dữ liệu song song

Khai phá dữ liệu song song đòi hỏi phân chia công việc để các bộ xử lý có thể thực hiện phần công việc của mình, nhằm đạt kết quả cuối cùng một cách nhanh nhất. Vấn đề là ở chỗ phân chia công việc như thế nào, ta có thể phân chia việc tính toán, cũng có thể phân chia việc truy cập tới dữ liệu, và giảm thiểu sự truyền thông giữa các bộ xử lý trong khi thực hiện. Trong các ứng dụng khai phá dữ liệu, ta cần giảm thiểu nguồn tài nguyên được dùng để sinh các khái niệm có vẻ có giá trị địa phương, dựa trên lượng hạn chế dữ liệu có sẵn tại mỗi bộ xử lý, nhưng không có giá trị toàn phần. Có ba chiến lược để song song hóa các thuật toán khai phá dữ liệu [13, 15], đó là:

- *Tìm kiếm độc lập:* Mỗi bộ xử lý được truy cập tới toàn bộ tập dữ liệu, nhưng chỉ tập trung vào một phần không gian tìm kiếm, bắt đầu từ một điểm được chọn ngẫu nhiên.

Cách này phù hợp với các bài toán mà kết quả cần tìm là một giải pháp tối ưu, tuy nhiên nó đòi hỏi mỗi bộ xử lý phải truy cập toàn bộ tập dữ liệu, khiến tốc độ bị chậm lại.

- *Song song hóa một thuật toán khai phá dữ liệu tuần tự:* Tập các khái niệm được phân chia giữa các bộ xử lý, mỗi bộ xử lý kiểm tra toàn bộ tập dữ liệu để kiểm tra xem các khái niệm cục bộ của nó có đúng trên phạm vi toàn cục không. Do việc tạo khái niệm mới thường đòi hỏi phải biết các khái niệm nhỏ hơn hay đơn giản hơn nào là đúng, các bộ xử lý phải thường xuyên trao đổi thông tin về các khái niệm của chúng. Một cách khác là tập dữ liệu được phân chia theo các cột, mỗi bộ xử lý tìm ra các khái niệm trên các cột mà chúng giữ.

Theo cách này cũng cần thường xuyên trao đổi thông tin để xác định xem các khái niệm cục bộ nào có thể ghép lại thành khái niệm đúng trên toàn cục.

- *Lặp lại một thuật toán khai phá dữ liệu tuần tự:* Mỗi bộ xử lý làm việc trên một phần của tập dữ liệu (theo hàng), và thực hiện thuật toán tuần tự. Do chỉ có một phần thông tin, nó tạo nên các khái niệm đúng cục bộ, nhưng có thể không đúng trên toàn cục - các khái niệm xấp xỉ. Các bộ xử lý trao đổi các khái niệm xấp xỉ này, hoặc các số liệu về chúng, để kiểm tra xem chúng có đúng trên toàn cục không. Khi làm như vậy mỗi bộ xử lý học được về những phần dữ liệu mà chúng không nhìn thấy.

Cách này có hai ưu điểm đáng chú ý: tập dữ liệu được phân chia giúp cho chi phí truy cập được chia đều cho các bộ xử lý, và dữ liệu cần được trao đổi giữa các pha thường nhỏ hơn rất nhiều so với bản thân các khái niệm, vì thế không tốn nhiều chi phí cho truyền thông.

I.2.3. Các mô hình chi phí

Để xem một kỹ thuật khai phá dữ liệu có nên được thực hiện song song hay không, ta cần xác định độ phức tạp của nó, dựa trên mô hình chi phí thực tế. Chi phí cho một chương trình song song gồm hai phần: lượng tính toán mà nó thực hiện và truyền thông giữa các bộ xử lý. Chi phí này có thể được tính chính xác bởi [15]:

$$C = \underset{\text{các bộ xử lý}}{Max}\, w_i + \underset{\text{các bộ xử lý}}{Max}\, h_i g$$

với hàm Max được tính trên tập toàn bộ các bộ xử lý; wi là số các lệnh được thực hiện bởi bộ xử lý i; hi là số byte được gửi hoặc nhận bởi bộ xử lý i; và g là khả năng truyền của mạng, tính bằng số đơn vị thời gian để truyền mỗi byte.

Xét một thuật toán khai phá dữ liệu tuần tự: giả sử tập dữ liệu gồm *n* hàng có kích thước *m* và các thuật toán tạo ra *s* khái niệm. Mỗi thuật toán gồm một số *k_s* lần lặp để tạo ra các khái niệm lớn hơn từ các khái niệm nhỏ hơn. Độ phức tạp được cho bởi công thức:

$$\cos t\_s = k\_s[STEP(nm, s) + ACCESS(nm)]$$

với *STEP* là chi phí cho một bước lặp, *ACCESS* là chi phí xử lý dữ liệu.

Dưới đây trình bày các chiến lược song song hoá các thuật toán tuần tự nêu trên [15].

I.2.3.1. Xấp xỉ các khái niệm

- Chia dữ liệu làm p tập con, mỗi tập trên một bộ xử lý.
- Thực hiện thuật toán tuần tự (hoặc cải biên của thuật toán trên mỗi tập con).
- Trao đổi thông tin mỗi bộ xử lý học được với các bộ xử lý khác.
- Lặp lại.

Giá của thuật toán xấp xỉ khái niệm có dạng:

$$cost\_a = k\_a * \left[STEP\left(\frac{nm}{p}, r\right) + ACCESS\left(\frac{nm}{p}\right) + rpg + RES\ (rp) \right]$$

trong đó *r* là kích thước của các khái niệm xấp xỉ được tạo bởi mỗi bộ xử lý; *rpg* là tổng chi phí của việc trao đổi các khái niệm xấp xỉ này giữa các bộ xử lý; *RES(rp)* là chi phí tính toán của việc sắp xếp các khái niệm xấp xỉ này để tính được các xấp xỉ tốt hơn cho lần lặp sau. Có thể giả thiết rằng:

$$STEP\left(\frac{nm}{p}, r\right) = \frac{STEP\ (nm, r)}{p} \quad \text{và} \quad ACCESS\left(\frac{nm}{p}, r\right) = \frac{ACCESS\ (nm, r)}{p}$$

$$\Rightarrow \qquad cost\_a = \frac{cost\_s}{p} + k\_a(rpg + RES\ (rp))$$

Như vậy, ta tăng tốc được *p* lần, không kể tổng chi phí, với điều kiện *k_s* và *k_a* có kích thước có thể so sánh được. Trao đổi kết quả thường xuyên có thể cải thiện độ chính xác nhanh hơn với một số thuật toán, ví thế *k_a* << *k_s*. Điều này cho ta sự tăng tốc gấp đôi:

$$cost\_a = \frac{k\_a}{k\_s} * \frac{cost\_s}{p} + (tổng\_chi\_phí)$$

I.2.3.2. Phân chia các khái niệm

- Chia tập dữ liệu làm *p* tập con dựa trên các thuộc tính
- Thực hiện một biến đổi cụ thể của một thuật toán khai phá dữ liệu trên tập con này để lấy ra các khái niệm hoặc các tham số mô hình chỉ ứng với tập con các thuộc tính này.
- Lặp lại.
- Kết hợp các khái niệm cục bộ hợp lý với nhau để tạo ra các khái niệm mô tả toàn bộ tập dữ liệu.

Giá của thuật toán khái niệm bộ phận có dạng:

$$cost\_p = k\_p\left[SPECIAL\left(n,\frac{m}{p},r\right)+ACCESS\left(n,\frac{m}{p}\right)+EXCH\ (n,m,r,p)*g+RES\ (n,m,r,p)\right]$$

với hàm *SPECIAL* tính độ phức tạp của một bước của thuật toán; *EXCH* là chi phí trao đổi các khái niệm cục bộ.

I.2.3.3. Các chiến lược tìm kiếm độc lập

Không phân chia dữ liệu. Thay vì thực hiện cùng một thuật toán p lần, sử dụng một số kỹ thuật ngẫu nhiên để hướng thuật toán sang một phần khác của không gian tìm kiếm các khái niệm.

Giá của một thuật toán tìm kiếm độc lập có dạng:

$$cost\_i = k\_i * [STEP\ (nm,s)+ACCESS\ (nm)]+s*g+s$$

Rõ ràng cách tiếp cận này chỉ có ý nghĩa nếu ta có lý do để giả sử rằng $k\_i = \frac{k\_s}{p}$.

I.2.3.4. So sánh các chiến lược

Giá của các chiến lược thực hiện khác nhau ở trên phụ thuộc vào giá trị của các giá trị *k_a*, *k_p*, *k_i*. Tuy nhiên, giả sử rằng chúng không tồi hơn *k_s* thì ta có thể xếp hạng các cách song song hóa như sau:

Thuật toán khái niệm xấp xỉ tốt hơn thuật toán khái niệm cục bộ, tốt hơn thuật toán tìm kiếm độc lập. Các kỹ thuật kết hợp từng phần tạo nhiều khái niệm mà chúng không thể là một bộ phận của tập khái niệm lớn hơn. Vì thế các tập tạo bởi mỗi thuật toán độc lập là lớn hơn nhiều so với các tập có được sau khi giải quyết, có nghĩa là mỗi thuật toán đã làm một việc vô ích. Cũng vậy, tìm kiếm độc lập không lợi dụng được một thực tế là các bộ xử lý khác cũng đã có thể phát hiện ra các tri thức có ích mà có thể tỉa bớt sự tìm kiếm ở bộ xử lý này, và nó lại thực hiện công việc vô ích.

KẾT LUẬN CHƯƠNG 1

Chương này đã trình bày những nội dung chính yếu về khai phá dữ liệu, hệ thống tính toán song song và áp dụng trong khai phá dữ liệu song song. Việc phát hiện tri thức (những mẫu, những xu hướng tiềm ẩn và hữu ích) trong cơ sở dữ liệu là rất cần thiết và là một quá trình gồm nhiều giai đoạn, trong đó giai đoạn khai phá dữ liệu là một giai đoạn chính yếu nhất (*mục 1.1.1*). Trong các cơ sở dữ liệu đồ sộ, các phương pháp khai phá dữ liệu (điển hình là phân lớp, phân cụm) cho phép phát hiện được các mẫu tiềm ẩn và đánh giá giá trị của chúng một cách tự động trong một khoảng thời gian nhanh nhất để hỗ trợ cho người sử dụng.

Thực hiện khai phá dữ liệu trên các hệ thống với dữ liệu lớn, trên các hệ thống phân tán, việc nghiên cứu và đề xuất các thuật toán khai phá dữ liệu song song trong các mô hình song song là rất có ý nghĩa. Tùy thuộc vào cơ sở dữ liệu thực tế, việc lựa chọn cách thức song song để song song hóa thuật toán khai phá dữ liệu tuần tự là rất quan trọng *(mục 1.2.2)*, nó ảnh hưởng trực tiếp tới giá thành thực hiện việc khai phá dữ liệu. Một số mô hình chi phí hình thức cho khai phá dữ liệu song song đã được tổng kết *(mục 1.2.3)*.

CHƯƠNG II. LUẬT KẾT HỢP THEO TIẾP CẬN LÝ THUYẾT TẬP THÔ

II.1. KHÁI NIỆM LUẬT KẾT HỢP VÀ MỘT SỐ CÔNG NGHỆ PHÁT HIỆN

II.1.1. Luật kết hợp

Phát hiện luật kết hợp là sự khai phá dữ liệu không được định hướng hoặc không có giám sát trên dữ liệu có độ dài thay đổi, nó cho ra các kết quả rõ ràng và dễ hiểu. Mục đích của khai phá luật kết hợp là tìm tất cả các tập con các đối tượng hoặc thuộc tính xuất hiện thường xuyên trong nhiều giao dịch hoặc bản ghi trong cơ sở dữ liệu, thêm vào đó là rút ra các luật về một tập con đối tượng có ảnh hướng tới sự xuất hiện của tập con các đối tượng khác như thế nào [15].

Mặc dù phát hiện luật kết hợp có cách đặt bài toán đơn giản, nó đòi hỏi lượng tính toán và truy xuất dữ liệu rất lớn. Khi dữ liệu tăng lên cả về số hướng (số các thuộc tính) và kích thước (số giao dịch), một trong những tính chất cần thiết của phát hiện luật kết hợp là khả năng mở rộng được: khả năng xử lý kho dữ liệu rất lớn. Các thuật toán tuần tự không thể cho khả năng này trong các cơ sở dữ liệu lớn. Vì vậy ta phải dựa vào tính toán song song và phân tán hiệu suất cao.

Tập phổ biến là cơ sở để tạo các luật kết hợp [4]. Chúng ta xem xét một ví dụ khai phá luật kết hợp. Cho một tập các thuộc tính I = $\{I_1, I_2,..., I_m\}$, một giao dịch T được định nghĩa là một tập con bất kỳ các thuộc tính trong I. Giả sử cơ sở dữ liệu D là một tập n giao dịch, mỗi giao dịch được gán một định danh giao dịch duy nhất *TID*. Giao dịch T là hỗ trợ một tập $X \subseteq I$ nếu nó chứa tất cả các thuộc tính trong X, tức là $X \subseteq T$. Độ hỗ trợ của một tập thuộc tính X, ký hiệu $\sigma(X)$, là tỉ lệ của tất cả các giao dịch trong D hỗ trợ X.

Định nghĩa 2.1 ***(Tập phổ biến)***

Tập $X \subseteq I$ được gọi là tập phổ biến nếu có $\sigma(X) \geq s_{min}$ với s_{min} là độ hỗ trợ tối thiểu cho trước.

Một tập X có lực lượng $k = |X|$ được gọi là *k-itemset*. Có ba tính chất quan trọng của các tập phổ biến, đó là:

- Nếu $A \subseteq B$ với A, B là các tập thuộc tính thì $\sigma(A) > \sigma(B)$, bởi tất cả các giao dịch trong D hỗ trợ B thì đều phải hỗ trợ A.
- Tập cha của một tập không phổ biến là tập không phổ biến: Nếu tập thuộc tính A không đủ độ hỗ trợ, tức là $\sigma(A) \leq s_{min}$ thì mọi tập B chứa A cũng sẽ không phổ biến, bởi vì $\sigma(B) \leq \sigma(A) \leq s_{min}$.
- Tập con của tập phổ biến là tập phổ biến: Nếu tập thuộc tính B là phổ biến trong D, tức là $\sigma(B) \geq s_{min}$, thì mọi tập con A của B cũng sẽ là phổ biến, bởi $\sigma(A) \geq \sigma(B) \geq s_{min}$.

Một tập phổ biến là cực đại nếu nó không là tập con của bất kỳ tập phổ biến nào khác. Với khái niệm và các tính chất nêu trên của tập phổ biến, người ta đưa ra khái niệm luật kết hợp như sau đây.

Định nghĩa 2.2 ***(Luật kết hợp)***

Một luật kết hợp là một biểu thức $R: X \rightarrow Y$, với X và Y là các tập thuộc tính không giao nhau $X \cap Y = \varnothing$ và $Y \neq \varnothing$.

Định nghĩa 2.3 ***(Độ hỗ trợ và độ tin cậy của luật)***

Đỗ hỗ trợ của luật là xác suất của một giao dịch chứa cả X và Y: $\sigma(X \cup Y)$. Độ tin cậy của một luật là xác suất có điều kiện để một giao dịch chứa Y, nếu nó đã chứa X, và được tính bởi:

$$\alpha(R) = p(Y \subseteq T \mid X \subseteq T) = \frac{p(Y \subseteq T \wedge X \subseteq T)}{p(X \subseteq T)} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Độ hỗ trợ của một luật là tần suất nó có thể xảy ra, trong khi độ tin cậy của luật cho biết luật đó đáng tin ra sao. Một luật là thích hợp nếu nó có đủ độ hỗ trợ và độ tin cậy: $\sigma(R) \geq s_{min}$ (luật phổ biến) và $\alpha(R) \geq c_{min}$ (luật mạnh), điều này chỉ xảy ra nếu cả vế trái và vế phải của luật đó là các tập phổ biến.

