Mở đầu

Hơn một thập niên trở lại đây, khai phá dữ liệu (KPDL) đã trở thành một trong những hướng nghiên cứu chính trong lĩnh vực khoa học máy tính và công nghệ tri thức. Hàng loạt nghiên cứu, đề xuất ra đời đã được thử nghiệm và ứng dụng thành công vào đời sống cùng với hơn mười năm lịch sử cho thấy rằng KPDL là một lĩnh vực nghiên cứu ổn định, có một nền tảng lý thuyết vững chắc chứ không phải được xem là "sớm nở tối tàn" như một số ít nhà tin học nghi ngờ *tại thủa ban đầu của lĩnh vực này*.

KPDL bao hàm rất nhiều hướng tiếp cận. Các kỹ thuật chính được áp dụng trong lĩnh vực này phần lớn được thừa kế từ lĩnh vực cơ sở dữ liệu (CSDL), machine learning, trí tuệ nhân tạo, lý thuyết thông tin, xác suất thống kê, và tính toán hiệu năng cao. Các bài toán chủ yếu trong KPDL là phân lớp/dự đoán (classification/prediction), phân cụm (clustering), khai phá luật kết hợp (association rules mining), khai phá chuỗi (sequence mining), v.v. Lĩnh vực này cũng là điểm hội tụ và giao thoa của rất nhiều lĩnh vực khác. KPDL đã và đang được ứng dụng thành công vào thương mại, tài chính & thị trường chứng khoán, sinh học, y học, giáo dục, viễn thông, .v.v.

Ý thức được đây là một lĩnh vực nghiên cứu có nhiều triển vọng, tôi đã chọn hướng nghiên cứu *Khai phá song song luật kết hợp mờ* cho đề tài luận văn của mình. Luận văn được xây dựng dựa trên nền các nghiên cứu đã có trong lĩnh vực khai phá luật kết hợp kể từ năm 1993, đồng thời tôi cũng mạnh dạn trình bày một vài đề xuất của riêng mình mà hai trong số những đề xuất đó là "nêu lên mối liên hệ giữa luật kết hợp mờ và lý thuyết tập mờ" và "thuật toán song song khai phá luật kết hợp mờ".

Luận văn được tổ chức thành 5 chương như sau:

- *Chương I* trình bày tổng quan về KPDL như định nghĩa thế nào là KPDL và khám phá tri thức từ cơ sở dữ liệu, các bước chính trong quá trình khám phá tri thức. Chương này cũng đề cập đến các kỹ thuật và hướng tiếp cận chính trong KPDL và phân loại các hệ thống khai phá theo nhiều tiêu chí khác nhau. Phần cuối của chương này phác họa những ứng dụng chính của

lĩnh vực này và những hướng nghiên cứu đang và sẽ được chú trọng trong thời gian tới.

- *Chương II* trình bày về bài toán "khai phá luật kết hợp". Để đi vào những nghiên cứu cụ thể ở hai chương sau, chương này cung cấp những hiểu biết cần thiết về bài toán khai phá luật kết hợp. Phần cuối chương sẽ là tổng hợp những đề xuất chính trong hơn 10 năm lịch sử tồn tại và phát triển của bài toán này.

- *Chương III* trình bày về "khai phá luật kết hợp mờ". Phần đầu của chương phát biểu lại bài toán khai phá luật kết hợp với thuộc tính số và thuộc tính hạng mục cùng các phương pháp rời rạc hóa dữ liệu cho bài toán này. Dạng luật kết hợp này cùng với các phương pháp rời rạc hóa đi kèm có một vài hạn chế như ngữ nghĩa của luật hay vấn đề "điểm biên gãy". Luật kết hợp mờ được đề xuất như một hướng khắc phục các nhược điểm của bài toán trên. Bên cạnh sự tổng hợp về các nghiên cứu trước đó về dạng luật này, luận văn cũng nêu lên mối liên hệ giữa luật kết hợp và lý thuyết tập mờ và giải quyết câu hỏi "tại sao lại chọn phép tích đại số và phép lấy *min* cho toán tử T-norm". Phần cuối của chương này là một đề xuất về cách chuyển đổi luật kết hợp mờ về dạng luật kết hợp mờ với thuộc tính số dựa vào ngưỡng w_f tương ứng với các tập mờ f của từng thuộc tính mờ.

- *Chương IV* tập trung vào bài toán "khai phá song song luật kết hợp". Phần đầu của chương này, luận văn tóm tắt lại các thuật toán đã được đề xuất và thử nghiệm thành công. Các thuật toán này giống nhau ở một điểm là phải đồng bộ hóa dù nhiều hay ít trong suốt quá trình tính toán và đây chính là nhược điểm cần khắc phục. Nắm bắt được tính chất của luật kết hợp mờ, luận văn đã đề xuất một thuật toán mới theo đó các bộ xử lý (BXL) trong hệ thống song song hạn chế được tối đa quá trình trao đổi dữ liệu và đồng bộ hóa. Thuật toán khai phá song song luật kết hợp mờ này được xem là gần lý tưởng bởi ngoài việc tránh được nhược điểm truyền thông, nó còn đạt được sự cân bằng tải giữa các BXL nhờ một chiến thuật chia tập thuộc tính ứng cử viên phù hợp.

- *Chương V* tổng kết luận văn bằng việc nêu lại những công việc đã thực hiện và kết quả đạt được của luận văn này. Ngoài ra, chương này cũng đề

cập những vấn đề chưa được giải quyết hoặc giải quyết thấu đáo trong toàn luận văn cũng như công việc và hướng nghiên cứu trong tương lai.

Lời cảm ơn:

Đầu tiên, tôi muốn gửi lời cảm ơn sâu sắc nhất đến cán bộ hướng dẫn khoa học, thầy giáo, TS. Hà Quang Thụy, người đã truyền cho tôi nguồn cảm hứng nghiên cứu khoa học, người đã đưa tôi đến với lĩnh vực nghiên cứu này, và là người đã giảng dạy, hướng dẫn tôi hết sức tận tình trong suốt bốn năm qua.

Tôi xin bày tỏ lời cảm ơn tới các thầy cô giáo đã giảng dạy tôi trong suốt hai năm học qua như GS. Huỳnh Hữu Tuệ, GS, TSKH. Nguyễn Xuân Huy, PGS, TS. Ngô Quốc Tạo, TS. Vũ Đức Thi, TS. Nguyễn Kim Anh, .v.v. Tôi cũng xin trân trọng cảm ơn các nhà khoa học và đồng thời là các thầy giáo trong ban chủ nhiệm lớp cao học $K8T_1$ như GS. VS. Nguyễn Văn Hiệu, GS. TSKH. Bạch Hưng Khang, PGS. TS. Hồ Sỹ Đàm, GS. TSKH. Phạm Trần Nhu, và PGS. TS. Đỗ Đức Giáo.

Tôi cũng muốn gửi lời cảm ơn tới những thành viên trong nhóm seminar về "Khai phá dữ liệu & tính toán song song" như TS. Đỗ Văn Thành, ThS. Phạm Thọ Hoàn, ThS. Đoàn Sơn, CN. Bùi Quang Minh, ThS. Nguyễn Trí Thành, CN. Nguyễn Thành Trung, CN. Tào Thị Thu Phượng, CN. Vũ Bội Hằng, .v.v. Họ là những người thầy, người bạn đã sát cánh bên tôi trong lĩnh vực nghiên cứu này và có những góp ý chuyên môn cũng như sự động viên về tinh thần rất đáng trân trọng.