Phát hiện luật kết hợp liên quan tới việc tìm ra tất cả các luật kết hợp trong cơ sở dữ liệu có độ hỗ trợ > s_{min} và có độ tin cậy > c_{min} (các luật phổ biến và mạnh). Công việc này gồm hai bước:

1. Tìm tất cả các tập thuộc tính phổ biến có độ hỗ trợ tối thiểu. Không gian tìm kiếm để liệt kê tất cả các tập thuộc tính phổ biến là 2^m, với m là số thuộc tính. Tuy nhiên, nếu ta giả sử chiều dài giao dịch là có giới hạn, thì có thể chỉ ra rằng phát hiện luật kết hợp về cơ bản là tuyến tính với kích thước của cơ sở dữ liệu.
2. Tạo các luật mạnh có độ tin cậy tối thiểu từ các tập thuộc tính phổ biến. Ta tạo và thử độ tin cậy của tất cả các luật có dạng $X \backslash Y \rightarrow Y$, với $Y \subset X$ và X phổ biến. Vì ta phải xét mỗi tập con của X như là vế phải của luật, độ phức tạp của bước tạo luật là $O(r.2^l)$, với r là số tập thuộc tính phổ biến, l là kích thước của tập phổ biến lớn nhất.

Các tính chất của luật kết hợp:

- *Không có phép hợp các luật:* Nếu $X \rightarrow Z$ và $Y \rightarrow Z$, không có nghĩa là $X \cup Y \rightarrow Z$. Xét trường hợp $X \cap Y = \varnothing$, một giao dịch trong D hỗ trợ Z khi và chỉ khi nó hỗ trợ hoặc X, hoặc Y. Độ hỗ trợ của $X \cup Y$ là bằng 0, và do đó độ tin cậy của $X \cup Y \rightarrow Z$ là bằng 0%.

- *Phép tách các luật:* Nếu $X \cup Y \rightarrow Z$ thích hợp, các luật $X \rightarrow Z$ và $Y \rightarrow Z$ có thể không thích hợp. Ví dụ trong trường hợp Z chỉ xuất hiện khi cả X và Y xuất hiện, tức là $\sigma(X \cup Y) = \sigma(Z)$, nếu X và Y có độ hỗ trợ khá lớn so với $X \cup Y$ thì hai luật tạo thành sẽ không có đủ độ tin cậy. Trường hợp ngược lại: $X \rightarrow Y \cup Z \Rightarrow X \rightarrow Y \wedge X \rightarrow Z$ lại đúng, bởi $\sigma(XY) \geq \sigma(XYZ)$ và $\sigma(XZ) \geq \sigma(XYZ)$, do đó độ hỗ trợ và độ tin cậy của luật nhỏ hơn đều tăng so với luật ban đầu.
- *Không có tính chất bắc cầu:* Nếu $X \rightarrow Y$ và $Y \rightarrow Z$, ta không thể suy ra $X \rightarrow Z$. Ví dụ trong trường hợp $T(X) \subset T(Y) \subset T(Z)$, với $T(X)$ là tập các giao dịch hỗ trợ X, ... và độ tin cậy tối thiểu là c_{min}. Giả sử $\alpha(X \rightarrow Y) = \alpha(Y \rightarrow Z) = c_{min}$, dựa trên các giá trị độ hỗ trợ tương đối ta có $\alpha(X \rightarrow Z) = c^2_{min} < c_{min}$ (vì $c_{min} < 1$), như thế $X \rightarrow Z$ không có đủ độ tin cậy và do đó không thích hợp.

II.1.2. Một số công nghệ phát hiện luật kết hợp tuần tự [16]

Không gian tìm kiếm luật kết hợp tuần tự có thể được thiết đặt theo những cách dưới đây [17].

- *Tìm kiếm từ dưới lên/ Tìm kiếm lai*

Trong phát hiện luật kết hợp có sử dụng quan hệ tập con $\subseteq$ định nghĩa một thứ tự bộ phận trên tập các itemset. Quan hệ này là đơn điệu so với độ hỗ trợ $\sigma(X)$. Thuật toán phát hiện luật kết hợp khác với cách tìm kiếm trong mạng các itemset kết nối bởi quan hệ tập con. Hầu hết các tiếp cận sử dụng cách tìm kiếm theo mức hoặc tìm-từ-dưới-lên trong mạng để liệt kê các itemset phổ biến. Nếu dự đoán là có itemset dài, cách tiếp cận trên-xuống nguyên thủy có thể được ưa dùng hơn. Người ta cũng dùng cách tìm kiếm lai, kết hợp cả tìm-từ-trên-xuống và tìm-từ-dưới-lên.

- *Tạo ứng viên theo cách ngẫu nhiên/ Tạo ứng viên đầy đủ*

Các thuật toán phát hiện luật kết hợp có thể khác nhau trong cách tạo ứng viên mới. Một tìm kiếm đầy đủ đảm bảo rằng ta có thể tạo và thử tất cả các tập con phổ biến. Ở đây, đầy đủ không có nghĩa là tìm đến kiệt sức, ta có thể tỉa bớt để hạn chế các nhánh vô ích trong không gian tìm kiếm. Trong cách tạo phỏng đoán, tính đầy đủ bị mất đi cho mục đích tăng tốc. Tại mỗi bước, nó chỉ kiểm tra một số hạn chế các "nhánh tốt". Cũng có thể tìm kiếm ngẫu nhiên để định vị itemset phổ biến cực đại.

- *Liệt kê tất cả các itemset/ Liệt kê các itemset phổ biến cực đại*

Các thuật toán phát hiện luật kết hợp khác nhau phụ thuộc vào việc chúng tạo ra tất cả các tập con phổ biến hay chỉ một số tập con phổ biến cực đại. Xác định các tập con cực đại là nhiệm vụ cốt lõi, vì việc rà quét lại cơ sở dữ liệu có thể tạo ra tất cả các tập con khác. Tuy nhiên, đa số các thuật toán đều liệt kê tất cả các tập con phổ biến.

- *Trình bày dữ liệu theo hàng/theo cột*

Hầu hết các thuật toán phát hiện luật kết hợp đều sử dụng cách trình bày dữ liệu theo hàng ngang, lưu mỗi định danh giao dịch của khách cùng các mục có trong giao dịch đó. Một số phương pháp cũng dùng cách thể hiện dữ liệu theo chiều dọc, kết hợp với mỗi mục X một danh sách các định danh giao dịch chứa nó.

i. ***Thuật toán Apriori -*** *do Rakesh Agrawal và cộng sự đề xuất*

Đây là một trong các thuật toán phát hiện luật kết hợp tốt nhất. Nó cũng là nền tảng cho hầu hết các thuật toán song song. Apriori sử dụng cách tìm kiếm đầy đủ từ dưới lên trong dữ liệu trình bày theo chiều ngang và liệt kê tất cả các itemset phổ biến. Là một thuật toán lặp, Apriori đếm các itemset có chiều dài cụ thể trong cơ sở dữ liệu. Quá trình bắt đầu với việc duyệt tất cả các giao dịch trong cơ sở dữ

liệu và tính các itemset phổ biến. Tiếp theo, tạo một tập các ứng viên 2-itemset phổ biến từ các itemset phổ biến. Một lần duyệt cơ sở dữ liệu nữa để tính độ hỗ trợ của chúng. Các 2-itemset phổ biến được duy trì cho lần sau. Quá trình lặp lại tới khi liệt kê hết các itemset phổ biến. Thuật toán có 3 bước chính:

- Tạo các ứng viên có độ dài k từ các (k-1)-ietmset phổ biến bằng cách tự kết hợp trên F_{k-1}
- Tỉa bớt các ứng viên có ít nhất một tập con không phổ biến
- Duyệt tất cả các giao dịch để có độ hỗ trợ của các ứng viên. Apriori lưu các ứng viên trong một cây băm (hash tree) để đếm nhanh độ hỗ trợ. Trong một cây băm, các itemset được lưu tại các lá, các nút trong chứa các bảng băm (trộn bởi các mục) đẻ định hướng tìm kiếm các ứng viên.

ii. Thuật toán tỉa và băm động *(Dynamic Hashing & Pruning - DHP) - do Jong Soo Park và cộng sự đề xuất*

Thuật toán DHP mở rộng cách tiếp cận Apriori bằng cách dùng bảng trộn để tính trước độ hỗ trợ xấp xỉ cho các 2-itemset trong quá trình lặp. Lần lặp thứ hai chỉ cần tính các ứng viên trong các phần tử băm có độ hỗ trợ tối thiểu. Kỹ thuật dùng hàm băm này có thể loại đi rất tốt những cặp ứng viên mà cuối cùng sẽ là không phổ biến.

iii. Thuật toán phân hoạch *(Partition) - do Ashok và cộng sự đề xuất*

Thuật toán này phân chia hợp lý cơ sở dữ liệu theo chiều ngang thành các phần không giao nhau. Mỗi phần được đọc và tạo ra cho mỗi item các danh sách theo hàng dọc các định danh giao dịch có chứa item đó (tidlist). Sau đó tìm các itemset phổ biến địa phương qua phần giao của các tidlist. Các itemset phổ biến tại mỗi phần sẽ tập hợp lại để tạo một tập các ứng viên toàn phần. Thuật toán duyệt lần thứ hai qua tất cả các phần và có được con số toàn cục của mọi ứng viên qua phần giao của các tidlist.

iv. Các thuật toán SEAR & SPEAR *- do Andreas Muller đề xuất*

Thuật toán SEAR (Sequential Efficial Association Rules - Phát hiện tuần tự luật kết hợp một cách hiệu quả) giống hệt Apriori, ngoại trừ việc nó lưu các ứng viên trong một cây tiền tố thay vì một cây băm. Trong một cây tiền tố, mỗi cạnh được gán nhãn bởi các tên thuộc tính, các tiền tố phổ biến được biểu diễn bởi các nhánh cây, và các hậu tố duy nhất được lưu tại các lá. Ngoài ra, SEAR dùng một cách tối ưu hóa gộp nhiều lần duyệt, trong đó nó tìm ứng viên cho nhiều lượt nếu các ứng viên đó vừa trong bộ nhớ.

Thuật toán SPEAR (SEAR with Partition technique) tương tự với SEAR nhưng nó dùng kỹ thuật phân hoạch, nó là bản sao của SEAR nhưng không dùng định danh giao dịch. SPEAR dùng dữ liệu định dạng theo hàng ngang, nó duyệt hai lần: trước hết tập trung vào các itemset phổ biến tiềm năng, sau đó tính độ hỗ trợ toàn phần của chúng.

Mục tiêu của Muller là đánh giá những lợi ích nội tại của việc phân hoạch, bất kể định định dạng dữ liệu được dùng. Ông kết luận rằng phân hoạch không giúp được gì do phải xứ lý thêm nhiều phân hoạch và do phân hoạch tìm ra nhiều itemset phổ biến địa phương nhưng không phổ biến toàn phần. SEAR ưu việt hơn do nó thực hiện cả việc gộp các lần duyệt.

v. Thuật toán đếm itemset động *(Dynamic Itemset Counting - DIC) do Sergey Brin và cộng sự đề xuất*

Đây là sự tổng quát hóa của thuật toán Apriori. Dữ liệu được chia làm *p* phần có kích thước bằng nhau để mỗi phần vừa trong bộ nhớ. Với phần 1, DIC tập hợp độ hỗ trợ của từng item. Các item phổ biến địa phương (chỉ trong phần này) tạo nên các ứng viên ứng viên 2-itemset. Sau đó DIC đọc phần 2, có độ hỗ trợ của tất cả các ứng viên hiện tại - tức là các item đơn lẻ và các ứng viên 2-itemset. Quá trình này lặp lại

cho các phần còn lại. DIC bắt đầu đếm số ứng viên k-itemset trong khi xử lý phần k trong lần duyệt cơ sở dữ liệu lần đầu tiên. Sau khi xử lý hết phần cuối cùng p, DIC quay trở lại phần 1. Độ hỗ trợ toàn phần của ứng viên được tính mỗi khi quá trình quay lại và đạt đến phần nơi nó được tính lần đầu. DIC có hiệu quả trong việc giảm số lần quét cơ sở dữ liệu nếu hầu hết các phần là đồng nhất (có sự phân bố các itemset phổ biến giống nhau). Nếu dữ liệu không đồng nhất, DIC có thể tạo ra nhiều số liệu sai - tức các itemset phổ biến địa phương nhưng không phổ biến toàn phần - và duyệt cơ sở dữ liệu nhiều hơn Apriori. DIC đưa ra một kỹ thuật phân hoạch ngẫu nhiên để giảm độ lệch của các phần dữ liệu.

vi. ***Các thuật toán Eclat, MaxEclat, Clique, MaxClique -*** *do Mohammed J. Zaki và cộng sự đề xuất*

Đây là một cách thiết kế hoàn toàn khác mô tả các thuật toán dựa trên các lớp tương đương. Các phương pháp này sử dụng định dạng dữ liệut theo cột dọc, tìm kiếm đầy đủ và kết hợp giữa cách tìm kiếm lai và tìm kiếm từ dưới lên, chúng tạo ra một hỗn hợp các itemset phổ biến cực đại và không cực đại. Lợi thế chính của việc dùng định dạng dữ liệu theo cột dọc là ta có thể xác định độ hỗ trợ của bất kỳ k-itemset nào, đơn giản bằng cách giao các danh sách định danh giao dịch tidlist của hai tập con kích thước (k-1) đầu tiên có chung phần tiền tố (các itemset phát sinh). Các phương pháp này chia không gian tìm kiếm lớn thành các phần nhỏ, độc lập và có thể quản lý được. Các phần này có thể được xử lý trong bộ nhớ qua các lớp tương đương dựa trên các nhóm hoặc tiền tố; cách tiếp cận dựa trên nhóm tạo ra nhiều lớp nhỏ hơn. Mỗi lớp là độc lập theo nghĩa chúng có đầy đủ thông tin để tạo tất cả các itemset phổ biến có cùng tiền tố.

Trong bốn thuật toán này, Eclat sử dụng các lớp dựa trên tiền tố và tìm kiếm từ dưới lên, MaxEclat sử dụng các lớp dựa trên tiền tố và tìm kiếm lai, Clique dùng các lớp dựa trên nhóm và tìm kiếm từ dưới lên, MaxClique dùng các lớp dựa trên

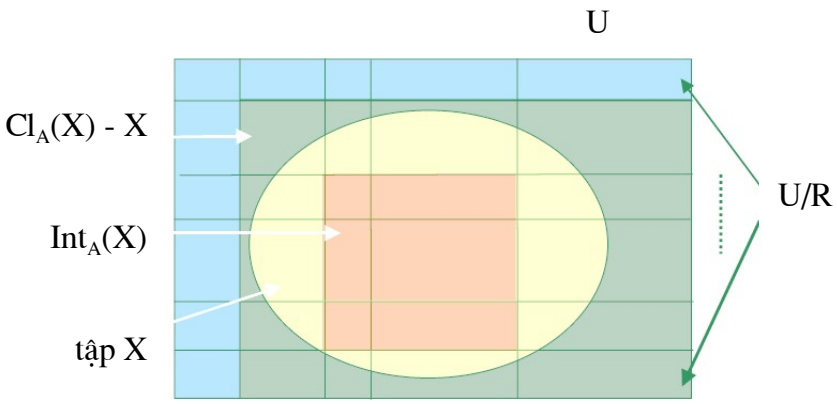
nhóm và tìm kiếm lai. Cách tiếp cận tốt nhất là MaxClique, nó tốt hơn cả Apriori và Eclat.

II.2. LUẬT KẾT HỢP THEO TIẾP CẬN LÝ THUYẾT TẬP THÔ

II.2.1. Tập thô [14]

Lý thuyết tập thô được phát triển bởi Zdzislaw Pawlak vào đầu những năm 1980. Mục đích chính của phân tích tập thô là suy dẫn ra các xấp xỉ của các khái niệm, nó cung cấp các công cụ toán học cho việc phát hiện các mẫu ẩn chứa trong dữ liệu và được dùng để lựa chọn thuộc tính, rút gọn dữ liệu, sinh luật quyết định và rút ra các mẫu.

Gọi cặp $A = (U, R)$ là một không gian xấp xỉ, với U là một tập hữu hạn, và R là tập các lớp tương đương trên U. Mỗi phần tử của tập R được gọi là một tập cơ sở (hay tập nguyên tử). Một *tập có thể định nghĩa* trong A là thu được nhờ áp dụng một số hữu hạn các phép hợp ($\cup$) trên R. Gọi R^* là họ các tập con của R. Khi đó, R^* tạo một không gian tôpô $T_A = (U, R^*)$. Ta gọi mỗi phần tử của U là một đối tượng. Một khái niệm đáng quan tâm X là một tập con của U. Một tập có thể định nghĩa trong A chứa X, $Cl_A(X)$ được gọi là tập đóng (còn gọi là tập trên) của X trong A. Tương tự, tập có thể định nghĩa lớn nhất trong A được chứa bởi X, $Int_A(X)$, được gọi là tập trong (hay còn gọi là tập dưới) của X trong A.



Đinh nghĩa 2.4 *(Tập mô tả được và tập thô)*

Tập X là mô tả được trong A nếu với một số tập $Y \in R^*$, X bằng hợp của tất cả các tập trong Y. Ngược lại, X được gọi là tập thô hay tập không mô tả được.

Ta muốn tạo một thuật toán quyết định trong A, ký hiệu là $D_A(X)$, sao cho với mỗi $x \in U$ nó cho một trong 3 câu trả lời: (a) x nằm trong X, (b) x không nằm trong X, (c) không biết. Ta định nghĩa các tập của X trong A tương ứng với mỗi câu trả lời:

- $POS_A(X)$ là tập các đối tượng được $D_A(X)$ coi là một phần tử của khái niệm X;
- $BND_A(X)$ là tập các đối tượng mà $D_A(X)$ cho câu trả lời "*không biết*";
- $NEG_A(X)$ là tập các đối tượng không được $D_A(X)$ coi là phần tuwr của X;

Theo định nghĩa trên, dễ thấy $NEG_A(X) = U - (POS_A(X) \cup BND_A(X))$. Nói cách khác, thuật toán quyết định dùng các luật sau để trả lời câu hỏi $x \in X$ hay không:

i. $x \in POS_A(X) \Rightarrow$ x ∈ X,

ii. $x \in BND_A(X) \Rightarrow$ không biết,

iii. $x \in NEG_A(X) \Rightarrow$ x ∉ X

Có hai phương pháp xấp xỉ được định nghĩa trong không gian xấp xỉ đại số:

- Xấp xỉ dưới: $POS^l_A(X) = Int_A(X)$ và
- Xấp xỉ trên: $POS^u_A(X) = Cl_A(X)$

Trong cả hai phương pháp, vùng biên của khái niệm X bằng $Cl_A(X) - POS_A(X)$. Độ mơ hồ được biểu diễn bởi độ đo chính xác: $\mu_A(X) = \frac{|Int_A(A)|}{|Cl_A(X)|}$

Đặt $F = \{X_1, X_2,..., X_k\}$, với $X_i \in U$, là một phân loại của U. Các tập trong và tập đóng của F trong A được định nghĩa tương ứng là các họ:

$Int_A(F) = \{Int_A(X_1), Int_A(X_2),..., Int_A(X_n)\}$

và $Cl_A(F) = \{Cl_A(X_1), Cl_A(X_2),..., Cl_A(X_n)\}$

Một bài toán phân lớp được mô tả như việc tạo một thuật toán quyết định $D_A(R, F)$ để liên kết các tập có thể định nghĩa với các khái niệm. Nếu $D_A(R, F)$ là một quan hệ thì nó được gọi là một thuật toán quyết định không nhất quán, ngược lại nó là một thuật toán quyết định nhất quán. Do $POS_A(R, F) = \cup_{x \in F} POS_A(R, X)$ nên sự mở rộng một phương pháp xấp xỉ cho bài toán phân lớp là dễ hiểu. Tương tự, độ chính xác phân lớp bằng:

$$\beta_A(F) = \frac{\sum_{i=1}^{k} |Int_A(X_i)|}{\sum_{i=1}^{k} |Cl_A(X_i)|}$$

Trong bài toán phân lớp, một độ đo thứ hai thường được nói tới, đó là chất lượng của phân loại F trong A, đươc cho bởi công thức:

$$\eta_A(F) = \frac{\sum_{i=1}^{k} |Int_A(x_i)|}{\sum_{i=1}^{k} |U|}$$

Nếu $\eta_A(F) = \beta_A(F)$ thì phân loại nàyđược gọi là có thể định nghĩa, ngược lại nó được gọi là phân loại có thể định nghĩa thô.

II.2.2 Luật kết hợp theo tiếp cận lý thuyết tập thô

Các cơ sở dữ liệu luôn chứa rất nhiều thuộc tính, một số thuộc tính có thể là dư thừa và không cần thiết cho quá trình phát hiện luật. Nếu các thuộc tính dư thừa này không bị loại bớt thì không chỉ thời gian phát hiện luật tăng lên, mà chất lượng của các luật tìm được cũng không cao.

Để sử dụng lý thuyết tập thô, ta coi cơ sở dữ liệu là một hệ quyết định [16]:

$T = (U, A \cup \{d\})$, trong đó

U là tập hữu hạn khác rỗng các đối tượng.

A là tập hữu hạn khác rỗng các thuộc tính sao cho $a: U \rightarrow V_a$ với $\forall a \in A$,

V_a được gọi là tập giá trị của a, các phần tử của A được gọi là các thuộc tính điều kiện.

$d \notin A$ là thuộc tính quyết định.

Ví dụ: Hệ quyết định:

U	***Đau đầu***	***Nhiệt độ***	***Mỏi cơ***	***Bị cúm***
u1	Có	Bình thường	Có	Không
u2	Có	Cao	Có	Có
u3	Có	Bình thường	Có	Có
u4	Không	Cao	Có	Không
u5	Không	Bình thường	Không	Không
u6	Có	Rất cao	Có	Có
u7	Không	Cao	Có	Có

Các thuộc tính điều kiện là *Đau đầu, Nhiệt độ, Mỏi cơ* với

$V_{\text{đau đầu}}$={*Không (0), Có (1)*},

$V_{\text{nhiệt độ}}$={*Bình thường(0), Cao (1), Rất cao (2)*},

$V_{\text{mỏi cơ}}$= {*Không (0), Có (1)*}

và thuộc tính quyết định là *Bị cúm* với $V_{\text{bị cúm}}$={ *Không (0), Có (1)*}.

Bảng quyết định tương ứng của nó là:

F(x)	0-0-0	0-0-1	...	1-0-0	...	1-2-1
*-0-0	1/2		...	1/2	.....	
*-0-1		1/2			.....	
*-1-0					.....	
*-1-1					.....	
*-2-0					.....	
*-2-1					.....	1/2
0-*-0	1/3				.....	
.....	.....				.....	

1-1-*					.....	
1-2-*					.....	1/2
.....					.....	
--0	1/6			1/6	.....	
.....	.....				.....	
0-*-*	1/6	1/6			.....	
1-*-*				1/6	.....	1/6

Gọi *F(x)* là các đối tượng có thể (*PI*)

G(x) là các bộ sinh có thể (*PG*)

G(x) $\rightarrow$ *F(x)* là quan hệ xác suất giữa *PI* và *PG*:

$$p(PI_j \mid PG_i) = \begin{cases} \dfrac{1}{N_{PG_i}} & \text{nếu } PI_j \in PG_i \\ 0 & \text{trong các trường hợp còn lại} \end{cases}$$

$N_{PG_i} = \prod_{k \in \{l \mid PG[l]=*\}} n_k$ là số *PI* thỏa mãn trong PG_i

Độ mạnh của luật: $S(X \rightarrow Y) = s(X)(1 - r(X \rightarrow Y))$ với $s(X) = s(PG_k)$

Nếu không sử dụng tri thức kinh nghiệm:

$$s(X) = s(PG_k) = \sum_l p(PI_l \mid PG_k) = \frac{N_{ins\_rel}(PG_k)}{N_{PG_k}}$$

$N_{ins\_rel}(PG_k)$ là số đối tượng quan sát được là thỏa mãn trong lần thứ i

Nếu sử dụng tri thức kinh nghiệm:

$$s(X) = s(PG_k) = \sum_l p_{bk}(PI_l \mid PG_k) = \frac{\sum_l BKF(PI_l \mid PG_k)}{N_{ins\_rel(X)}}$$

Độ nhiễu được tính bằng: $r(X \rightarrow Y) = \dfrac{N_{ins\_rel}(X) - N_{ins\_class}(X,Y)}{N_{ins\_rel}(X)}$

$N_{ins\_class}(X,Y)$ là số đối tượng thuộc lớp Y trong số các đối tượng thỏa mãn X

Quá trình khai phá trong bảng quyết định có thể phát hiện ra các đối tượng chưa biết và nó dùng độ mạnh của luật để biểu diễn tường minh tính không chắc chắn của luật, bao gồm cả khả năng luật dự đoán các đối tượng có thể.

Để chọn ra các luật trong bảng quyết định, ta dựa trên các tiêu chuẩn như:

- Chọn các luật phủ càng nhiều đối tượng càng tốt
- Chọn các luật chứa càng ít thuộc tính càng tốt, nếu chúng phủ cùng số đối tượng
- Chọn các luật mạnh hơn, nếu chúng có cùng số thuộc tính điều kiện và phủ cùng số đối tượng

Các thuộc tính tham gia trong luật cần được chọn sao cho số đối tượng tăng nhanh hơn, nhằm có tập con các thuộc tính càng nhỏ càng tốt, và chúng nên có số giá trị ít hơn, để đảm bảo số đối tượng do luật này bao phủ càng lớn càng tốt.

Xét bảng cơ sở dữ liệu nêu trên:

U	*Đau đầu*	*Nhiệt độ*	*Mỏi cơ*	*Bị cúm*
u1	Có	BT	Có	Có
u2	Có	Cao	Có	Có
u3	Có	BT	Có	Có
u4	Ko	Cao	Ko	Ko
u5	Có	BT	Có	Ko
u6	Có	Rất cao	Có	Ko
u7	Ko	Cao	Có	Có

$\Rightarrow$

U	*Đau đầu*	*Nhiệt độ*	*Mỏi cơ*	*Bị cúm*
u1’	Có	BT	Có	$\perp$
u2	Có	Cao	Có	Có
u4	Ko	Cao	Ko	Ko
u6	Có	Rất cao	Có	Ko
u7	Ko	Cao	Có	Có

$r_{(có)}(u1') = 1 - 2/3 = 0{,}33$

$r_{(không)}(u1') = 1 - 1/3 = 0{,}67$

Đặt $T_{nhiễu} = 0 \Rightarrow r_{(có)}(u1') > T_{nhiễu}$; $r_{(không)}(u1') > T_{nhiễu}$ và *Bị cúm*$(u1') = \perp$

Tạo véctơ phân biệt cho *u2*:

	u1’	u2	u4	u6	u7
u2	Nhiệt độ	λ	Đau đầu, Mỏi cơ	Nhiệt độ	λ

Tìm rút gọn cho *u2*:

$f_T(u2)$ = *(Nhiệt độ)* $\wedge T \wedge$ *(Đau đầu* $\vee$ *Mỏi cơ)* $\wedge$ *(Nhiệt độ)* $\wedge T$

= *(Nhiệt độ)* $\wedge$ *(Đau đầu* $\vee$ *Mỏi cơ)*

= *(Đau đầu* $\wedge$ *Nhiệt độ)* $\vee$ *(Nhiệt độ* $\wedge$ *Mỏi cơ)*

Tạo luật từ *u2*:

$f_T(u2)$ = (*Đau đầu* $\wedge$ *Nhiệt độ*) $\vee$ (*Nhiệt độ* $\wedge$ *Mỏi cơ*)

{Có đau đầu, Nhiệt độ cao} *{Nhiệt độ cao, Có mỏi cơ}*

(*{Có đau đầu, Nhiệt độ cao}* $\rightarrow$ *Bị cúm*) có

s(*{Có đau đầu, Nhiệt độ cao}*) = 0.5 và

r(*{Có đau đầu, Nhiệt độ cao}* $\rightarrow$ *Bị cúm*) = 0 $\Rightarrow$

S(*{Có đau đầu, Nhiệt độ cao}* $\rightarrow$ *Bị cúm*) = (1 x 1/2) x (1-0) = 0.5

(*{Nhiệt độ cao,Có mỏi cơ}* $\rightarrow$ *Bị cúm*) có

s(*{Nhiệt độ cao, Có mỏi cơ}*) = 1 và

r(*{Nhiệt độ cao,Có mỏi cơ}* $\rightarrow$ *Bị cúm*) = 0 $\Rightarrow$

S(*{Nhiệt độ cao, Có mỏi cơ}* $\rightarrow$ *Bị cúm*) = (2 x 1/2) x (1-0) = 1

Tạo véctơ phân biệt cho *u4*:

	u1'	u2	u4	u6	u7
u4	Đau đầu, Nhiệt độ, Mỏi cơ	Đau đầu, Mỏi cơ	λ	λ	Mỏi cơ

$f_T(u4)$ = *(Đau đầu* $\vee$ *Nhiệt độ* $\vee$ *Mỏi cơ)* $\wedge$ *(Đau đầu* $\vee$ *Mỏi cơ)* $\wedge T \wedge T \wedge$ *(Mỏi cơ)*

= *(Mỏi cơ)*

Tạo luật từ *u4*:

$f_T(u4)$ = (*Mỏi cơ*)

{Không mỏi cơ}

(*{Không mỏi cơ}* $\rightarrow$ *Không bị cúm)* có

s(*Không mỏi cơ*) = 1/6 và

r(*{Không mỏi cơ}* $\rightarrow$ *Không bị cúm*) = 0 $\Rightarrow$

S(*{Không mỏi cơ}* $\rightarrow$ *Không bị cúm*) = (1 x 1/6) x (1-0) = 0,167

Sau khi tạo luật từ tất cả các đối tượng ta có:

u2: *{Có đau đầu, Nhiệt độ cao} → Bị cúm,* *S = 0.5*

{Nhiệt độ cao, Có mỏi cơ} → Bị cúm, *S = 1*

u4: *{Không mỏi cơ} → Không bị cúm,* *S = 0,167*

u6: *{Nhiệt độ rất cao} → Không bị cúm,* *S = 0.25*

u7: *{Không đau đầu, Có mỏi cơ } → Bị cúm,* *S = 0.5*

{Nhiệt độ cao, Có mỏi cơ } → Bị cúm, *S = 1*

Bộ sinh thuộc lớp *Bị cúm*:

	u2	*u7*
	Đau đầu, Nhiệt độ cao, Mỏi cơ	*Không đau đầu, Nhiệt độ cao, Mỏi cơ*
**, Nhiệt độ cao, Mỏi cơ*	1/2	1/2
*Không đau đầu, *, Mỏi cơ*		1/3
*Có đau đầu, Nhiệt độ cao, **	1/2	

{Nhiệt độ cao, Có mỏi cơ } → Bị cúm với *S = 1*

{Không đau đầu, Có mỏi cơ} → Bị cúm với *S = 1/2* phủ *u7*

{Có đau đầu, Nhiệt độ cao} → Bị cúm với *S = 1/2* phủ *u2*

Bộ sinh thuộc lớp *Không bị cúm*:

	u2	*u7*
	Có đau đầu, Nhiệt độ rất cao, Có mỏi cơ	*Không đau đầu, Nhiệt độ cao, Không mỏi cơ*
**, *, Không mỏi cơ*		*1/6*
**, Nhiệt độ rất cao, **	*1/4*	