Tôi xin ghi nhận những tình cảm, sự giúp đỡ về chuyên môn cũng như trong cuộc sống của các thầy giáo, các bạn đồng nghiệp trong Bộ môn Các Hệ thống thông tin, Khoa Cộng nghệ, ĐHQG Hà Nội. Sự quan tâm của những người thầy như TS. Nguyễn Tuệ, PGS. TS. Trịnh Nhật Tiến, ThS. Nguyễn Quang Vinh, ThS. Vũ Bá Duy, ThS. Lê Quang Hiếu .v.v. đã động viên và khích lệ tôi rất nhiều trong thời gian qua.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc tới tất cả người thân trong gia đình tôi, bạn bè tôi. Họ thật sự là nguồn động viên vô tận đối với tôi trong cuộc sống.

Học viên thực hiện luận văn

Mục lục

Danh sách hình vẽ

Danh sách bảng biểu

Bảng từ viết tắt

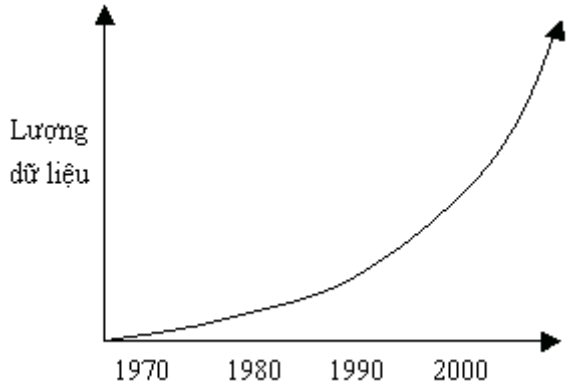
Từ hoặc cụm từ	Từ viết tắt	Từ tiếng Anh
Cơ sở dữ liệu	CSDL	Database
Khai phá dữ liệu	KPDL	Data Mining
BXL	BXL	Processor

Chương I. Tổng quan về *Khai phá dữ liệu*

1.1 Khai phá dữ liệu

1.1.1 Tại sao lại *Khai phá dữ liệu*?

Hơn một thập niên trở lại đây, lượng thông tin được lưu trữ trên các thiết bị điện tử (đĩa cứng, CD-ROM, băng từ, .v.v.) không ngừng tăng lên. Sự tích lũy dữ liệu này xảy ra với một tốc độ bùng nổ. Người ta ước đoán rằng, lượng thông tin trên toàn cầu tăng gấp đôi sau khoảng hai năm và theo đó số lượng cũng như kích cỡ của các CSDL cũng tăng lên một cách nhanh chóng [AR95].



Hình 1 - Lượng dữ liệu được tích lũy tăng mạnh theo thời gian

Chúng ta quả thực đang "ngập" trong dữ liệu, nhưng lại cảm thấy "đói" tri thức và thông tin hữu ích. Lượng dữ liệu khổng lồ này thực sự là một nguồn "tài nguyên" rất giá trị bởi thông tin là yếu tố then chốt trong hoạt động kinh doanh vì nó giúp những người điều hành và quản lý có một cái nhìn sâu sắc, chính xác, khách quan vào tiến trình kinh doanh trước khi ra quyết định. KPDL – khai thác những thông tin tiềm ẩn có tính dự đoán từ những CSDL lớn – là một hướng tiếp cận mới với khả năng giúp các công ty chú trọng vào những thông tin có nhiều ý nghĩa từ những tập hợp dữ liệu lớn (databases, data warehouses, data repositories) mang tính lịch sử. Những công cụ KPDL có thể dự đoán những xu hướng trong tương lai và do đó cho phép doanh nghiệp ra những quyết định kịp thời được định hướng bởi tri thức mà KPDL đem lại. Sự phân tích dữ liệu một cách tự động và mang tính dự báo của KPDL có ưu thế hơn hẳn so với sự phân tích thông thường dựa trên những sự kiện trong quá khứ của các hệ hỗ trợ ra quyết định (decision support systems - DSSs) truyền thống trước đây. Công cụ KPDL cũng có thể trả

lời những câu hỏi trong lĩnh vực kinh doanh mà trước đây được xem là tốn nhiều thời gian để xử lý. Với tất cả những ưu thế trên, KPDL đã chứng tỏ được tính hữu dụng của nó trong môi trường kinh doanh đầy tính cạnh tranh ngày nay. Giờ đây, KPDL đã và đang trở thành một trong những hướng nghiên cứu chính của lĩnh vực khoa học máy tính và công nghệ tri thức.

Phạm vi ứng dụng ban đầu của KPDL chỉ là trong lĩnh vực thương mại (bán lẻ) và tài chính (thị trường chứng khoán). Nhưng ngày nay, KPDL đã được ứng dụng rộng rãi trong các lĩnh vực khác như tin-sinh (bio-informatics), điều trị y học (medical treatment), viễn thông (telecommunication), giáo dục (education), .v.v.

1.1.2 Định nghĩa *Khai phá dữ liệu*

Trước khi nêu một vài định nghĩa về KPDL, tôi xin có giải thích nho nhỏ để độc giả tránh được nhầm lẫn về tên gọi. Với những gì tôi trình bày ở trên, chúng ta có thể hiểu một cách sơ lược rằng KPDL là quá trình tìm kiếm những thông tin (tri thức) hữu ích, tiềm ẩn và mang tính dự báo trong các tập dữ liệu lớn. Như vậy, chúng ta nên gọi quá trình này là *khám phá tri thức* (Knowledge Discovery in Databases – KDD) thay vì là KPDL. Tuy nhiên các nhà khoa học trong lĩnh vực này đồng ý với nhau rằng hai thuật ngữ trên là tương đương và có thể thay thế cho nhau. Họ lý giải rằng, mục đích chính của quá trình khám phá tri thức là thông tin và tri thức có ích, nhưng đối tượng mà chúng ta phải xử lý rất nhiều trong suốt quá trình đó lại chính là dữ liệu.

Mặt khác, khi chia các bước trong quá trình khám phá tri thức, một số nhà nghiên cứu lại cho rằng, KPDL chỉ là một bước trong quá trình *khám phá tri thức* [FSSU96].

Như vậy, khi xét ở mức tổng quan thì hai thuật ngữ này là tương đương nhau, nhưng khi xét cụ thể thì KPDL được xem là một bước trong quá trình *khám phá tri thức*.

Có rất nhiều định nghĩa về KPDL, các định nghĩa này đều là những định nghĩa mang tính mô tả. Tôi xin trích một vài định nghĩa ở nguyên bản tiếng Anh nhằm chuyển tải được y nguyên ý của tác giả và tránh được những sai sót chủ quan:

Định nghĩa 1. William J Frawley, Gregory Piatetsky-Shapiro, và Christopher J Matheus 1991 [FSSU96]:

> *"Knowledge discovery in databases, also known Data mining, is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."*

Định nghĩa 2. Marcel Holshemier và Arno Siebes (1994):

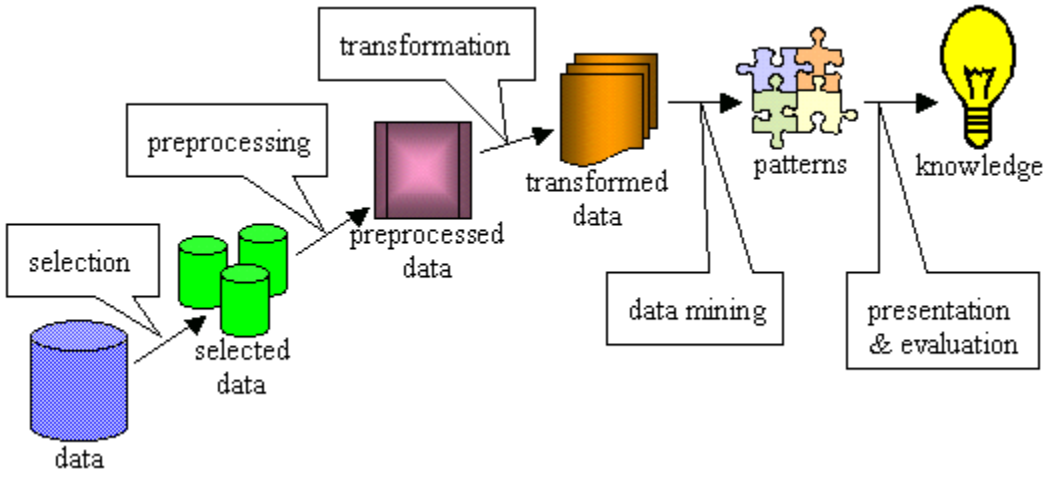
> *"Data Mining is the search for relationships and global patterns that exist in large databases but are 'hidden' among the vast amount of data, such as a relationship between patient data and their medical diagnosis. These relationships represent valuable knowledge about the database and the objects in the database and, if the database is a faithful mirror, of the real world registered by the database."*