Không mỏi cơ → Không bị cúm với *S = 1/6* phủ *u4*

Nhiệt độ rất cao → Không bị cúm với *S = 1/4* phủ *u6*

Như vậy với độ nhiễu bằng *0*, từ cơ sở dữ liệu trên ta có:

Các luật chắc chắn:	**Phủ các đối tượng**
Không mỏi cơ → Không bị cúm với *S = 1/6*	*u4*

Nhiệt độ rất cao $\rightarrow$ *Không bị cúm* với $S = 1/4$ *u6*

{Nhiệt độ cao, Có mỏi cơ } $\rightarrow$ *Bị cúm* với $S = 1$ *u2, u7*

Các luật có thể:

Nhiệt độ bình thường $\rightarrow$ *Bị cúm* với $S = (1/4)*(2/3)$

Có đau đầu & Nhiệt độ Bình thường $\rightarrow$ *Bị cúm* với $S = (1/2)*(2/3)$

Có đau đầu & Có mỏi cơ $\rightarrow$ *Bị cúm* với $S = (1/3)*(2/3)$

Nhiệt độ bình thường & Có mỏi cơ $\rightarrow$ *Bị cúm* với $S = (1/2)*(2/3)$

Nếu độ nhiễu khác *0*, giả sử $T_{nhiễu} = 0,5$:

U	*Đau đầu*	*Nhiệt độ*	*Mỏi cơ*	*Bị cúm*
u1	Có	BT	Có	Có
u1' u3	Có	BT	Có	Có
u5	Có	BT	Có	Ko
u2	Có	Cao	Có	Có
u4	Ko	Cao	Ko	Ko
u6	Có	Rất cao	Có	Ko
u7	Ko	Cao	Có	Có

$\Rightarrow$

U	*Đau đầu*	*Nhiệt độ*	*Mỏi cơ*	*Bị cúm*
u1'	Có	BT	Có	$\perp$
u2	Có	Cao	Có	Có
u4	Ko	Cao	Ko	Ko
u6	Có	Rất cao	Có	Ko
u7	Ko	Cao	Có	Có

$r_{(có)}(u1') = 1 - 2/3 = 0,33 < T_{nhiễu}$ và $r_{(không)}(u1') = 1 - 1/3 = 0,67 > T_{nhiễu} \Rightarrow$ d(u1') = *Có*

Khi đó các luật có được từ tất cả các đối tượng là:

u1': *{Nhiệt độ bình thường}* $\rightarrow$ *Bị cúm,* $S = 0,5$

u2: *{Có đau đầu, Nhiệt độ cao}* $\rightarrow$ *Bị cúm* $S = 0,5$

{Nhiệt độ cao, Có mỏi cơ} $\rightarrow$ *Bị cúm* $S = 1$

u4: *{Không mỏi cơ}* $\rightarrow$ *Không bị cúm* $S = 0,167$

u6: *{Nhiệt độ rất cao}* $\rightarrow$ *Không bị cúm* $S = 0,25$

u7: *{Không đau đầu, Có mỏi cơ}* $\rightarrow$ *Bị cúm* $S = 0,5$

{Nhiệt độ cao, Có mỏi cơ} $\rightarrow$ *Bị cúm* *S = 1*

Nếu sử dụng tri thức kinh nghiệm, từ bảng

	0-0-0	*0-0-1*	*0-1-0*	*0-1-1*	*0-2-0*	*0-2-1*	...	*1-2-1*
*0-0-**	1/2	1/2						
*0-1-**			1/2	1/2				
0--1*		1/3		1/3		1/3		
0--**	1/6	1/6	1/6	1/6	1/6	1/6		

với tri thức là *Có đau đầu* $\rightarrow$ *Có mỏi cơ*, chắc chắn *100%* ta sẽ có độ mạnh của các luật thay đổi như sau:

	0-0-0	*0-0-1*	*0-1-0*	*0-1-1*	*0-2-0*	*0-2-1*	...	*1-2-1*
*0-0-**	0	1						
*0-1-**			0	1				
0--1*		1/3		1/3		1/3		
0--**	0	1/3	0	1/3	0	1/3		

Trong [16], các tác giả đã đưa ra thuật toán rút ra các luật từ cơ sở dữ liệu có dạng bảng quyết định.

Thuật toán 2.1: **Tìm tập tối ưu các luật [16]**

Input: Bảng quyết định *U* gồm *n* đối tượng, mỗi đối tượng *u* có thể có *m* thuộc tính.

Output: Tập tối ưu các luật cùng độ mạnh của mỗi luật

Nội dung thuật toán

Bước 1: Coi các đối tượng với cùng giá trị của các thuộc tính điều kiện như một đối tượng, gọi là đối tượng kép

Bước 2: Tính độ nhiễu *r* cho mọi đối tượng kép

Bước 3: Chọn một đối tượng *u* từ *U* và tính vec-tơ không phân biệt được cho *u*

Bước 4: Tính tất cả các rút gọn cho đối tượng u, sử dụng hàm không phân biệt được

Bước 5: Rút ra các luật từ các rút gọn của đối tượng *u*, tính lại độ mạnh của mẫu cho mỗi luật

Bước 6: Chọn ra các luật tốt hơn từ các luật tính được ở *bước 5* bằng cách chọn ngẫu nhiên

Bước 7: $U = U \setminus \{u\}$. Nếu $U \neq \varnothing$ thì quay lại *bước 3,* nếu không chuyển sang *bước 8*

Bước 8: Kết thúc nếu số luật trong *bước 6* cho mỗi đối tượng bằng 1, ngược lại tìm tập luật tối thiểu, chứa tất cả các đối tượng trong bảng quyết định.

Độ phức tạp về thời gian của thuật toán này là $O\left(mn^3 + mn^2 N\left(G_T\right)\right)$ với $N\left(G_T\right)$ là số lần sinh và nhỏ hơn $O\left(2^{m-1}\right)$.

Thuật toán này là không phù hợp cho các cơ sở dữ liệu có số thuộc tính lớn, để giải quyết vấn đề này, ta sẽ tìm một rút gọn của các thuộc tính điều kiện trong giai đoạn tiền xử lý và tìm một giải pháp gần tối ưu, sử dụng một số phép phỏng đoán hiệu quả.

Thuật toán 2.2: **Giải pháp gần tối ưu [16]**

Bước 1: Đặt $R = \{\}$, $COVER = \{\}$, $SS = \{$*định danh của tất cả các đối tượng*$\}$. Với mỗi lớp D_c, chia bảng quyết định T làm 2 phần: lớp hiện tại T_+ và các lớp còn lại T_-

Bước 2: Từ các giá trị v_{ij} của các đối tượng I_k (v_{ij} là giá trị thứ j của thuộc tính thứ i, $I_k \in T_+$, $I_k \in SS$) chọn một giá trị v có số lần xuất hiện nhiều nhất trong các đối tượng chứa trong T_-.

Bước 3: Thêm v vào R

Bước 4: Xóa định danh của đối tượng khỏi *SS* nếu đối tượng đó không chứa *v*.

Bước 5: Quay lại *bước 2* tới khi độ nhiễu nhỏ hơn giá trị ngưỡng

Bước 6: Tìm một tập con tối thiểu *R'* của *R* tuỳ theo độ mạnh của nó. Thêm luật ($R' \rightarrow D_c$) vào *RS*. Đặt *R* = *{}*, sao chép định danh của các đối tượng từ *SS* vào *COVERED* và đặt *SS* =*{mọi định danh của các đối tượng }\ COVERED*

Bước 7: Quay lại *bước 2* tới khi mọi đối tượng của T_+ đều ở trong *COVERED*

Bước 8: Quay lại *bước 1* tới khi mọi lớp được xử lý xong.

Độ phức tạp về thời gian của thuật toán này là: $O\left(m^2 n^2\right)$.

KẾT LUẬN CHƯƠNG II

Phát hiện luật kết hợp là một trong các kỹ thuật đơn giản và hiệu quả của khai phá dữ liệu. Theo cách này, tri thức được phát biểu dưới dạng các luật biểu diễn sự phụ thuộc giữa các tập con thuộc tính xuất hiện thường xuyên trong cơ sở dữ liệu *(mục 2.1.1)*. Việc phát hiện ra các luật kết hợp từ cơ sở dữ liệu đòi hỏi lượng tính toán và truy xuất dữ liệu vô cùng lớn. Nhiều thuật toán tuần tự đã được phát triển cho các mô hình khác nhau *(mục 2.1.2)*. Trên thực tế, hiện tượng dữ liệu không đầy đủ hoặc không chính xác là có thể xảy ra, nó ảnh hưởng không tốt tới quá trình nhằm phát hiện ra tri thức chính xác từ dữ liệu. Lý thuyết tập thô đã được phát triển bởi Pawlak [14] cho phép suy dẫn ra các xấp xỉ của các khái niệm, giúp rút gọn dữ liệu trong quá trình tìm kiếm mẫu và sinh luật *(mục 2.2.1)*. Lý thuyết này được sử dụng trong việc phát hiện luật kết hợp từ cơ sở dữ liệu dạng bảng quyết định *(mục 2.2.2)*. Việc sử dụng tri thức kinh nghiệm trong chọn luật cũng giúp giảm bớt được số thuộc tính cần xem xét để tạo luật. Hai tác giả A. Skowron & N. Zhong [16] đã đưa ra một

thuật toán tuần tự tìm tập tối ưu các luật từ bảng quyết định cùng với cải tiến của nó nhằm giảm bớt độ phức tạp tính toán.

CHƯƠNG III. PHÁT HIỆN SONG SONG LUẬT KẾT HỢP

III.1. KHÔNG GIAN THIẾT KẾ SONG SONG

Người ta hi vọng song song hóa sẽ làm giảm được khó khăn cho các phương pháp phát hiện luật kết hợp tuần tự, cung cấp khả năng mở rộng cho các tập dữ liệu lớn và cải thiện được tốc độ thực hiện. Có được hiệu suất cao từ các hệ đa xử lý hiện thời không phải là một chuyên dễ. Các thách thức chính bao gồm việc giảm thiểu sự truyền thông và đồng bộ hóa, cân bằng khối lượng công việc, tìm được cách tốt để trình bày dữ liệu, phân tích dữ liệu và giảm thiểu việc truy/xuất đĩa (đây là điều rất quan trọng cho việc phát hiện luật kết hợp). Không gian thiết kế song song gồm 3 phần chính: nền phần cứng, kiểu song song hóa, và chiến lược cân bằng tải công việc.

III.1.1. Nền phần cứng: Các hệ thống phân tán bộ nhớ/ Các hệ thống chia sẻ bộ nhớ

Trong một kiến trúc có bộ nhớ chia sẻ, mỗi bộ xử lý có quyền truy nhập ngang bằng và trực tiếp tới tất cả bộ nhớ của hệ thống. Các chương trình song song dễ thực hiện trên hệ thống như vậy. Một cách tiếp cận đa xử lý khác là xây dựng một hệ thống từ nhiều bộ phận, mỗi bộ phận chứa một bộ xử lý và bộ nhớ. Trong kiến trúc bộ nhớ phân tán, mỗi bộ xử lý có riêng bộ nhớ cục bộ mà chỉ mình nó có quyền truy nhập. Nếu một bộ xử lý muốn truy cập dữ liệu trên bộ nhớ cục bộ của bộ xử lý khác, một bản sao của dữ liệu cần dùng phải được gửi giữa hai bộ xử lý. Mặc dù kiến trúc chia sẻ bộ nhớ cho phép lập trình đơn giản hơn, nhưng băng thông hữu hạn của đường truyền sẽ hạn chế khả năng mở rộng. Kiến trúc dùng bộ nhớ phân tán và truyền thông báo giải quyết vấn đề phát triển hệ thống, nhưng lập trình không đơn giản.

Một mô hình thứ ba rất phổ biến, kết hợp những ưu điểm của các cách tiếp cận dùng bộ nhớ chia sẻ và bộ nhớ phân tán. Mô hình này bao gồm các hệ thống chia sẻ bộ nhớ với phần cứng hoặc phần mềm phân tán. Các hệ thống này phân chia bộ nhớ vật lý giữa các nút nhưng cung cấp một không gian địa chỉ toàn cục được chia sẻ trên mỗi bộ xử lý. Phần cứng và phần mềm đảm bảo sự mạch lạc trong bộ lưu trữ, giúp dữ liệu được lưu trữ cục bộ luôn phản chiếu các thay đổi lên mọi bộ xử lý. Nhóm các máy trạm chia sẻ bộ nhớ (clump) cũng là một phần của mô hình này. Các clump đòi hỏi phải có cách tiếp cận song song có phân cấp, với bộ nhớ chia sẻ nguyên thủy được dùng trong một nút và các thông báo được truyền giữa các nút chia sẻ bộ nhớ.

Mục đích tối ưu hóa hiệu suất cho các máy có bộ nhớ phân tán so với các hệ thống chia sẻ bộ nhớ phụ thuộc vào kiến trúc cơ sở. Trong hệ thống phân tán bộ nhớ, sự đồng bộ hóa nằm ẩn trong việc truyền thông báo, ví thế mục tiêu trở thành tối ưu hóa sự truyền thông. Với các hệ chia sẻ bộ nhớ, sự đồng bộ hóa xuất phát từ các khóa và các hàng rào ngăn cách, và mục tiêu là phải tối ưu hóa những điểm này. Sự phân rã dữ liệu là rất quan trọng trong các hệ phân tán, song lại không quan trọng trong các hệ chia sẻ bộ nhớ. Trong khi việc nhập/xuất song song là đơn giản với các hệ có bộ phân tán, nó lại là vấn đề khó đối với các hệ chia sẻ bộ nhớ, nơi mà việc nhập/xuất thường được tuần tự hóa. Thách thức chủ yếu để đạt hiệu suất cao cho các hệ có bộ nhớ phân tán là phải tìm được một sự phân rã tốt dữ liệu giữa các nút và giảm thiểu sự truyền thông.

Với các hệ chia sẻ bộ nhớ, mục đích là có được vị trí tốt cho dữ liệu, nghĩa là ta phải tối đa sự truy nhập tới bộ nhớ địa phương và tránh/ giảm lỗi chia sẻ bộ nhớ. Như vậy ta cần giảm hiệu ứng "bóng bàn" - nhiều bộ xử lý cùng cố thay đổi các biến khác nhau cùng nằm trùng khớp tại một hàng trong bộ nhớ. Với các máy phân cấp lai truy nhập bộ nhớ không đồng bộ, các tham số tối ưu được lấy từ cả hai mô hình bộ nhớ phân tán và bộ nhớ chia sẻ.

III.1.2. Song song hóa dữ liệu và song song hóa công việc

Song song hóa dữ liệu và song song hóa công việc là hai mô hình chính để khai thác các thuật toán song song. Với việc phát hiện luật kết hợp, song song hóa dữ liệu ứng với việc cơ sở dữ liệu được chia ra trên *p* bộ xử lý - phân chia về mặt logic cho bộ nhớ chia sẻ, phân chia về mặt vật lý cho các hệ phân tán bộ nhớ. Mỗi bộ xử lý làm việc trên phần cơ sở dữ liệu của nó nhưng thực hiện cùng một phép đếm độ hỗ trợ cho các itemset ứng viên toàn cục. Song song hóa công việc ứng với việc các bộ xử lý thực hiện các phép tính khác nhau một cách độc lập, như là đếm một tập không giao nhau các ứng viên, nhưng không cần truy nhập tới toàn bộ cơ sở dữ liệu. Các hệ chia sẻ bộ nhớ có truy nhập tới toàn bộ dữ liệu, nhưng với các hệ phân tán bộ nhớ, quá trình truy nhập cơ sở dữ liệu có thể liên quan tới việc tái tạo một cách có chọn lựa hoặc truyền thông giữa các phần cục bộ. Cũng có thể kết hợp cả song song hóa dữ liệu và song song hóa công việc, đây là cách được ưa dùng để khai thác hết được các cách song song hóa có sẵn cho các thuật toán phát hiện luật kết hợp.

III.1.3. Cân bằng tải tĩnh và cân bằng tải động

Cân bằng tải tĩnh ban đầu phân chia công việc giữa các bộ xử lý dùng hàm chi phí theo kinh nghiệm, không có thay đổi nào trên dữ liệu hoặc trong tính toán nào có thể điều chỉnh sự mất cân bằng về tải do bản chất động của các thuật toán phát hiện luật kết hợp. Việc cân bằng tải động giải quyết vấn đề này bằng cách lấy công việc từ các bộ xử lý phải chịu tải nặng và gán sang các bộ xử lý có lượng công việc ít hơn. Các thay đổi trong tính toán cũng dẫn tới các thay đổi trên dữ liệu, do bộ xử lý chịu trách nhiệm cho nhiệm vụ tính toán cần các dữ liệu kết hợp với nhiệm vụ này. Vì thế cân bằng tải động phải chịu thêm chi phí dịch chuyển dữ liệu và công việc, và cho cả cơ chế xác định sự mất cân bằng. Tuy nhiên cân bằng tải động là cần thiết nếu có sự mất cân bằng lớn về tải hoặc tải thay đổi theo thời gian.

Cân bằng tải động đặc biệt quan trọng trong môi trường nhiều người sử dụng với các tải tạm thời và trên nền không đồng nhất có bộ xử lý và mạng với tốc độ khác nhau. Các kiểu môi trường này bao gồm các máy chủ song song và các cluster, metacluster và supercluster không đồng nhất. Tất cả các thuật toán phát hiện luật kết hợp mở rộng chỉ dùng cách cân bằng tải tĩnh có sẵn nhờ sự phân chia cơ sở dữ liệu sẵn có giữa các nút. Điều này là do giả thiết có một môi trường đồng nhất và chuyên dụng.

Vấn đề thiết kế chính trong các hệ phân tán bộ nhớ là việc giảm thiểu sự truyền thông và thậm chí cả phân tán dữ liệu để có sự cân bằng tải. Các thuật toán phát hiện luật kết hợp dưới đây giả sử rằng cơ sở dữ liệu được chia thành các phần có kích thước như nhau trên ổ đĩa cục bộ của các bộ xử lý.

III.2. MỘT SỐ MÔ HÌNH PHÁT HIỆN SONG SONG LUẬT KẾT HỢP

Tồn tại nhiều thuật toán thực hiện việc phát hiện song song các luật kết hợp. Dưới đây là những thuật toán điển hình nhất [15, 17].

III.2.1. Các hệ phân tán bộ nhớ

- ***Thuật toán PEAR - dựa trên SEAR và SPEAR***

Andrea Muller đưa ra một số phương pháp phát hiện song song luật kết hợp trên nền các phương pháp tuần tự dựa trên các thuật toán Partition và Apriori. PEAR là bản song song của SEAR, trong mỗi lần lặp, mỗi bộ xử lý tạo một cây tiền tố các ứng viên từ các itemset phổ biến toàn phần của lần duyệt trước. Mỗi bộ xử lý có toàn bộ bản sao của tập ứng viên đó. Tiếp theo mỗi nút tập hợp các độ hỗ trợ địa phương, cùng với một rút gọn của tổng để có được độ hỗ trợ toàn phần trên mỗi bộ xử lý.

Cách phân chia song song các luật kết hợp (PPAR) là dựa trên SPEAR, nhưng PPAR dùng định dạng dữ liệu theo hàng ngang. Thuật toán này hoạt động

như sau: Mỗi bộ xử lý tập hợp các itemset phổ biến địa phương với mọi kích thước trên cơ sở dữ liệu địa phương của nó trong một lần duyệt. PPAR thông báo các itemset phổ biến tiềm năng tới tất cả các bộ xử lý khác. Trong lần duyệt cục bộ thứ hai, mỗi bộ xử lý tập hợp con số các ứng viên toàn phần. Cuối cùng, một thông báo về tập itemset phổ biến toàn phần được phát ra. Các thử nghiệm cho thấy PEAR luôn tốt hơn PPAR, bởi vì PEAR dùng các gộp nhiều lần duyệt trong khi PPAR có thể tạo nên một cách không cần thiết nhiều ứng viên mà rốt cục là không phổ biến.

- ***Thuật toán PDM - dựa trên DHP***

Trong thuật toán PDM, mỗi bộ xử lý tạo các độ hỗ trợ địa phương của các 1-itemset và tính xấp xỉ số các 2-itemset với một bảng băm. Thông báo từ mỗi nút tới tất cả các nút khác về số đếm cục bộ sẽ giúp tính được số các 1-itemset toàn phần. Do bảng băm chứa các 2-itemset có thể rất lớn, sự trao đổi trực tiếp các con số qua thông báo giữa tất cả các nút sẽ rất tốn kém. Các tác giả đã dùng cách tối ưu hóa là chỉ trao đổi các phần tử được đảm bảo là sẽ phổ biến. Tuy nhiên, phương pháp này đòi hỏi hai lần truyền thông. Với lần duyệt thứ hai, PDM tạo các ứng viên địa phương dùng bảng băm các 2-itemset toàn phần. Chỉ lần duyệt thứ hai dùng bảng băm, các lần duyệt sau tạo các ứng viên trực tiếp từ F_{k-1} (như trong thuật toán Apriori).

Các ứng viên được tạo một cách song song. Mỗi bộ xử lý tạo tập ứng viên của riêng nó, sau đó sẽ trao đổi qua thông báo giữa tất cả các bộ xử lý để có được tập ứng viên toàn phần. Tiếp đó, PDM có số các ứng viên địa phương và trao đổi chúng giữa các bộ xử lý để có các itemset phổ biến toàn phần. Công việc được chuyển sang bước tiếp theo. PDM có một số hạn chế: Trước hết, nó song song hóa việc tạo các ứng viên và đòi hỏi các thông báo phải được truyền giữa tất cả các bộ xử lý để tạo một tập ứng viên toàn phần. Chi phí truyền thông này có thể làm giảm hiệu quả của song song hóa.

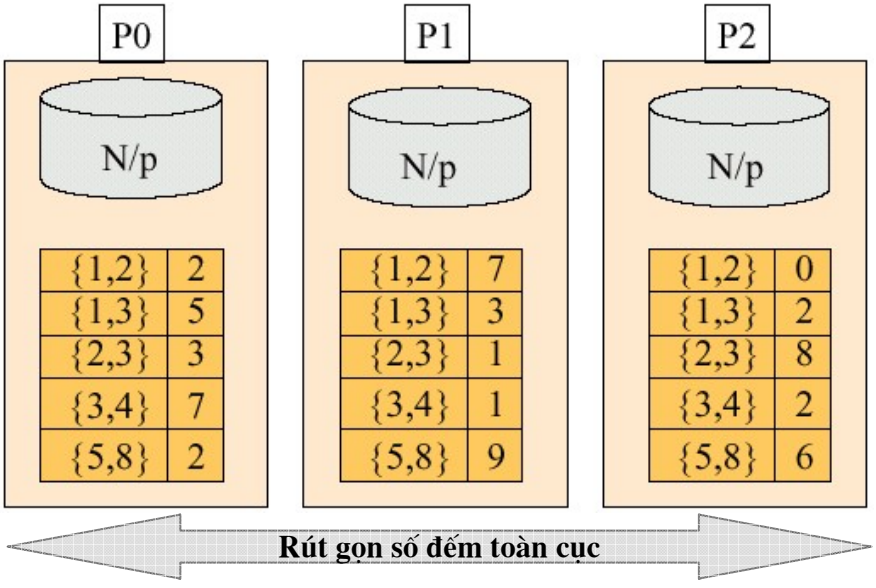
- ***Các thuật toán dựa trên Apriori***

Nhiều thuật toán song song sử dụng Apriori như là phương pháp cơ bản bởi sự thành công của nó trong thiết đặt tuần tự.

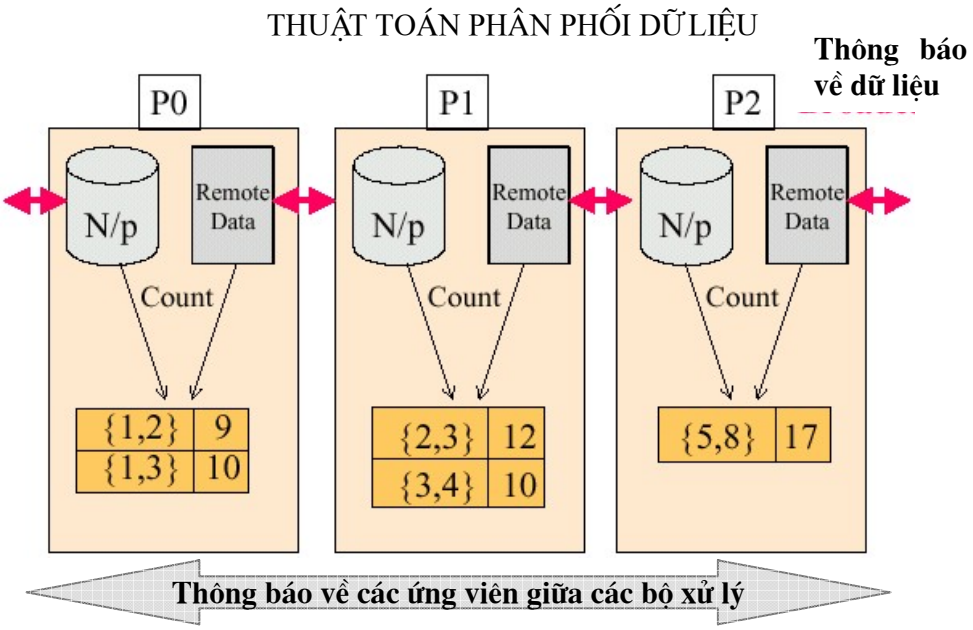
- **Phân phối số đếm, dữ liệu và ứng viên:**

Thuật toán phân phối số đếm là một cách song song hóa đơn giản của Apriori. Tất cả các bộ xử lý tạo một cây băm các ứng viên toàn phần từ F_{k-1}. Mỗi bộ xử lý khi đó có được một cách độc lập độ hỗ trợ địa phương từ phần cơ sở dữ liệu bộ phận của nó. Tiếp theo, thuật toán làm một phép cộng giản ước để có số đếm toàn phần bằng cách trao đổi các số đếm địa phương với các bộ xử lý khác. Thay vì trộn các cây băm khác nhau, thuật toán chỉ cần trao đổi các số đếm địa phương, vì tất cả các bộ xử lý đã có bản sao của toàn bộ cây băm. Mỗi khi xác định được F_k toàn phần, mỗi bộ xử lý xây dựng song song toàn bộ ứng viên C_{k-1}, và lặp lại quá trình tới khi tìm được tất cả các itemset phổ biến. Thuật toán này giảm thiểu việc truyền thông, tuy nhiên, do nó sao toàn bộ cây băm trên mỗi bộ xử lý, nó không sử dụng hiệu quả toàn bộ bộ nhớ hệ thống.

THUẬT TOÁN PHÂN PHỐI SỐ ĐẾM



Thuật toán phân phối dữ liệu dùng toàn bộ bộ nhớ hệ thống bằng cách tạo các tập ứng viên rời nhau trên mỗi bộ xử lý. Tuy nhiên, để tính độ hỗ trợ toàn phần, mỗi bộ xử lý phải duyệt toàn bộ cơ sở dữ liệu trong tất cả các lần lặp. Do đó, thuật toán này phải chịu gánh nặng lớn về truyền thông và hiệu quả không cao so với thuật toán phân phối số đếm.



Thuật toán phân phối ứng viên phân chia các ứng viên trong lần lặp *l* để mỗi bộ xử lý có thể tạo các ứng viên không giao nhau độc lập với các bộ xử lý khác. Việc phân chia sử dụng kinh nghiệm dựa trên độ hỗ trợ, khiến mỗi bộ xử lý có lượng công việc như nhau. Đồng thời, cơ sở dữ liệu cũng được sao chép một cách có chọn lựa để mỗi bộ xử lý có thể tính số đếm toàn phần một cách độc lập. Sự lựa chọn pha phân phối lại liên quan tới sự thỏa hiệp giữa việc tách riêng các bộ xử lý độc lập càng sớm càng tốt và việc đợi tới khi có đủ sự cân bằng tải. Trong các thí nghiệm của Agrawal và Shafer, sự phân chia lại được thực hiện trong bốn lần. Sau đó, chỉ bộ xử lý độc lập với các bộ xử lý khác là bị cắt bớt các ứng viên. Mỗi bộ xử lý thông báo không đồng bộ tập phổ biến địa phương của mình tới các bộ xử lý khác trong mỗi lần lặp. Nếu sự cắt xén thông tin này xảy ra đúng lúc, nó được dùng, nếu

không nó sẽ được lưu lại dùng cho lần lặp sau. Mỗi bộ xử lý vẫn phải duyệt dữ liệu cục bộ của nó trong mỗi lần lặp. Thậm chí khi có thông tin đặc trưng cho bài toán, thuật toán phân phối ứng viên vẫn thực hiện tồi hơn thuật toán phân phối số đếm, do nó phải trả giá cho việc phân phối lại cơ sở dữ liệu khi lặp lại việc duyệt phần dữ liệu cục bộ.

- **Các thuật toán Apriori không phân chia, phân chia đơn giản và phân chia băm**

Apriori không phân chia (NPA) về cơ bản giống với thuật toán phân phối số đếm, ngoại trừ việc rút gọn tổng được thực hiện trên bộ xử lý chính. Thuật toán Apriori phân chia đơn giản (SPA) giống hệt thuật toán phân phối dữ liệu.

Thuật toán Apriori phân chia băm (HPA) tương tự thuật toán phân phối ứng viên. Mỗi bộ xử lý tạo các ứng viên từ tập phổ biến tạo ở mức trước và áp dụng hàm băm để xác định bộ xử lý chủ cho một ứng viên. Nếu một bộ xử lý là chủ của một ứng viên, nó sẽ thêm ứng viên đó vào cây băm tại chỗ, nếu không, nó bỏ qua ứng viên. Để đếm, thuật toán này không lặp lại cơ sở dữ liệu một cách có chọn lựa như thuật toán phân phối ứng viên, mỗi bộ xử lý tạo một tập con k phân tử cho mỗi giao dịch địa phương, tìm bộ xử lý đích, và gửi tập con đó đến bộ xử lý. Bộ xử lý chủ có trách nhiệm tăng số đếm sử dụng cơ sở dữ liệu địa phương và bất kỳ thông báo nào từ các bộ xử lý khác.

Shitani và Kitsuregawa cũng đề xuất một biến đổi cho thuật toán Apriori phân chia băm, gọi là Apriori phân chia băm với sự nhân đôi các tập dữ liệu cực lớn (HPA-ELD). Động cơ của thuật toán này là dù ta có phân chia các ứng viên một cách đồng đều trên các bộ xử lý, vẫn có ứng viên phổ biến hơn các ứng viên khác. Vì thế, bộ xử lý chủ của nó sẽ có tải nặng hơn, trong khi các bộ xử lý khác có tải nhẹ. HPA-ELD giải quyết chuyện này bằng cách sao chép các itemset rất phổ biến trên tất cả các bộ xử lý và xử lý chúng bằng sơ đồ NPA. Do đó, không cần truyền

tập con cho các ứng viên này. Khi đã tính được các số đếm cục bộ, tiếp theo là phép tính rút gọn tổng để có độ hỗ trợ toàn phần.