1.1.3. Các bước chính trong *Khám phá tri thức* (*KDD*)

Người ta thường chia quá trình *khám phá tri thức* thành các bước sau [AR95] [MM00] [HK02]:

- Trích chọn dữ liệu (data selection): là bước trích chọn những tập dữ liệu cần được khai phá từ các tập dữ liệu lớn (databases, data warehouses, data repositories) ban đầu theo một số tiêu chí nhất định.
- Tiền xử lý dữ liệu (data preprocessing): là bước làm sạch dữ liệu (xử lý với dữ liệu không đầy đủ, dữ liệu nhiễu, dữ liệu không nhất quán, .v.v.), rút gọn dữ liệu (sử dụng hàm nhóm và tính tổng, các phương pháp nén dữ liệu, sử dụng histograms, lấy mẫu, .v.v.), rời rạc hóa dữ liệu (rời rạc hóa dựa vào histograms, dựa vào entropy, dựa vào phân khoảng, .v.v.). Sau bước này, dữ liệu sẽ nhất quán, đầy đủ, được rút gọn, và được rời rạc hóa.
- Biến đổi dữ liệu (data transformation): đây là bước chuẩn hóa và làm mịn dữ liệu để đưa dữ liệu về dạng thuận lợi nhất nhằm phục vụ cho các kỹ thuật khai phá ở bước sau.
- KPDL (data mining): đây là bước áp dụng những kỹ thuật khai phá (phần nhiều là các kỹ thuật của machine learning) để khai phá, trích chọn được những mẫu (patterns) thông tin, những mối liên hệ (relationships) đặc biệt trong dữ liệu. Đây được xem là bước quan trọng và tốn nhiều thời gian nhất của toàn quá trình KDD.

- Biểu diễn và đánh giá tri thức (knowledge representation & evaluation): những mẫu thông tin và mối liên hệ trong dữ liệu đã được khai phá ở bước trên được chuyển dạng và biểu diễn ở một dạng gần gũi với người sử dụng như đồ thị, cây, bảng biểu, luật, .v.v. Đồng thời bước này cũng đánh giá những tri thức khám phá được theo những tiêu chí nhất định.



Hình 2 - Các bước trong quá trình khám phá tri thức (KDD)

1.2 Các hướng tiếp cận và các kỹ thuật áp dụng trong Khai phá dữ liệu

1.2.1 Các hướng tiếp cận và các kỹ thuật chính trong *Khai phá dữ liệu*

Các hướng tiếp cận của KPDL có thể được phân chia theo chức năng hay lớp các bài toán khác nhau. Sau đây là một số hướng tiếp cận chính [HK02].

- Phân lớp và dự đoán (classification & prediction): xếp một đối tượng vào một trong những lớp đã biết trước. Ví dụ: phân lớp vùng địa lý theo dữ liệu thời tiết. Hướng tiếp cận này thường sử dụng một số kỹ thuật của *machine learning* như cây quyết định (decision tree), mạng nơ ron nhân tạo (neural network), .v.v. *Phân lớp* còn được gọi là học có giám sát (học có thầy – supervised learning).
- Luật kết hợp (association rules): là dạng luật biểu diễn tri thứ ở dạng khá đơn giản. Ví dụ: “60 % nam giới vào siêu thị nếu mua bia thì có tới 80%

trong số họ sẽ mua thêm thịt bò khô". Luật kết hợp được ứng dụng nhiều trong lĩnh vực kinh doanh, y học, tin-sinh, tài chính & thị trường chứng khoán, .v.v.

- Khai phá chuỗi theo thời gian (sequential/temporal patterns): tương tự như khai phá luật kết hợp nhưng có thêm tính thứ tự và tính thời gian. Hướng tiếp cận này được ứng dụng nhiều trong lĩnh vực tài chính và thị trường chứng khoán vì nó có tính dự báo cao.
- Phân cụm (clustering/segmentation): xếp các đối tượng theo từng cụm (số lượng cũng như tên của cụm chưa được biết trước. *Phân cụm* còn được gọi là học không giám sát (học không có thầy – unsupervised learning).
- Mô tả khái niệm (concept description & summarization): thiên về mô tả, tổng hợp và tóm tắt khái niệm. Ví dụ: tóm tắt văn bản.

1.2.2 Các dạng dữ liệu có thể khai phá

Do KPDL được ứng dụng rộng rãi nên nó có thể làm việc với rất nhiều kiểu dữ liệu khác nhau [HK02]. Sau đây là một số kiểu dữ liệu điển hình.

- CSDL quan hệ (relational databases)
- CSDL đa chiều (multidimensional structures, data warehouses)
- CSDL dạng giao dịch (transactional databases)
- CSDL quan hệ - hướng đối tượng (object-relational databases)
- Dữ liệu không gian và thời gian (spatial and temporal data)
- Dữ liệu chuỗi thời gian (time-series data)
- CSDL đa phương tiện (multimedia databases) như âm thanh (audio), hình ảnh (image), phim ảnh (video), .v.v.
- Dữ liệu Text và Web (text database & www)

1.3 Ứng dụng của Khai phá dữ liệu

1.3.1 Ứng dụng của *Khai phá dữ liệu*

KPDL tuy là một lĩnh vực mới nhưng thu hút được rất nhiều sự quan tâm của các nhà nghiên cứu nhờ vào những ứng dụng thực tiễn của nó. Chúng ta có thể liệt kê ra đây một số ứng dụng điển hình:

- Phân tích dữ liệu và hỗ trợ ra quyết định (data analysis & decision support)
- Điều trị y học (medical treatment): mối liên hệ giữa triệu chứng, chẩn đoán và phương pháp điều trị (chế độ dinh dưỡng, thuốc men, phẩu thuật, …).
- Text mining & Web mining: phân lớp văn bản và các trang web, tóm tắt văn bản, .v.v.
- Tin-sinh (bio-informatics): tìm kiếm, đối sánh các hệ gene và thông tin di truyền, mối liên hệ giữa một số hệ gene và một số bệnh di truyền, .v.v.
- Tài chính và thị trường chứng khoán (finance & stock market): phân tích tình hình tài chính và dự báo giá của các loại cổ phiếu trong thị trường chứng khoán, .v.v.
- Bảo hiểm (insurance)
- .v.v.

1.3.2 Phân loại các hệ *Khai phá dữ liệu*

KPDL là một công nghệ tri thức liên quan đến nhiều lĩnh vực nghiên cứu khác nhau như CSDL, kỹ thuật máy học (machine learning), giải thuật, trực quan hóa (visualization), .v.v. Chúng ta có thể phân loại các hệ thống KPDL dựa trên các tiêu chí khác nhau.