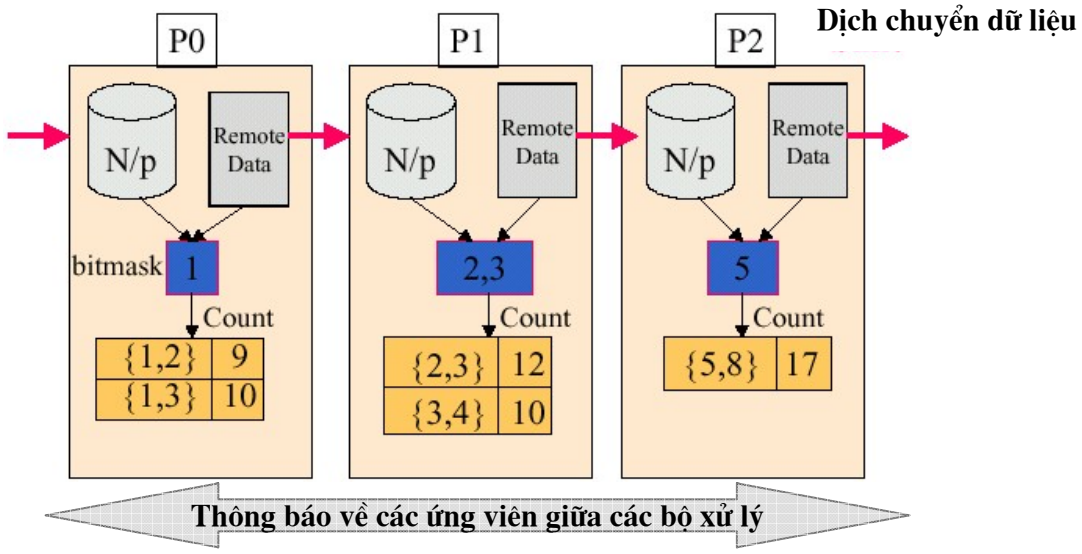
Shitani và Kitsuregawa đã khẳng định bằng các thí nghiệm rằng HPA-ELD thực hiện tốt hơn các cách tiếp cận khác. Tuy nhiên, họ chỉ dùng SPA, HPA và HPA-ELD cho lần lặp thứ hai, và với các lần lặp sau, họ dùng apriori không phân chia. Điều này cho thấy cách tiếp cận tốt nhất là cách lai: dùng HPA-ELD chừng nào các ứng viên còn chưa vừa trong bộ nhớ, sau đó chuyển sang dùng Apriori không phân chia (NPA). Điều này rất có ý nghĩa bởi Apriori không phân chia và phân phối số đếm là các thuật toán đòi hỏi lượng truyền thông ít nhất.

- **Phân phối dữ liệu thông minh và Phân phối lai**

Eui-Hong Han và các cộng sự đã đề xuất hai phương pháp phát hiện luật kết hợp dựa trên phân phối dữ liệu. Họ quan sát thấy thuật toán phân phối dữ liệu sử dụng cách truyền thông báo giữa tất cả các bộ xử lý rất tốn kém để gửi các phần dữ liệu cục bộ tới mọi bộ xử lý khác. Hơn nữa, mặc dù phân phối dữ liệu chia các ứng viên đồng đều trên giữa các bộ xử lý, nó không phân chia được công việc trong mỗi giao dịch. Tức là nó vẫn tạo một tập con của giao dịch và xác định xem liệu cây băm có chứa tập con đó không.

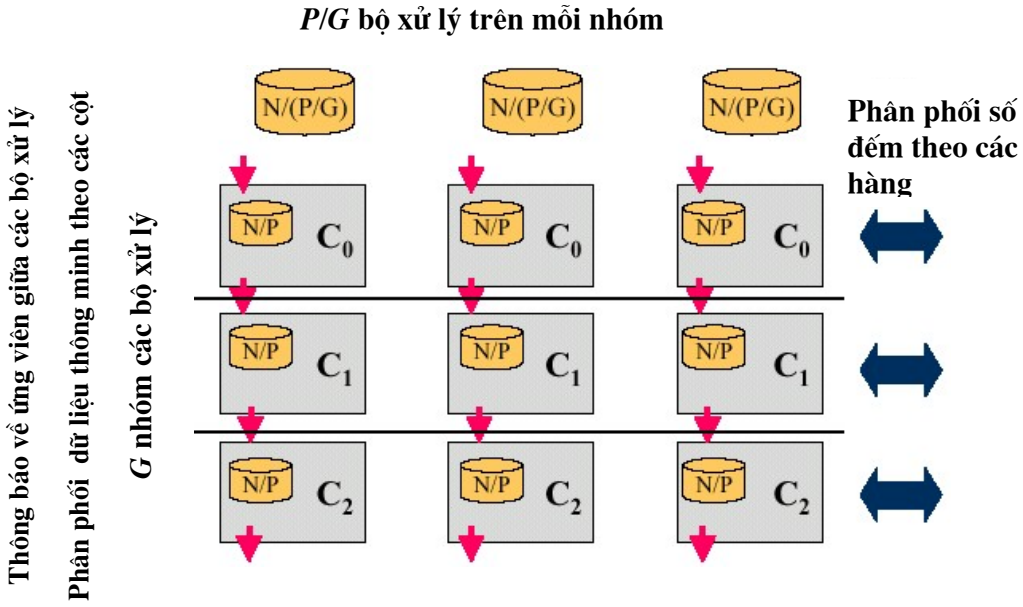
Trong cách phân phối dữ liệu thông minh, Han và cộng sự đã dùng cách truyền thông giữa các bộ xử lý theo vòng tròn, tuyến tính về thời gian. Họ chuyển sang cách phân phối số đếm mỗi khi các ứng viên vừa trong bộ nhớ. Thay vì phân chia ứng viên theo vòng tròn, họ dùng cách phân chia dựa trên tiền tố, cho từng thuộc tính một. Trước khi xử lý một giao dịch, họ đảm bảo rằng nó chứa các tiền tố tương ứng. Nếu không, giao dịch này có thể bị bỏ qua. Toàn bộ cơ sở dữ liệu vẫn giao tiếp với nhau, nhưng một giao dịch có thể không được xử lý nếu nó không chứa các thuộc tính tương ứng.

THUẬT TOÁN PHÂN PHỐI DỮ LIỆU THÔNG MINH



Thuật toán phân phối lai kết hợp cả phân phối số đếm và phân phối dữ liệu thông minh. Nó phân chia P bộ xử lý thành G nhóm kích thước bằng nhau, mỗi nhóm được coi là một bộ siêu xử lý. Phân phối số đếm được dùng giữa G bộ siêu xử lý, và P/G bộ xử lý trong mỗi nhóm dùng cách phân phối dữ liệu thông minh. Cơ sở dữ liệu được phân chia theo hàng ngang giữa G bộ siêu xử lý, các ứng viên được phân chia giữa P/G bộ xử lý trong mỗi nhóm. Thêm vào đó, phân phối lai điều chỉnh số các nhóm một cách linh hoạt trong mỗi lần lặp. Các ưu điểm của cách phân phối lai là nó giảm được chi phí truyền thông trên cơ sở dữ liệu còn 1/G lần, và nó luôn giữ cho các bộ xử lý bận rộn, đặc biệt trong các lần lặp sau. Các thí nghiệm cho thấy là, trong khi phân phối lai có cùng hiệu suất với phân phối số đếm, nó có thể xử lý lượng dữ liệu lớn hơn rất nhiều.

THUẬT TOÁN PHÂN PHỐI LAI



- **Phát hiện phân tán nhanh (FDM)**

David Cheung và cộng sự đề xuất thuật toán phân tán nhanh để phát hiện luật kết hợp. Sự khác biệt chính giữa khai phá dữ liệu song song và phân tán là băng thông và độ trễ trên mạng, ngoài ra, các khác biệt khác là không rõ nét. Với một mạng chậm, bất cứ biển đổi nào của cách phân phối dữ liệu đều không có giá trị thực tế do chúng phải truyền thông toàn bộ cơ sở dữ liệu trong mỗi lần lặp. Do thuật toán phân phối số đếm không tốn nhiều chi phí cho truyền thông, nó là cách lý tưởng để làm cơ sở cho các phương pháp khác trong môi trường phân tán.

Khai phá phân tán nhanh dựa trên phân phối số đếm và đưa ra các kỹ thuật mới để giảm số các ứng viên cần xem xét khi đếm, theo cách đó nó giảm thiểu sự truyền thông. Thuật toán này giả sử rằng cơ sở dữ liệu được phân chia theo chiều ngang giữa các trạm phân tán. Vì tất cả các itemset phổ biến toàn phần đều phải là itemset phổ biến địa phương tại mỗi trạm, các ứng viên duy nhất mà các trạm phải xem xét phải được tạo nên từ cả các ứng viên phổ biến địa phương và phổ biến toàn

phần (ký hiệu là GL_i cho trạm thứ i). Ví dụ, trên tất cả các thuộc tính phổ biến F_i = {*A, B, C, D, E*}, đặt GL_1 = {*A, B, C*} và GL_2 = {*C, D, E*}. Khi đó trạm đầu tiên chỉ xét các ứng viên CG_1 = {*AB, AC, BC*} và CG_2 = {*CD, CE, DE*}. Thay vì sáu ứng viên này, thuật toán phân phối số đếm sẽ tạo ra 5 x 2 = 10 ứng viên. Khai phá phân tán nhanh cũng đề nghị ba cách tối ưu hóa: tỉa cục bộ, tỉa toàn phần và kiểm số đếm.

Trong khai phá phân tán nhanh dùng cách tỉa cục bộ và kiểm số đếm. Mỗi trạm tạo các ứng viên sử dụng GL_i từ tất cả các trạm và gán một trạm chủ cho mỗi ứng viên. Sau đó, mỗi trạm tính độ hỗ trợ địa phương cho các ứng viên. Tiếp theo là bước tỉa: loại bất cứ itemset X nào không phổ biến địa phương tại trạm hiện tại, vì nếu X là phổ biến toàn phần, thì nó sẽ phải xuất hiện tại một số trạm khác.

Bước tiếp theo là sự tối ưu hóa việc kiểm số đếm. Mỗi trạm chủ yêu cầu tất cả các ứng viên được gán cho nó số đếm cục bộ từ tất cả các trạm khác và tính độ hỗ trợ toàn phần của chúng. Trạm chủ sẽ thông báo độ hỗ trợ toàn phần tới tất cả các trạm khác. Cuối cùng, mỗi trạm có tập phổ biến toàn phần và bắt đầu một vòng lặp mới. Thuật toán phân phối số đếm thông báo các số đếm địa phương của tất cả các ứng viên tới tất cả các trạm khác, trong khi khai phá phân tán nhanh chỉ gửi chúng tới một trạm chủ của mỗi ứng viên. Vì thế, phương pháp này giảm lượng truyền thông rất nhiều, và việc cắt tỉa địa phương thậm chí còn giảm nó thấp hơn.

Một cách tối ưu hóa khác được đề xuất là tỉa toàn bộ. Ngoài việc gửi độ hỗ trợ toàn phần của các itemset phổ biến, cách này gửi cả độ hỗ trợ địa phương tại mỗi phần vào cuối vòng lặp thứ (*k-1*). Với vòng lặp tiếp theo, nếu *X* là một ứng viên thì độ hỗ trợ địa phương của tất cả các tập con (*k-1*) phân tử của nó đã có sẵn. Ta có thể thay thế cận trên của độ hỗ trợ của *X* tại trạm *i* bằng:

$$UB(X) = X.sup_i + \Sigma^{s}_{j=1, j\neq i}\, ub_j(X)$$

với *s* là số trạm, $ub_j(X)$ là độ hỗ trợ địa phương tối thiểu của tập con *(k-1)* phần tử bất kỳ của X tại trạm *j* (cận trên của độ hỗ trợ địa phương của X tại trạm *j*). Nếu *UB(X)*

nhỏ hơn độ hỗ trợ tối thiểu, ta có thể loại *X*. Các tác giả đánh giá thuật toán này trên một nhóm 6 máy trạm kết nối bằng Ethernet LAN 10-Mbyte, các thí nghiệm chỉ kiểm tra việc cắt tỉa địa phương với việc kiểm tra số đếm, đã cho thấy rằng có thể giảm được 75%-90% kích thước của tập các ứng viên trên mỗi trạm, và giảm được 85%-90% kích thước các thông báo.

- **Phát hiện song song nhanh (FPM)**

Vấn đề trong cơ chế đếm của bài toán phát hiện phân tán nhanh là nó cần 2 lần gửi thông báo trong mỗi vòng lặp: một lần để tính độ hỗ trợ toàn phần, một lần để thông báo các itemset phổ biến. Sơ đồ hai vòng này có thể làm giảm hiệu suất trong thiết lập song song. Phát hiện song song nhanh tạo ít ứng viên hơn và giữ lại các bước cắt tỉa toàn phần và cắt tỉa bộ phận. Nhưng thay vì kiểm tra các số đếm và sau đó thông báo các itemset phổ biến, nó chỉ thông báo độ hỗ trợ địa phương tới tất cả các bộ xử lý.

Một khía cạnh thú vị hơn của cách tiếp cận này là các tác giả dùng một ma trận để lưu độ nghiêng của dữ liệu (sự phân bố các itemset trên các phần khác nhau). Với itemset *X*, ký hiệu *pX(i)* là xác suất *X* xuất hiện trong phần *i*, khi đó entropy của *X* được cho bởi: $H(X) = -\sum_{i=1}^{n} pX(i)\log(pX(i))$

Số entropy đo độ phân bố của các số đếm độ hỗ trợ địa phương của X trong tất cả các phần. Độ nghiêng của một itemset X được cho bởi:

$$S(X) = \frac{H_{\max} - H(X)}{H_{\max}}$$

với $H_{max} = log(n)$ với *n* phần. S(*X*) = 0 nếu *X* có độ hỗ trợ như nhau trong tất cả các phần, bằng 1 nếu *X* chỉ xuất hiện trong một phần. Độ nghiêng dữ liệu toàn phần của cơ sở dữ liệu là tổng độ nghiêng của tất cả các itemset tính bằng độ hỗ trợ của chúng. Trên thực tế, ta chỉ cần xem xét độ nghiêng của các itemset phổ biến. Thí

nghiệm của các tác giả trên 32-nodes IBM SP2 cho thấy rằng phát hiện song song nhanh có thể tốt hơn thuật toán phân phối số đếm 3 lần với các tập dữ liệu có độ nghiêng rất lớn, và gấp 1.5 lần với các tập dữ liệu có độ nghiêng nhỏ.

III.2.2. Các hệ chia sẻ bộ nhớ

Bài toán thiết kế chính trong các hệ thống chia sẻ bộ nhớ liên quan tới việc giảm thiểu các lỗi chia sẻ và duy trì sự tốt sự định hướng dữ liệu. Các nền hệ thống chia sẻ bộ nhớ thường không được chú ý nhiều trong lĩnh vực phát hiện luật kết hợp. Tuy nhiên, với sự phát triển của các máy tính đa xử lý và các clump, các nền hệ thống chia sẻ bộ nhớ đang trở nên ngày càng quan trọng.

- ***Thuật toán dựa trên Apriori***

Một trong các thuật toán cho các hệ chia sẻ bộ nhớ là *ứng viên chung-cơ sở dữ liệu được phân chia* (CCPD). CCPD sử dụng cách tiếp cận dữ liệu song song. Cơ sở dữ liệu được phân chia một cách hợp lý thành các đoạn bằng nhau, và tất cả các bộ xử lý đồng thời xử lý một cây băm toàn phần hoặc cây băm các ứng viên phổ biến. CCPD song song hóa quá trình tạo ứng viên. Mỗi bộ xử lý tạo một tập con rời nhau các ứng viên, dẫn tới sự phân chia tốt về mặt tính toán. Để xây dựng song song một cây băm. CCPD kết hợp một khóa với mỗi nút lá. Khi một bộ xử lý muốn thêm một ứng viên vào cây, nó bắt đầu từ gốc và liên tiếp băm trên các ứng viên cho tới khi đạt tới lá. Sau đó nó có được khóa và chèn ứng viên đó. Với cơ chế khóa này, mỗi bộ xử lý có thể chèn các itemset trong các phần khác nhau của cây băm một cách song song. Để đếm độ hỗ trợ, mỗi bộ xử lý tính tần số từ phần logic của nó.