- Phân loại dựa trên kiểu dữ liệu được khai phá: CSDL quan hệ (relational database), kho dữ liệu (data warehouse), CSDL giao dịch (transactional database), CSDL hướng đối tượng, CSDL không gian (spatial database), CSDL đa phương tiện (multimedia database), CSDL Text và WWW, .v.v.
- Phân loại dựa trên dạng tri thức được khám phá: tóm tắt và mô tả (summarization & description), luật kết hợp (association rules), phân lớp

(classification), phân cụm (clustering), khai phá chuỗi (sequential mining), .v.v.

- Phân loại dựa trên kỹ thuật được áp dụng: hướng CSDL (database-oriented), phân tích trực tuyến (OnLine Analytical Processing – OLAP), machine learning (cây quyết định, mạng nơ ron nhân tạo, k-min, giải thuật di truyền, máy vectơ hỗ trợ - SVM, tập thô, tập mờ, .v.v.), trực quan hóa (visualization), .v.v.
- Phân loại dựa trên lĩnh vực được áp dụng: kinh doanh bán lẻ (retail), truyền thông (telecommunication), tin-sinh (bio-informatics), y học (medical treatment), tài chính & thị trường chứng khoán (finance & stock market), Web mining, .v.v.

1.4 Những vấn đề được chú trọng trong Khai phá dữ liệu

KPDL là một lĩnh vực mới, do đó đang còn rất nhiều vấn đề chưa được nghiên cứu một cách trọn vẹn. Sau đây là một số hướng nghiên cứu đã và đang thu hút được sự chú ý của các nhà tin học.

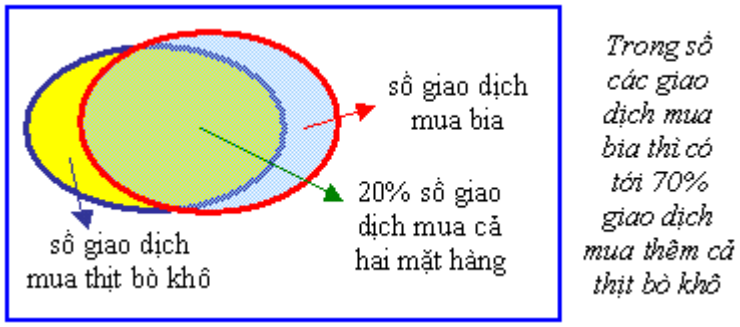
- OLAM (OnLine Analytical Mining) - Sự tích hợp giữa CSDL, kho dữ liệu, và KPDL. Hiện nay một số hệ quản trị CSDL như Oracle, MS SQL Server, DB2 đã tích hợp tính năng xây dựng kho dữ liệu và phân tích trực tuyến (OLAP). Những tính năng này được hỗ trợ dưới dạng những công cụ đi kèm và người dùng phải trả tiền thêm nếu cần sử dụng những tính năng đó. Những nhà nghiên cứu trong lĩnh vực CSDL không muốn dừng lại ở đó mà họ muốn có một sự tích hợp giữa CSDL, kho dữ liệu và KPDL [HK02].
- Khám phá được nhiều dạng tri thức khác nhau từ nhiều kiểu dữ liệu [HK02] [KV01].
- Tính hiệu quả, tính chính xác, độ phức tạp tính toán, khả năng mở rộng và tích hợp, xử lý nhiễu và dữ liệu không đầy đủ, tính hữu dụng (ý nghĩa) của tri thức [HK02].
- Kết hợp KPDL với tri thức cơ sở (background knowledge) [KV01] [PDD99].

- Vấn đề song song hóa và phân tán quá trình KPDL [AS96] [AM95] [MHT02] [HHMT02] [HKK97] [PCY95] [JPO01] [ZHL98] [DP01].
- Ngôn ngữ truy vấn trong KPDL (Data Mining Query Language – DMQL): cung cấp cho người sử dụng một ngôn ngữ hỏi thuật tiện tương tự như SQL đối với CSDL quan hệ [HK02].
- Biểu diễn và trực quan hóa tri thức khai phá được sao cho gần gũi với người sử dụng (human-readable expression). Tri thức có thể biểu diễn đa chiều, đa tầng để người dùng sử dụng tri thức hiệu quả hơn [HK02].

Chương II. Luật kết hợp

2.1 Tại sao lại luật kết hợp?

Luật kết hợp là những luật có dạng "70% khách hàng mua *bia* thì mua thêm *thịt bò khô*, 20% giao dịch có mua cả *bia* lẫn *thịt bò khô*" hoặc "75% bệnh nhân *hút thuốc lá* và *sống ven vùng ô nhiễm* thì *bị ung thư phổi*, trong đó 25% số bệnh nhân vừa *hút thuốc lá*, *sống ven vùng ô nhiễm* vừa *ung thư phổi*" [AIS93]. "*mua bia*" hay "*hút thuốc lá* và *sống ven vùng ô nhiễm*" ở đây được xem là vế trái (tiền đề - antecedent) của luật, còn "*mua thịt bò khô*" hay "*ung thư phổi*" là vế phải (kết luận - consequent) của luật. Những con số 20% hay 25% là độ hỗ trợ của luật (support - số phần trăm các giao dịch chứa cả vế trái lẫn vế phải), còn 70% hay 75% là độ tin cậy của luật (confidence - số phần trăm các giao dịch thỏa mãn vế trái thì cũng thỏa mãn vế phải).



Hình 3 - Minh họa về luật kết hợp

Chúng ta nhận thấy rằng tri thức đem lại bởi những luật kết hợp ở dạng trên có một sự khác biệt cơ bản so với thông tin thu được từ các câu lệnh truy vấn dữ liệu thuông thường (ngôn ngữ SQL chẳng hạn). Đó thường là những tri thức, những mối liên hệ chưa được biết trước và mang tính dự báo đang tiềm ẩn trong dữ liệu. Những tri thức này không đơn giản chỉ là kết quả của các phép nhóm, tính tổng hay sắp xếp mà là kết quả của một quá trình tính toán khá phức tạp và tốn nhiều thời gian.

Tuy luật kết hợp là một dạng luật khá đơn giản nhưng lại mang rất nhiều ý nghĩa. Thông tin mà dạng luật này đem lại là rất đáng kể và hỗ trợ không nhỏ trong quá trình ra quyết định. Tìm kiếm được những luật kết hợp "quý hiếm" và mang nhiều thông tin từ CSDL tác nghiệp là một trong những hướng tiếp cận

chính của lĩnh vực KPDL và đây chính là một động lực không nhỏ thúc đẩy việc tập trung nghiên cứu của nhiều nhà tin học.

2.2 Phát biểu bài toán khai phá luật kết hợp

Cho $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ là tập mục bao gồm *n* mục (item – còn được gọi là thuộc tính - attribute). $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ là tập gồm *m* giao dịch (transaction – còn được gọi là bản ghi - record), mỗi giao dịch được định danh bởi **TID** (Transaction IDentification). Một CSDL **D** là một quan hệ nhị phân δ trên **I** và **T**, hay $\delta \subseteq \mathbf{I}x\mathbf{T}$. Nếu mục **i** xuất hiện trong giao dịch **t** thì ta viết $(\mathbf{i}, \mathbf{t}) \in \delta$ hoặc $\mathbf{i}\delta\mathbf{t}$. Về ý nghĩa, một CSDL là một tập các giao dịch, mỗi giao dịch **t** là một tập mục: $\mathbf{t} \in 2^{\mathbf{I}}$ (với $2^{\mathbf{I}}$ là tập các tập con của **I**) [AIS93] [ZH99].

Sau đây là một ví dụ về CSDL (dạng giao dịch): **I** = {A, C, D, T, W}, **T** = {1, 2, 3, 4, 5, 6} với thông tin về các giao dịch cho ở bảng sau:

Định danh giao dịch (TID)	Tập mục (itemset)
1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

Bảng 1 - Ví dụ về một CSDL dạng giao dịch

$\mathbf{X} \subseteq \mathbf{I}$ được gọi là tập mục (itemset). Độ hỗ trợ (support) của một tập mục **X** được ký hiệu $s(\mathbf{X})$ – là phần trăm số giao dịch trong CSDL chứa **X**. Một tập mục **X** được gọi là tập phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng một ngưỡng *minsup* nào đó được xác định bởi người sử dụng: $s(\mathbf{X}) \geq minsup$ [AIS93].

Bảng sau đây sẽ liệt kê tất cả những tập mục phổ biến (frequent-itemset) trong CSDL cho ở bảng 1 với giá trị *minsup* bằng 50%.

Các tập mục phổ biến	Độ hỗ trợ tương ứng
C	100% (6)
W, CW	83% (5)
A, D, T, AC, AW, CD, CT, ACW	67% (4)
AT, DW, TW, ACT, ATW, CDW, CTW, ACTW	50% (3)

Bảng 2 - Các tập phổ biến trong CSDL ở bảng 1 với độ hỗ trợ tối thiểu là 50%

Luật kết hợp có dạng $X \xrightarrow{c} Y$, trong đó **X** và **Y** là các tập mục thỏa mãn điều kiện $\mathbf{X} \cap \mathbf{Y} = \varnothing$, còn *c* là độ tin cậy (confidence) của luật, $c = s(\mathbf{X} \cup \mathbf{Y}) / s(\mathbf{X})$. Về mặt xác suất, độ tin cậy *c* của một luật là xác suất (có điều kiện) xảy ra **Y** với điều kiện đã xảy ra **X**. Một luật được xem là *tin cậy* nếu độ tin cậy *c* của nó lớn hơn hoặc bằng một ngưỡng *minconf* nào đó do người dùng xác định: $c \geq minconf$ [AIS93].

Bài toán khai phá luật kết hợp (ở dạng đơn giản nhất) đặt ra như sau:

Cho một CSDL **D**, độ hỗ trợ tối thiểu *minsup*, độ tin cậy tối thiểu *minconf*. Hãy tìm kiếm tất cả các luật kết hợp có dạng $X \rightarrow Y$ thỏa mãn độ hỗ trợ $s(\mathbf{X} \cup \mathbf{Y}) \geq minsup$ và độ tin cậy của luật $c(X \rightarrow Y) = s(\mathbf{X} \cup \mathbf{Y}) / s(\mathbf{X}) \geq minconf$.

Hầu hết các thuật toán được đề xuất để khai phá luật kết hợp thường chia bài toán này thành hai pha [AS94] [MTV94] [AM95] [AS96] [ZH99] [AG00]:

- Pha 1: Tìm tất cả các tập mục phổ biến từ CSDL tức là tìm tất cả các tập mục **X** thỏa mãn $s(\mathbf{X}) \geq minsup$. Đây là pha tốn khá nhiều thời gian của CPU (CPU-bound) và thời gian vào ra ổ đĩa (I/O-bound).
- Pha 2: Sinh các luật tin cậy từ các tập phổ biến đã tìm thấy ở pha thứ nhất. Pha này tương đối đơn giản và tốn kém ít thời gian so với pha trên. Nếu **X** là một tập phổ biến thì luật kết hợp được sinh từ **X** có dạng $X' \xrightarrow{c} X \setminus X'$, với **X'** là tập con khác rỗng của **X**, **X** \ **X'** là hiệu của hai tập hợp, và *c* là độ tin cậy của luật thỏa mãn $c \geq minconf$.

Ví dụ, với tập phổ biến ACW có độ tin cậy 67% ở bảng 2 và *minconf* = 70% thì chúng ta có thể sinh các luật kết hợp sau đây:

Luật kết hợp	**Thỏa mãn *minconf* ≥ 70%?**
$A \xrightarrow{100\%} CW$	Có
$C \xrightarrow{67\%} AW$	Không
$W \xrightarrow{80\%} AC$	Có
$AC \xrightarrow{100\%} W$	Có
$AW \xrightarrow{100\%} C$	Có
$C \xrightarrow{80\%} AW$	Có

Bảng 3 - Luật kết hợp sinh từ tập phổ biến ACW

2.3 Những hướng tiếp cận chính trong khai phá luật kết hợp

Kể từ khi được R. Agrawal đề xuất vào năm 1993 [AIS93], lĩnh vực khai phá luật kết hợp đến nay đã được nghiên cứu và phát triển theo nhiều hướng khác nhau. Có những đề xuất nhằm vào cải tiến tốc độ thuật toán, có những đề xuất nhằm tìm kiếm luật có ý nghĩa hơn, v.v. Sau đây là một số hướng chính.

- Luật kết hợp nhị phân (binary association rule hoặc boolean association rule): là hướng nghiên cứu đầu tiên của luật kết hợp. Hầu hết các nghiên cứu ở thời kỳ đầu về luật kết hợp đều liên quan đến luật kết hợp nhị phân [AIS93] [AS94] [MTV94]. Trong dạng luật kết hợp này, các mục (thuộc tính) chỉ được quan tâm là có hay không xuất hiện trong giao dịch của CSDL chứ không quan tâm về "mức độ" xuất hiện. Có nghĩa là việc mua 20 chai bia và 1 chai bia được xem là giống nhau. Thuật toán tiêu biểu nhất khai phá dạng luật này là thuật toán Apriori và các biến thể của nó [AS94]. Đây là dạng luật đơn giản và như sau này ta biết các dạng luật khác cũng có thể chuyển về dạng luật này bằng một số phương pháp như rời rạc hóa, mờ hóa, .v.v. Một ví dụ về dạng luật này: "*Mua bánh mì = 'yes'* AND *mua đường= 'yes' => mua sữa = 'yes'* AND *mua bơ = 'yes'*, với độ hỗ trợ 20% và độ tin cậy 80%"

- Luật kết hợp có thuộc tính số và thuộc tính hạng mục (quantitative and categorical association rule): các thuộc tính của các CSDL thực tế có kiểu rất đa dạng (nhị phân – binary, số - quantitative, hạng mục – categorical, .v.v.). Để phát hiện luật kết hợp với các thuộc tính này, các nhà nghiên cứu đã đề xuất một số phương pháp rời rạc hóa nhằm chuyển dạng luật này về dạng nhị phân để có thể áp dụng các thuật toán đã có [AS96] [MY98]. Một ví dụ về dạng luật này: "*Giới tính = 'Nam'* AND *Tuổi ∈ '50..65'* AND *Cân nặng ∈ '60..80'* AND *Lượng đường trong máu > 120mg/dl => Huyết áp = 'Cao'*, với độ hỗ trợ 30%, độ tin cậy 65%".

- Luật kết hợp mờ (fuzzy association rule): với những hạn chế còn gặp phải trong quá trình rời rạc hóa các thuộc tính số (quantitative attributes), các nhà nghiên cứu đã đề xuất luật kết hợp mờ nhằm khắc phục những hạn chế trên và chuyển luật kết hợp về một dạng tự nhiên hơn, gần gũi hơn với người sử dụng [KFW98] [AG00]. Một ví dụ về dạng luật này: "*Ho khan =*

'yes' AND ***sốt cao*** AND *đau cơ = 'yes'* AND *khó thở = 'yes'* => *Bị nhiễm SARS = 'yes'*, với độ hỗ trợ 4% và độ tin cậy 85%". Trong luật trên, điều kiện ***sốt cao*** ở vế trái của luật là một thuộc tính đã được mờ hóa.