Các tác giả cũng đề xuất khả năng tối ưu hóa, như là kết hợp vòng ngắn và cân bằng cây băm. Kết hợp vòng ngắn sử dụng phép đánh dấu bit trên cây băm để tránh việc xử lý cây con đã được xử lý từ trước. Họ dùng một hàm băm mới để cân bằng cây băm, bởi vì một hàm *mod* đơn giản có thể dẫn tới các cây bị nghiêng. Một cây cân bằng sẽ tăng tốc các tiến trình, vì nó có có chiều cao ngắn hơn. Thuật toán

CCPD có được sự tăng tốc đáng kể nhưng quá trình nhập/xuất lại không có lợi cho hiệu suất.

Thuật toán *ứng viên được phân chia-cơ sở dữ liệu chung* cũng được thực hiện, trong đó các bộ xử lý tạo nên các cây ứng viên rời nhau và duyệt toàn bộ cơ sở dữ liệu để tính độ hỗ trợ. Tuy nhiên, gánh nặng nhập/xuất và sự tranh chấp về không gian là không thể chấp nhận được, nó khiến nhiều bộ xử lý bị chậm lại. Do bản chất của phép băm, các cây băm các ứng viên định vị dữ liệu rất kém. Hơn nữa, một cây dùng chung có thể dẫn tới chia sẻ sai trong pha tính độ hỗ trợ. Nhiều cơ chế và chính sách được đề xuất để điều chỉnh cách bố trí bộ nhớ của hàm băm dựa trên viện truy nhập các mẫu để tính độ hỗ trợ. Sơ đồ này đảm bảo rằng các nút có khả năng được truy cập kế tiếp nhau sẽ được đặt gần nhau về mặt vật lý, điều này giúp định vị tốt dữ liệu. Ngoài ra còn cơ chế tư nhân hóa, mỗi bộ xử lý tập hợp các số đếm từ một mảng địa phương, tiếp theo là rút gọn tổng để giảm lỗi chia sẻ.

- ***Thuật toán dựa trên DIC***

Cheung và cộng sự đề xuất thuật toán phát hiện song song không đồng bộ (APM) dựa trên DIC. Thuật toán này sử dụng kỹ thuật tỉa toàn phần của thuật toán phát hiện phân tán nhanh để giảm kích thước các ứng viên 2-itemset. Việc cắt tỉa này hiệu quả nhất khi giữa các phần có độ lệch dữ liệu lớn. Tuy nhiên, DIC đòi hỏi rằng các phần phải đồng nhất. Phát hiện song song không đồng nhất chia cơ sở dữ liệu một cách hợp lý thành các phần ảo nhỏ, kích thước bằng nhau. Số các phần ảo l độc lập với số bộ xử lý p, thường thì $l \geq p$. Gọi m là số các thuộc tính, APM tập hợp các số đếm địa phương của các m thuộc tính trong mỗi phần. Nó tạo ra tập dữ liệu lxm, với l véctơ độ hỗ trợ các thuộc tính và không gian m chiều. APM chia l véctơ vào k cluster, tối đa hóa khoảng cách giữa các cluster và giảm thiểu khoảng cách trong mỗi cluster. Vì thế, k cluster có độ lệch tối đa và chúng được dùng để tạo các tập ứng viên 2-itemset nhỏ.

Tiếp theo APM áp dụng song song DIC, ý tưởng là chia cơ sở dữ liệu thành *p* phần đồng nhất. Mỗi bộ xử lý áp dụng độc lập thuật toán DIC trên phần cục bộ của mình. Tuy nhiên, có một cây tiền tố được xây dựng không đồng bộ được chia sẻ giữa các bộ xử lý. APM dừng khi tất cả các bộ xử lý đã đã xử lý hết các ứng viên tạo bởi nó hay bởi bộ xử lý khác, và khi không có ứng viên mới được tạo thêm. Để áp dụng DIC trên mỗi phần, mỗi bộ xử lý phải chia phần cục bộ của nó thành *r* phần con. Hơn nữa, DIC đòi hỏi rằng cả *p* phần giữa các bộ xử lý và *r* phần trong mỗi bộ xử lý phải càng đồng nhất càng tốt. APM đảm bảo rằng p phần là đồng nhất bằng cách gán các phần ảo từ mỗi cluster trong *k* cluster của vòng lặp đầu tiên theo kiểu quay vòng giữa *p* bộ xử lý. Do đó, mỗi bộ xử lý có sự kết hợp các phần ảo như nhau từ các cluster riêng biệt, cho ra sự phân chia các bộ xử lý đồng nhất.

Để có tính đồng nhất giữa các phần trong mỗi bộ xử lý, APM thực hiện phân *k* nhóm lần thứ hai. Chúng nhóm *r* phần vào *k* cluster, và lại gán các phần tử từ mỗi cluster vào *r* phần theo cách quay vòng. Các thí nghiệm trên 12-node Sun Enterprise 4000 chia sẻ bộ nhớ cho thấy APM thực hiện tốt hơn các thuật toán Phân phối số đếm/CCPD từ 4-5 lần. Một sự thỏa hiệp thú vị trong APM là mặc dù độ lệch dữ liệu là tốt cho để cắt tỉa toàn phần, nó lại không tốt để cân bằng tải.

III.2.3. Các hệ phân cấp

Một hệ thống phân cấp có cả các phần với bộ nhớ phân tán và bộ nhớ được chia sẻ. Các hệ thống phân cấp đang trở nên ngày càng phổ biến, đặc biệt với sự phát triển của các máy tính để bàn đa xử lý và của các mạng tốc độ cao. Các nhóm này cung cấp khả năng mở rộng và có hiệu suất ngang với các máy đắt tiền, nhưng với giá thành rẻ. Trong một hệ thống phân cấp, ta phải tối ưu hóa truyền thông giữa các nút và phân tách dữ liệu và tối ưu hóa sự định vị dữ liệu trong mỗi nút và tránh lỗi chia sẻ cho mỗi nút chia sẻ bộ nhớ.

- ***Thuật toán dựa trên Eclat***

Bốn thuật toán ParEclat, ParMaxEclat, ParClique va ParMaxClique được phát triển dựa trên bốn thuật toán tuần tự tương ứng. Thuật toán đang xét giả sử rằng hệ thống có *n* máy chủ, mỗi máy chủ gồm *p* nút chia sẻ bộ nhớ, cơ sở dữ liệu được định dạng theo chiều dọc và được phân chia giữa các máy chủ sao cho mỗi máy chủ có toàn bộ danh sách định danh *tidlist* của tất cả các thuộc tính đơn lẻ. Tổng chiều dài của các *tidlist* cục bộ là xấp xỉ bằng nhau trên tất cả các máy chủ.

Cả 4 thuật toán đều có cách song song hóa tương tự và chỉ khác nhau ở chiến lược tìm kiếm, có ký thuâth phân lớp giống nhau. Mỗi thuật toán đều có 3 pha chính:

- Pha nạp giá trị: thực hiện tính toán và phân chia dữ liệu
- Pha không đồng bộ: mỗi bộ xử lý độc lập tạo các itemset phổ biến
- Pha rút gọn: kết hợp các kết quả cuối cùng.

Trong pha đầu tiên, máy chủ tạo tiền tố hoặc các lớp cân bằng dựa trên clique, dùng các 2-itemset phổ biến. Tiếp đó một thuật toán xếp các lớp này vào các bộ xử lý sẵn có. Mỗi lớp có một độ đo dựa trên số các phần tử của nó. Sau đó thuật toán xếp lịch sẽ sắp xếp các lớp theo độ đo và gán lớp có độ đo lớn nhất cho bộ xử lý có tổng độ đo nhỏ nhất, và lặp lại quá trình này cho các lớp theo thứ tự được sắp. Sau khi sắp xếp xong lớp cha, các danh sách định danh đối tượng *tidlist* được sao chép một cách chọn lọc trên mỗi máy chủ, nhờ thế tất cả các *tidlist* là một phần của lớp được gán trên một bộ xử lý sẽ có sẵn trên ổ đĩa cục bộ của các máy chủ. Chỉ có các máy chủ tham gia vào qua trình truyền thông này.

Trong pha thứ hai, mỗi bộ xử lý có sẵn các lớp được gán cho nó, cùng danh sách định danh của tất cả các thuộc tính. Vì thế, mỗi bộ xử lý có thể độc lập tạo tất cả các itemset phổ biến từ các lớp của mình. Trong pha này không cần đến sự truyền thông và đồng bộ hóa. Hơn nữa, toàn bộ bộ nhớ hệ thống là sẵn có để sử dụng,

không cần lưu trong bộ nhớ cây tiền tố hoặc cây băm. Chỉ cần đến các thao tác đơn giản để đếm các itemset.

Bốn thuật toán khác nhau phụ thuộc vào chiến lược phân tách và tìm kiếm được dùng. ParEclat và ParMaxEclat dùng các lớp dựa trên tiền tố, và dùng chiến thuật tìm kiếm từ dưới lên và tìm kiếm lai. ParClique và ParMaxClique thì dùng các lớp nhỏ dựa trên clique, cũng tương ứng dùng cách tìm kiếm từ dưới lên và tìm kiếm lai.

Bảng dưới đây cho thấy sự khác biệt cơ bản giữa các phương pháp khác nhau và nhóm các thuật toán có liên quan với nhau. Ta cũng thấy là chỉ có một số ít các mô hình khác nhau. Nhiều thuật toán đề xuất sự tối ưu hóa cho các thuật toán khác. Vì thế, các phương pháp song song này có cùng độ phức tạp và các tính chất của các thuật toán tuần tự cơ sở của nó.

Thuật toán	**Đặc điểm**
Phân phối số đếm	Dựa trên Apriori
PEAR	Cây tiền tố các ứng viên
PDM	Bảng băm cho các 2-itemset, tạo ứng viên song song
NPA	Chỉ các máy chủ thực hiện việc rút gọn số đếm
FDM	Cắt tỉa cục bộ và toàn cục, kiểm số đếm
FPM	Cắt tỉa cục bộ và toàn cục, xử lý độ nghiêng của dữ liệu
CCPD	Chia sẻ bộ nhớ
Phân phối dữ liệu	Trao đổi toàn bộ cơ sở dữ liệu trong mỗi lần lặp
SPA	Giống như phân phối dữ liệu
IDD	Truyền thông báo theo vòng tròn, phân đoạn ứng viên dựa trên thuộc tính
PCCD	Chia sẻ bộ nhớ (trao đổi cơ sở dữ liệu logic)
Phân phối lai	Kết hợp phân phối số đếm và phân phối dữ liệu
Phân phối ứng viên	Lặp lại cơ sở dữ liệu một cách chọn lọc, không đồng bộ
HPA	Không lặp lại cơ sở dữ liệu, trao đổi các itemset
HPA-ELD	Lặp lại các itemset phổ biến
ParEclat	Dựa trên Eclat, không đồng bộ, cấu trúc phân cấp
ParMaxEclat	Dựa trên MaxEclat, không đồng bộ, cấu trúc phân cấp
ParClique	Dựa trên Clique, không đồng bộ, cấu trúc phân cấp
ParMaxClique	Dựa trên MaxClique, không đồng bộ, cấu trúc phân cấp

APM	Dựa trên DIC, chia sẻ bộ nhớ, không đồng bộ
PPAR	Dựa trên phân đoạn, cơ sở dữ liệu theo chiều ngang

III.3. MÔ HÌNH TẬP THÔ PHÁT HIỆN SONG SONG LUẬT KẾT HỢP

Chương 2 đã đề cập tới hai thuật toán phát hiện luật kết hợp theo cách tiếp cận của lý thuyết tập thô. Các tác giả [16] có nhận xét rằng thuật toán 2.1 không thích hợp đối với các cơ sở dữ liệu (bảng quyết định) với số lượng thuộc tính lớn. Trong thực tế giả thiết này là rất khó chấp nhận vì vậy các tác giả cho rằng cần có giai đoạn tiền xử lý trước khi áp dụng các thuật toán. Thuật toán 2.2 với mục tiêu tìm tập tối ưu luật kết hợp là một trong các giải pháp được đề xuất. Trong phần này, phát triển các ý tưởng từ [16], chúng tôi xây dựng một mô hình phát hiện song song luật kết hợp theo cách tiếp cận tập thô. Mô hình này dựa trên một số vấn đề liên quan đến mô hình phát hiện luật kết hợp. Trước hết chúng tôi xin đề cập tới một ví dụ xuất phát từ thực tế tại Sở Y tế Hà Nội.

Bắt đầu từ năm 2001, Sở Y tế Hà Nội có kế hoạch xây dựng hệ thống thông tin của toàn ngành và của các bệnh viện do Sở quản lý bao gồm các thông tin quản lý và các thông tin chuyên môn [3]. Sở Y tế Hà Nội quản lý hệ thống gồm 42 bệnh viện trên địa bàn Hà Nội, bao gồm cả các bệnh viện đa khoa và chuyên khoa mà theo chức năng mỗi bệnh viện chữa trị một chuyên khoa hoặc đa khoa, và còn được phân bố theo lãnh thổ (các bệnh viện quận, huyện). Cơ sở dữ liệu khám và điều trị bệnh của hệ thống toàn Sở được phân tán theo hệ thống 42 bệnh viện nói trên. Một trong những yêu cầu được đặt ra ở đây là sử dụng được các dữ liệu bệnh án sẵn có để đưa ra các luật cho thấy mối liên hệ giữa các triệu chứng của bệnh nhân và khả năng bị bệnh nào đó của họ. Các luật này bao gồm các luật cục bộ (cho mỗi bệnh viện) và luật toàn bộ, không chỉ áp dụng cho một bệnh viện nào đó mà còn phải đúng để áp dụng cho toàn bộ Thủ đô Hà Nội. Luật cục bộ (hy vọng nhận được) liên quan đến đặc thù của từng loại bệnh (đối với các bệnh viện chuyên khoa) hoặc liên

quan đến đặc thù của từng vùng lãnh thổ (quận - thành thị và huyện - nông thông, mức sống cao và mức sống thấp ...). Luật toàn cục (hy vọng nhận được) liên quan đến chương trình chung trong toàn Hà Nội để đưa ra các chính sách dự phòng, chăm sóc sức khoẻ ban đầu ... cũng như các phòng chống chung đối với mỗi loại bệnh. Bài toán trên được phát biểu dưới dạng tập thô trong bảng quyết định theo quan điểm của Pawlak như dưới đây.

Trong trường hợp dữ liệu cục bộ được trình bày dưới dạng một hệ thông tin theo quan điểm của Pawlak và sử dụng các thuật toán trong chương 2 [16], chúng ta cần tìm mô hình cho phép mô tả vấn đề phát hiện tập phổ biến toàn cục và tập phổ biến cục bộ. Dưới đây là những nét sơ bộ nhất về mô hình như vậy.