- Luật kết hợp nhiều mức (multi-level association rules): ngoài các dạng luật trên, các nhà nghiên cứu còn đề xuất một hướng nghiên cứu nữa về luật kết hợp là *luật kết hợp nhiều mức* [HF95] [SA95]. Với cách tiếp cận này, người ta sẽ tìm kiếm thêm những luật có dạng "*Mua máy tính PC => Mua hệ điều hành* AND *mua phần mềm tiện ích văn phòng*, …" thay vì chỉ những luật quá cụ thể như "*Mua máy tính IBM PC => Mua hệ điều hành Microsoft Windows* AND *mua Microsoft Office*, …". Rõ ràng, dạng luật đầu là dạng luật tổng quát hóa của dạng luật sau và tổng quát hóa cũng có nhiều mức khác nhau.

- Luật kết hợp với thuộc tính được đánh trọng số (association rule with weighted items): trong thực tế, các thuộc tính trong CSDL không phải có vai trò ngang bằng nhau. Có một số thuộc tính được chú trọng và lúc đó ta nói những thuộc tính đó có mức độ quan trọng cao hơn các thuộc tính khác. Ví dụ, khi khảo sát về khả năng lây nhiễm hội chứng SARS, thông tin về *thân nhiệt*, *đường hô hấp* rõ ràng là quan trọng hơn rất nhiều so với thông tin về *tuổi tác*. Trong quá trình tìm kiếm luật, chúng ta sẽ gán cho các thuộc tính *thân nhiệt*, *đường hô hấp* các trọng số lớn hơn so với trọng số của thuộc tính *tuổi tác*. Đây là một hướng nghiên cứu rất thú vị và đã được một số nhà nghiên cứu đề xuất cách giải quyết bài toán này [LHM99] [WYY01] [THH02]. Với luật kết hợp có thuộc tính được đánh trọng số, chúng ta sẽ khai phá được những luật mang rất nhiều ý nghĩa, thậm chí là những luật "hiếm" (tức có độ hỗ trợ thấp, nhưng mang một ý nghĩa đặc biệt).

- Bên cạnh những nghiên cứu về những biến thể của luật kết hợp, các nhà nghiên cứu còn chú trọng đề xuất những thuật toán nhằm tăng tốc quá trình tìm kiếm tập phổ biến từ CSDL. Người ta chứng minh rằng, chỉ cần tìm kiếm những tập phổ biến tối đại (maximal frequent itemsets) là đủ đại diện cho tập tất cả các tập phổ biến [BCJ01] (thuật toán MAFIA), hoặc chỉ cần tìm tập các tập phổ biến đóng (closed itemsset) là đủ như [PHM01] (thuật toán CLOSET), [ZH99] (thuật toán CHARM), [PBTL99]. Những thuật

toán này cải thiện đáng kể về mặt tốc độ do áp dụng được những chiến lược cắt tỉa "tinh xảo" hơn các thuật toán trước đó.

- Khai phá luật kết hợp song song (parallel mining of association rules): bên cạnh khai phá luật kết hợp với các giải thuật tuần tự, các nhà làm tin học cũng tập trung vào nghiên cứu các giải thuật song song cho quá trình phát hiện luật kết hợp. Nhu cầu song song hóa và xử lý phân tán là cần thiết bởi kích thước dữ liệu ngày càng lớn nên đòi hỏi tốc độ xử lý cũng như dung lượng bộ nhớ của hệ thống phải được đảm bảo. Có rất nhiều thuật toán song song khác nhau đã được đề xuất [AM95] [PCY95] [AS96] [HKK97] [ZHL98] [ZPO01] [DP01], chúng có thể phụ thuộc hoặc độc lập với nền tảng phần cứng.
- Luật kết hợp tiếp cận theo hướng tập thô (mining association rules based on rough set): tìm kiếm luật kết hợp dựa trên lý thuyết tập thô [MS00].
- Ngoài ra, còn một số hướng nghiên cứu khác về khai phá luật kết hợp như: khai phá luật kết hợp trực tuyến [AY98], khai phá luật kết hợp được kết nối trực tuyến đến các kho dữ liệu đa chiều (multidimensional data, data warehouse) thông qua công nghệ OLAP (Online Analysis Processing), MOLAP (Multidimensional OLAP), ROLAP (Relational OLAP), ADO (ActiveX Data Object) for OLAP .v.v.

Chương III. Khai phá luật kết hợp mờ

3.1 Luật kết hợp có thuộc tính số

3.1.1 Luật kết hợp có thuộc tính số

Khai phá luật kết hợp với thuộc tính số và thuộc tính hạng mục (quantitative and categorical association rule) là một trong những hướng tiếp cận quan trọng trong lĩnh vực khai phá luật kết hợp (đã được đề cập ở mục 2.3). Dạng luật này được đề xuất nghiên cứu lần đầu tiên trong [SA96].

Bảng dữ liệu sau đây minh họa một CSDL bao gồm các thuộc tính nhị phân (binary), thuộc tính số (quantitative), và thuộc tính hạng mục (categorical).

Tuổi	Giới tính	Dạng đau ngực (1, 2, 3, 4)	Lượng cholesterol (mg/ml)	Lượng đường trong máu (>120mg/ml)	Điện tâm đồ trạng thái nghỉ (0, 1, 2)	Nhịp tim cực đại	Bị bệnh tim (có, không)
60	1(nữ)	4	206	0(<120mg/ml)	2	132	2(có)
54	1	4	239	0	0	126	2
54	1	4	286	0	2	116	2
52	1	4	255	0	0	161	2
68	1	3	274	1(>120mg/ml)	2	150	2
54	1	3	273	0	2	152	1(không)
54	0(nam)	2	288	1	2	159	1
67	0	3	277	0	0	172	1
46	0	2	204	0	0	172	1
52	1	2	201	0	0	158	1
40	1	4	167	0	2	114	2
37	1	3	250	0	0	187	1
71	0	2	320	0	0	162	1
74	0	2	269	0	2	121	1
29	1	2	204	0	2	202	1
70	1	4	322	0	2	109	2
67	0	3	544	0	2	160	1

Bảng 4 - CSDL khám và chẩn đoán bệnh tim mạch của 17 bệnh nhân

Trong CSDL trên, *Tuổi*, *Lượng cholesterol trong máu*, *Nhịp tim cực đại* là các thuộc tính số (quantitative), *Dạng đau ngực*, *Dạng điện tâm đồ trạng thái nghỉ* là các thuộc tính hạng mục (categorical), còn các thuộc tính còn lại như *Giới tính, Bị bệnh tim, ...* là các thuộc tính nhị phân (binary hay boolean). Thực ra thuộc tính nhị phân cũng là một trường hợp đặc biệt của thuộc tính hạng mục. Với CSDL này, chúng ta có thể rút ra một số luật kết hợp sau:

- **<Tuổi: 54..74>** AND **<Giới tính: Nữ>** AND **<Cholesterol: 200..300>** => **<Bệnh tim: Có>**, với độ hỗ trợ 23.53% và độ tin cậy là 80%.

- **<Giới tính: Nam>** AND **<Điện tâm đồ trạng thái nghỉ: 0>** AND **<Lượng đường trong máu ≤ 120>** => **<Bệnh tim: Không>**, với độ hỗ trợ 17.65% và độ tin cậy là 100%.
- .v.v.

Hướng tiếp cận được đề xuất trong [AS96] nhằm tìm kiếm luật kết hợp dạng nêu trên bằng cách phân khoảng miền giá trị của các thuộc tính số và thuộc tính hạng mục để chuyển tất cả về thuộc tính nhị phân rồi sau đó áp dụng các thuật toán điển hình [AS94] [MTV94] [ZH99] khi phá luật kết hợp nhị phân trước đây.

3.1.2 Các phương pháp rời rạc hóa

Các thuật toán khai phá luật kết hợp nhị phân [AIS93] [AS94] [MTV94] [ZH99] chỉ có thể áp dụng trên những CSDL quan hệ chỉ có thuộc tính nhị phân hoặc CSDL dạng giao dịch như trong bảng 1. Chúng không thể áp dụng trực tiếp với các CSDL có thuộc tính số và thuộc tính hạng mục như trong CSDL ở bảng 4. Muốn thực hiện được điều này, người ta [AS96] [MY98] phải tiến hành rời rạc hóa dữ liệu cho các thuộc tính số để chuyển chúng về thuộc tính nhị phân. Mặc dù các thuật toán được đề xuất trong [SA96] [MY98] có thể giải quyết trọn vẹn bài toán này, tuy vậy kết quả tìm được vẫn chưa làm thỏa mãn những nhà nghiên cứu. Vấn đề không phải ở thuật toán mà là cách thức rời rạc hóa dữ liệu được áp dụng. Mục này sẽ trình bày một vài phương pháp rời rạc hóa, đồng thời đánh giá xem chúng có những nhược điểm gì.

- Nếu A là thuộc tính số rời rạc (quantitative & discrete) hoặc là thuộc tính hạng mục (categorical) với miền giá trị hữu hạn dạng $\{v_1, v_2, \ldots, v_k\}$ và *k* đủ bé (< 100) thì ta sẽ biến đổi thuộc tính này thành *k* thuộc tính nhị phân dạng A_V_1, A_V_2, … A_V_k. Giá trị của một bản ghi tại trường A_V_i bằng True (Yes hoặc 1) nếu giá trị của bản ghi đó tại thuộc tính A ban đầu bằng v_i, trong các trường hợp còn lại giá trị của A_V_i sẽ là False (No hoặc 0). Thuộc tính *Dạng đau ngực* và *Dạng điện tâm đồ trạng thái nghỉ* trong bảng 4 thuộc dạng này. Lúc đó *Dạng đau ngực* sẽ được chuyển thành bốn thuộc tính nhị phân là *Dạng đau ngực_1*, *Dạng đau ngực_2*, *Dạng đau ngực_3*, và *Dạng đau ngực_4*.

Dạng đau ngực (1, 2, 3, 4)	➔	Dạng đau ngực_1	Dạng đau ngực_2	Dạng đau ngực_3	Dạng đau ngực_4
4		0	0	0	1
1	sau khi	1	0	0	0
3	rời rạc	0	0	1	0
2	hóa	0	1	0	0

Bảng 5 - Rời rạc hóa thuộc tính số rời rạc hữu hạn hoặc thuộc tính hạng mục

- Nếu A là thuộc tính số liên tục (quantitative & continuous) hoặc A là thuộc tính số rời rạc hay thuộc tính hạng mục với miền giá trị dạng $\{v_1, v_2, \ldots, v_p\}$ (p lớn) thì ta sẽ ánh xạ thành q thuộc tính nhị phân <A: $start_1..end_1$>, <A: $start_2..end_2$>, …, <A: $start_q..end_q$>. Giá trị của một bản ghi tại trường <A: $start_i..end_i$> sẽ bằng True (Yes hoặc 1) nếu giá trị của bản ghi đó tại thuộc tính A ban đầu năm trong khoảng [$start_i..end_i$], ngược lại nó sẽ nhận giá trị False (No hoặc 0). Thuộc tính *Tuổi*, *Lượng cholesterol*, và *Nhịp tim cực đại* trong CSDL ở bảng 4 là những thuộc tính dạng này. Ví dụ ta chia thuộc tính *Cholesterol* và *Tuổi* thành các thuộc tính nhị phân ở hai bảng sau:

Lượng Cholesterol	➔	<Cholesterol: 150..249>	<Cholesterol: 250..349>	<Cholesterol: 350..449>	<Cholesterol: 450..549>
544		0	0	0	1
206		1	0	0	0
286		0	1	0	0
322		0	1	0	0

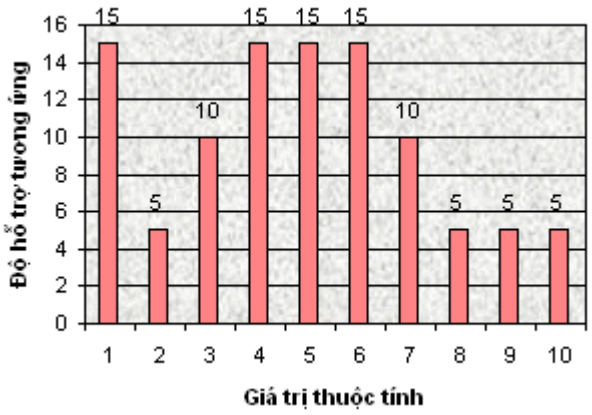
Bảng 6 - Rời rạc hóa thuộc tính số "Lượng cholesterol trong máu"

Tuổi	➔	<Tuổi: 1..29>	<Tuổi: 30..59>	<Tuổi: 60..120>
74		0	0	1
29		1	0	0
30		0	1	0
59		0	1	0
60		0	0	1

Bảng 7 - Rời rạc hóa thuộc tính số "Tuổi tác"

Phương pháp rời rạc hóa trên gặp phải vấn đề "điểm biên gãy" [AG00] [KFW98] (sharp boundary problem). Hình 4 dưới đây cho biết phân bố độ hỗ trợ của một thuộc tính A nào đó có miền giá trị từ 1 đến 10. Nếu chúng ta tiền hành rời rạc hóa thuộc tính A thành 2 khoảng là [1..5] và [6..10] và với độ hỗ trợ cực tiểu là 41% thì khoảng [6..10] sẽ không thỏa mãn độ hỗ trợ tối thiểu (40% < *minsup* = 41%) mặc dù lân cận biên trái của khoảng này có độ hỗ thỏa mãn lớn hơn *minsup*. Ví dụ [4..7] có độ hỗ trợ là 55%, [5..8] có độ hỗ trợ là 45%. Như vậy

phép phân khoảng này tạo nên một "điểm biên gãy" giữa giá trị 5 và 6 và do đó với cách rời rạc này, các thuật toán không thể khai phá ra những luật liên quan đến các giá trị năm trong khoảng [6..10].



Hình 4 - Ví dụ về vấn đề "Điểm biên gãy" khi tiến hành rời rạc hóa dữ liệu

Nhằm khắc phục "điểm biên gãy", [SA96] đã đề xuất một cách phân khoảng mới sao cho các khoảng liền kề có một phần "gối" lên nhau (overlap) ở phần đường biên giữa chúng. Cách phân khoảng này giải quyết được vấn đề trên, nhưng lại gặp phải một vấn đề mới là khi đó tổng độ hỗ trợ của các khoảng lớn hơn 100% và một số giá trị (nằm ở lân cận biên) được "coi trọng" hơn so với các giá trị khác của thuộc tính - điều này là rất thiếu tự nhiên và có phần mâu thuẫn.

Rời rạc hóa theo khoảng cũng nảy sinh một vấn đề về ngữ nghĩa. Ví dụ rời rạc hóa thuộc tính *Tuổi* trong bảng 7 cho thấy rằng 29 và 30 chỉ cách nhau một tuổi lại thuộc về hai khoảng khác nhau. Nếu ta cho khoảng [1..29] là trẻ, [30..59] là trung niên, còn [60..150] là già thì 59 tuổi được xem là trung niên trong khi 60 tuổi lại được xem là già. Đây là điều rất thiếu tự nhiên và không "thuận" với cách tư duy của con người bởi trong thực tế tuổi 60 chỉ "già hơn" tuổi 59 chút ít.

Để khắc phục những vấn đề nảy sinh ở trên, người ta [KFW98] [AG00] đã đề xuất một dạng luật mới: *Luật kết hợp mờ*. Dạng luật này không chỉ khắc phục những điểm yếu của vấn đề phân khoảng mà còn đem lại một dạng luật tự nhiên hơn về mặt ngữ nghĩa, gần gũi hơn với người sử dụng.