Phát biểu các nội dung trên đây theo cách diễn đạt trong hệ thông tin như sau: Cho các hệ thông tin $S_i = (O_i, A_i, V, \sigma_i)$ trong đó với $i \neq j$ thì $O_i \cap O_j = \varnothing$, cho phép $A_i \cap A_j \neq \varnothing$ và hạn chế xét $V=\{0, 1\}$ (Giả thiết V hạn chế không ảnh hưởng đến hoạt động của mô hình và thuật toán - xem thuật toán 2.1 và 2.2; ở đây, có giả thiết như vậy cho đơn giản). Như vậy mỗi hệ thông tin S_i là một hệ thông tin cục bộ, chứa dữ liệu về các bệnh án tại bệnh viện thứ i, mỗi đối tượng $o \in O_i$ là một phiếu khám bệnh. Đặt $O = \cup O_i$, $A = \cup A_i$, xây dựng hệ thông tin $S = (O, A, V, \sigma)$, trong đó σ được xác định như sau:

$$\sigma(o,a) = \begin{cases} \sigma_i(o,a) & khi\ o \in O_i, a \in A_i \\ 0 & \neq \end{cases}$$

Theo quan điểm của hệ phân tán, hệ thông tin S_i (hệ thông tin tại bệnh viện do Sở Y tế Hà Nội quản lý) nhận được từ hệ thông tin S (hệ thông tin toàn bộ Sở Y tế Hà Nội) theo phân đoạn vừa ngang vừa dọc (đặc biệt khi mọi $A_i = A$ thì chúng ta có phân đoạn ngang, mọi $O_i = O$ thì chúng ta có phân đoạn dọc). Giả sử, chúng ta sử

dụng thuật toán phát hiện luật kết hợp đối với mỗi hệ thông tin S_i. Một vấn đề được quan tâm là mối quan hệ giữa luật kết hợp trong S với các luật đã phát hiện từ trước trong các S_i.

Có thể xem xét hai mô hình xử lý song song đối với :

- Mô hình tập trung: Phát hiện luật kết hợp mà dữ liệu đã tập trung tại một hệ thông tin thống nhất. Theo mô hình này chú ý đến việc chia xẻ bộ nhớ, nhiều bộ dữ liệu cùng được đưa vào bộ nhớ để xử lý. Trong trường hợp này, các hệ thông tin con thực chất được tách ra từ một hệ thông tin tập trung.
- Mô hình phân tán: Dữ liệu tại các hệ thống con S_i là phân tán thực sự. Việc phát hiện luật kết hợp song song không chỉ thực hiện đối với mỗi hệ con mà còn cần phát hiện luật kết hợp cho toàn bộ hệ tổng thể.

Các phần trình bày dưới đây giới thiệu các giải pháp ở mức độ sơ lược nhất liên quan đến các nội dung trên.

III.2.1. Thuật toán 3.1. (Mô hình tập trung)

Kết hợp các gợi ý của [16] khi xem xét thuật toán 1 chương 2, và khảo sát hệ thống *Data Surveyor* trong [8], chúng ta đưa ra thuật toán sau đây nhằm phát hiện song song luật kết hợp. Trừ bước tiền xử lý, bước tách hệ thông tin và bước hợp nhất kết quả, nội dung các bước còn lại tương ứng như mô tả trong thuật toán 2.1.

Thuật toán 3.1: **Tìm tập tối ưu các luật**

Input: Hệ thông tin S gồm n đối tượng trong một tập đối tượng O, mối đối tượng u có thể có m thuộc tính.

Output: Tập tối ưu các luật cùng độ mạnh của mỗi luật

Nội dung thuật toán

Bước 1: Phân nhóm đối tượng thành các nhóm dựa theo chỉ tiêu của các thuộc tính bằng cách thực hiện một thuật toán phân nhóm: $O = \cup O_i$, $A = \cup A_i$ với chú ý là tập đối tượng cũng như tập thuộc tính trong mỗi hệ thông tin thành phần không nhất thiết rời nhau. Ghi nhận thông tin trọng số của mỗi hệ thông tin thành phần (có thể chọn là số đối tượng có trong O_i).

Bước 2: Thực hiện song song thuật toán 2.1 với dữ liệu đầu vào là các hệ thông tin thành phần S_i. Kết quả nhận được qua bước này là các luật kết hợp cục bộ trong mỗi hệ thông tin thành phần.

Quá trình thực hiện *bước 2* được tiến hành là việc kết hợp các nội dung trong thuật toán 2.1 và mô hình tính toán song song trong [8].

Bước 3: Hợp nhất kết quả thực hiện được trong *bước 2* với các trọng số đối với mỗi hệ thông tin thành phần.

III.2.2. Thuật toán 3.2 (Mô hình phân tán)

Trong trường hợp dữ liệu trong hệ thống được phân tán trong các hệ thông tin địa phương thực sự (không có bước tách hệ thông tin) thì thuật toán phân tán được trình bày như sau.

Thuật toán 3.2: **Tìm tập tối ưu các luật**

Input: Tập hợp các hệ thông tin $S_i = (O_i, A_i, V, \sigma_i)$, mỗi S_i gồm n_i đối tượng trong tập đối tượng con O_i, A_i là tập con của A (Tập hợp thuộc tính được thống nhất trong toàn bộ hệ thông tin S; chẳng hạn, Sở Y tế Hà Nội thống nhất bảng mã và tên các thuộc tính này trong toàn bộ Sở).

Output: Tập tối ưu các luật cùng độ mạnh của mỗi luật cục bộ cũng như các luật toàn cục.

Nội dung thuật toán

Bước 1: Áp dụng thuật toán 2.1. cho các hệ thông tin thành phần S_i, Kết quả nhận được luật kết hợp của mỗi bảng hệ thông tin thành phần cùng một đại lượng là trọng số của mỗi hệ thông tin thành phần đó.

Bước 2: Hợp nhất luật kết hợp từ các hệ thông tin thành phần theo trọng số đã có để nhận được các luật kết hợp toàn cục.

Kết quả của bước này bao gồm hai loại luật kết hợp:

- Các luật kết hợp toàn cục sau khi hợp nhất ở *bước 2*,
- Lớp các luật kết hợp cục bộ là kết quả của *bước 1*.

Chúng tôi đề xuất ý nghĩa của các khái niệm "trọng số" trong kết quả bước 1 và khái niệm "hợp nhất" khi thực hiện bước 2 như trình bày dưới dây.

Kết quả áp dụng bước 1 đối với S_i là (coi hai thành phần đầu tiên là trọng số):

- Tập S_i các thuộc tính trong S_i,
- Số n_i số lượng các đối tượng có trong S_i,
- {các luật phát hiện được qua bước 1 đối với S_i}. Chúng tôi quan niệm rằng mỗi luật ở đây bao gồm các thành phần:

- Luật với độ hỗ trợ, độ tin cậy tìm được,
- Tập các thuộc tính A^*_i xuất hiện trong luật,
- Số lượng n_i đối tượng trong S_i,

Chú ý rằng, một luật có thể được phát hiện trong nhiều hệ thông tin thành phần với các độ đo hỗ trợ và tin cậy khác nhau.

Biểu thức sau đây trình bày nội dung hợp nhất để nhận được các luật kết hợp toàn cục (μ áp dụng tính toán cho mỗi đại lượng hoặc độ hỗ trợ hoặc độ tin cậy) từ các luật kết hợp cục bộ $X \rightarrow Y$:

$$\mu(X \rightarrow Y) = \frac{\sum_{(X \cup Y) \subseteq A_i} (n_i * \mu_{S_i}(X \rightarrow Y))}{\sum_{(X \cup Y) \subseteq A_i} n_i} \qquad (3.1)$$

Công thức trên được giải thích như sau: Với một luật X→Y nào đó phát hiện ở bước 1, để hợp nhất chúng ta xem xét:

- Các hệ thông tin thành phần S_i mà A_i chứa X∪Y. Việc hợp nhất chỉ liên quan đến các hệ thông tin thành phần này,

- Với mỗi hệ thông tin thành phần trên đây, luật X→Y có đại lượng μ có giá trị được ký hiệu là $\mu_{S_I}(X \rightarrow Y)$ được cho bởi kết quả bước 1 hoặc bằng 0 nếu như không là kết quả của bước 1.

- Tính toán $\mu(X \rightarrow Y)$ toàn cục như công thức đã cho.

- So sánh độ hỗ trợ và độ tin cậy với ngưỡng để quyết định về việc có kết luận $X \rightarrow Y$ là luật toàn cục hay không.

Nhận xét: 1. Với đề xuất trên đây, có thể để xẩy ra tình huống "bỏ sót" luật kết hợp toàn cục, xuất phát từ lý do bước 1 đã loại bỏ một số luật cục bộ dưới ngưỡng vì vậy chúng không được tính toán trong công thức 3.1. Điều này có thể khắc phục bằng cách giảm ngưỡng một cách thích hợp khi khai phá luật kết hợp tại các hệ thông tin thành phần trong bước 1 để hợp nhất trong bước 2 và chú ý rằng bổ sung ngưỡng mới cho luật kết hợp cục bộ.

2. Thuật toán 3.1 và 3.2. không chỉ thực hiện song song đối với các bảng quyết định thành phần mà trong nhiều trường hợp, do việc phân nhóm, số thuộc tính

trong các bảng quyết định thành phần đã giảm đi nhiều so với bảng quyết định chung cho nên độ phức tạp tính toán tổng cộng được giảm đi đáng kể.

KẾT LUẬN CHƯƠNG 3

Lượng dữ liệu bùng nổ trong các hệ thông tin cùng với sự phát triển của các cơ sở dữ liệu trực tuyến đã thúc đẩy nhu cầu về khai phá dữ liệu song song và phân tán. Tính toán song song sẽ góp phần giảm bớt thời gian và chi phí xử lý, cho hệ thống khả năng phát triển. Nhiều thuật toán phát hiện song song luật kết hợp được phát triển dựa trên các thuật toán tuần tự cho các nền phần cứng khác nhau. Các thuật toán này được tổng kết và so sánh bởi Zaki [17], cung cấp một cái nhìn khái quát về sự phát triển của các mô hình phát hiện song song luật kết hợp *(mục 3.2)*. Trên cơ sở các thuật toán tìm hiểu được đã nêu ở chương 2, chúng tôi đề xuất mô hình phát hiện song song luật kết hợp theo cách tiếp cận tập thô cho hệ thông tin, với việc song song hóa được thực hiện trên các bước dữ liệu cho các mô hình tập trung và phân tán. Theo cách tiếp cận này, các luật tìm được trong các hệ thông tin con được sử dụng để tìm ra các luật có giá trị trên toàn hệ thống tổng thể, có sử dụng giá trị trọng số cho mỗi hệ con. Chúng tôi cũng đưa ra một công thức để hợp nhất các luật kết hợp cục bộ để nhận được luật kết hợp toàn cục (công thức 3.1).

PHẦN KẾT LUẬN

Sau một thời gian thu thập tài liệu, khảo sát và phân tích nội dung về việc phát hiện song song luật kết hợp theo cách tiếp cận tập thô, luận văn đã đạt được những kết quả như sau:

- Trình bày được những nội dung cơ bản nhất của một trong những lĩnh vực nghiên cứu và triển khai thời sự hiện nay là khai phá dữ liệu và phát hiện tri thức trong các cơ sở dữ liệu mà luật kết hợp là một trong những tri thức điển hình,

- Cùng với việc trình bày các phương pháp khai phá dữ liệu điển hình, luận văn cũng định hướng vào nội dung biểu diễn và khai phá luật kết hợp theo cách tiếp cận tập thô. Những kết quả gần đây nhất về nội dung này đã được giới thiệu, phân tích trong luận văn.

- Phát hiện luật kết hợp nói riêng cũng như khai phá dữ liệu nói chung trong những cơ sở dữ liệu lớn là một công việc đòi hỏi thời gian tính toán lớn, vì vậy luận văn đã trình bày một số mô hình, thuật toán liên quan đến việc phát hiện song song luật kết hợp, trong đó đáng chú ý là các thuật toán 2.1 và 2.2.

- Luận văn đã đề xuất sơ bộ một mô hình phát hiện luật kết hợp song song theo hướng tiếp cận tập thô trong hệ thông tin và bảng quyết định, trong đó quan niệm rằng một hệ thông tin tổng quát được tích hợp từ các hệ thông tin thành phần. Thông qua việc định nghĩa tính chất kết hợp luật kết hợp trong mô hình này, luận văn cũng giới thiệu một thuật toán sơ bộ về phát hiện song song luật kết hợp trong mô hình như vậy. Luận văn cũng đề xuất được công thức tính toán các đặc trưng của luật kết hợp toàn cục từ các luật kết hợp cục bộ (công thức 3.1) nhằm hoàn chỉnh thuật toán 3.2. Luận văn cũng đưa ra nhận xét về tính hợp lý của công thức tính toán đó.

Trong quá trình nghiên cứu để hoàn thành luận văn thông qua việc tổng hợp và phân tích nội dung chính yếu về một lĩnh vực hết sức thời sự là phát hiện tri thức mà cụ thể là phát hiện luật kết hợp, thử nghiệm đề xuất sơ bộ một mô hình phát hiện luật kết hợp, chúng tôi nhận thấy hướng nghiên cứu về khai phá dữ liệu song song nói chung và phát hiện luật kết hợp song song nói riêng là một hướng nghiên cứu còn rất rộng lớn và luôn là vấn đề thời sự. Chúng tôi tiếp tục công việc nghiên cứu theo các nội dung sau đây:

- Phát triển mô hình phát hiện luật kết hợp như đã trình bày trong mục 3.3.

- Thử nghiệm thuật toán trong một hệ thống tính toán song song thực sự, trước mắt dựa trên nền của hệ thống PC-cluster tại Bộ môn các Hệ thống Thông tin, khoa Công nghệ, Đại học Quốc gia Hà Nội.

TÀI LIỆU THAM KHẢO

TÀI LIỆU TIẾNG VIỆT

1. Hà Quang Thụy. *Một số vấn đề về không gian xấp xỉ, tập thô đối với hệ thông tin*. Luận án Phó Tiến sĩ Khoa học Toán Lý, 1996
2. Nguyễn Thanh Thủy. *Khai phá dữ liệu: Kỹ thuật và ứng dụng*. Trường thu "Hệ mờ và ứng dụng", 2001
3. Sở Y tế Hà Nội. *Đề cương chi tiết các hạng mục đầu tư công nghệ thông tin tại Sở Y tế Hà Nội năm 2001*.

TÀI LIỆU TIẾNG ANH

4. Rakesh Agrawal, John Shafer. *Parallel Mining of Association Rules*. IBM Almaden Research Center, 1996
5. Rakesh Agrawal, Heikki Mannila, Ramakrishaman Skikant, Hannu Toivonen, A. Inkeri Verkamo. *Fast Discovery of Association Rules*. Advances Knowledge Discovery and Data Mining. AAAI Press/ MIT Press, 1996.
6. Ho Tu Bao, Nguyen Duc Dung. *Integration of Rule Induction and Association Rule Mining*. The 1st Workshop of International Joint Research "Parallel Computing, Data Mining and Optical Networks", 2001
7. Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth. *From Dataming to Knowledge Discovery: An Overview*. Advances Knowledge Discovery and Data Mining. AAAI Press/ MIT Press, 1996.

8. Marcel Holsheimer, Martin L. Kersten, Arno P.J.M. Siebes. *Data Surveyor: Searching the Nuggets in Parallel*. Advances Knowledge Discovery and Data Mining. AAAI Press/ MIT Press, 1996.
9. Boris Kovalerchuk, Evgenii Vityaev. *Data Mining in Finance: Advances in Relational and Hybrid Methods.* Kluwer Academic Publishers, 2001
10. Vipin Kumar, Mohammed Zaki. *High Performance Data Mining*.
11. Milan Milenkovic. *Operating Systems: Concepts and Design*. McGraw-Hill Inc., 1992.
12. Tetsuya Murai, Yoshiharu Sato. *Association Rules from the Point of View of Modal Logic and Rough Set*. The 4th Asian Fuzzy Systems Symposium, 2000
13. S. Parthasarathy, S. Dwarkadas, M. Ogihara. *Active Mining in a Distributed Setting*. SIGKDD Workshop on Large-Scale Parallel KDD Systems, 1999
14. Zdzislaw Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991.
15. D.B. Skilicorn. *Strategies for Parallel Data Mining*. External Technical Report, 1999
16. Andrzej Skowron, Ning Zong. *Rough Sets in KDD.* Tutorial Notes, 2000
17. Mohammed J. Zaki. *Parallel and Distributed Association Mining: A Survey*. IEEE Concurrency, 1999.