Với dạng luật này, những luật kết hợp dạng "**<Tuổi: 54..74>** AND **<Giới tính: Nữ>** AND **<Cholesterol: 200..300>** => **<Bệnh tim: Có>**, với độ hỗ trợ 23.53% và độ tin cậy là 80%" sẽ được biểu diễn lại thành luật kết hợp mờ dạng "<**Tuổi_Già**> AND <**Giới tính: Nữ**> AND <**Cholesterol_Cao**> => <**Bệnh tim: Có**>". Trong đó *Tuổi_Già*

và *Cholesterol_Cao* là hai thuộc tính đã được mờ hóa gắn liền với hai thuộc tính *Tuổi* và *Cholesterol*.

3.2 Luật kết hợp mờ

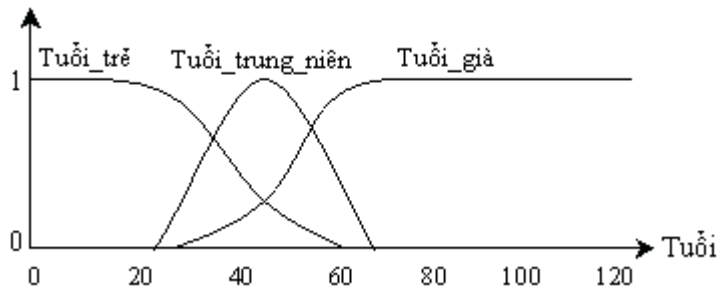
3.2.1 Rời rạc hóa thuộc tính dựa vào tập mờ

Theo lý thuyết tập mờ [LAZ65] [ZHJ91], một phần tử thuộc vào một tập nào đó với một "mức độ thuộc" (membership value) nằm trong khoảng [0, 1]. Giá trị này được xác định dựa vào *hàm thuộc* (membership function) tương ứng với mỗi tập mờ. Ví dụ, cho x là một thuộc tính cùng với miền xác định D_x (còn được gọi là tập vũ trụ), hàm thuộc xác định "mức độ thuộc" của mỗi giá trị x ($\in D_x$) vào tập mờ f_x có dạng sau:

$$m_{f_x}(x) : D_x \rightarrow [0,1] \quad (3.1)$$

Bây giờ chúng ta thử ứng dụng khái niệm tập mờ vào việc rời rạc hóa dữ liệu để giải quyết một số vấn đề còn vướng mắc ở phân trên.

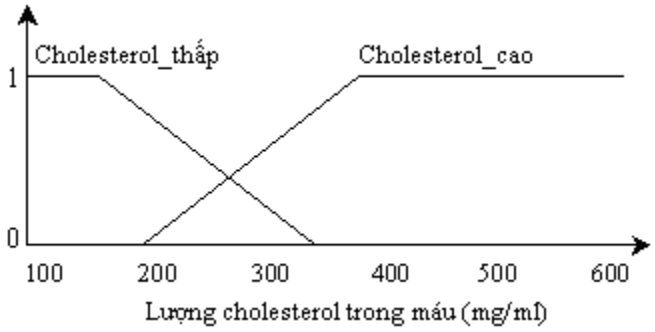
Ví dụ thuộc tính *Tuổi* với tập xác định trong khoảng [0, 120], chúng ta gắn cho nó ba tập mờ tương ứng là *Tuổi_trẻ*, *Tuổi_trung_niên*, và *Tuổi_già* và đồ thị hàm thuộc tương ứng với ba tập mờ này như sau:



Hình 5 - Đồ thị hàm thuộc của các tập mờ "Tuổi_trẻ", "Tuổi_trung_niên", và "Tuổi_già"

Dùng tập mờ để rời rạc hóa dữ liệu, chúng ta đã khắc phục được vấn đề "điểm biên gãy" nhờ tập mờ tạo ra những điểm biên mịn hơn rất nhiều. Ví dụ, trong đồ thị ở hình 5, tuổi 59 và 60 có "mức độ thuộc" vào tập mờ *Tuổi_già* tương ứng là 0.85 và 0.90. Tuổi 30 và 29 có "mức độ thuộc" vào tập mờ *Tuổi_trẻ* lần lượt là 0.70 và 0.75.

Một ví dụ khác về các tập mờ ứng với thuộc tính *Lượng cholesterol* trong máu là *Cholesterol_thấp* và *Cholesterol_cao*.



Hình 6 - Đồ thị hàm thuộc của hai tập mờ "Cholesterol_thấp" và "Cholesterol_cao"

Đối với những thuộc tính hạng mục (categorical) có tập giá trị $\{v_1, v_2, \ldots, v_k\}$ và *k* không quá lớn thì gắn với mỗi giá trị v_i một tập mờ A_V_i (A là tên thuộc tính) có hàm thuộc xác định như sau: $m_{A\_Vi}(x)$ bằng 1 nếu $x = v_i$ và bằng 0 nếu $x \neq v_i$. Thực ra, A_V_i hoàn toàn giống như tập rõ vì giá trị hàm thuộc của nó chỉ là 0 hoặc 1. Trường hợp *k* quá lớn, lúc đó chúng ta có thể chia khoảng và gán tập mờ cho từng khoảng hoặc hỏi ý kiến chuyên gia có hiểu biết về dữ liệu mà chúng ta đang khai phá.

Rời rạc hóa áp dụng tập mờ, chúng ta có một số điểm lợi sau:

- Giải quyết được vấn đề "điểm biên gãy" nhờ tập mờ có thể phân khoảng mịn hơn nhờ vào "độ trơn" của hàm thuộc.
- Rời rạc hóa bằng phân khoảng đôi khi tạo ra số khoảng rất lớn và do đó số thuộc tính nhị phân cũng rất lớn. Còn khi sử dụng tập mờ thì số lượng tập mờ gắn với mỗi thuộc tính là không đáng kể. Ví dụ, áp dụng phân khoảng cho thuộc tính *Lượng cholesterol* chúng ta sẽ thu được 5 khoảng con trong khoảng [100, 600] ban đầu, còn áp dụng tập mờ thì ta chỉ cần hai tập mờ là *Cholesterol_thấp* và *Cholesterol_cao*.
- Ưu điểm thứ ba tập mờ đem lại là nó cho phép chúng ta biểu diễn luật kết hợp dưới dạng tự nhiên hơn, gần gũi với người sử dụng hơn.
- Ưu điểm thứ tư mà tập mờ đem lại là giá trị thuộc tính sau khi rời rạc hóa (sau khi tính qua hàm thuộc) biến thiên trong khoảng [0, 1] cho biết "mức độ thuộc" ít hay nhiều (các thuộc tính nhị phân trước đây chỉ có một trong hai giá trị 0, 1). Điều này cho chúng ta khả năng ước lượng chính xác hơn "độ đóng góp" của các bản ghi trong CSDL vào một tập phổ biến nào đó.

- Ưu điểm thứ năm mà sang phần sau chúng ta sẽ thấy rõ hơn là mặc đù các thuộc tính đã được mờ hóa, nhưng vẫn giữ nguyên được một số tính chất của thuộc tính nhị phân, do đó vẫn có thể áp dụng các thuật toán khai phá luật kết hợp nhị phân vào khai phá luật kết hợp mờ với một chút sửa đổi. Ví dụ tính chất "*mọi tập con khác rỗng của tập phổ biến cũng là tập phổ biến và mọi tập chứa tập không phổ biến đều là tập không phổ biến*" (downward closure property) [AS94] vẫn còn đúng nếu chúng ta chọn được phép toán T-norm (T-chuẩn) phù hợp.
- Một ưu điểm nữa đối với rời rạc hóa dựa vào tập mờ là nó có thể áp dụng tốt cho cả hai dạng CSDL: CSDL quan hệ (relational databases) và CSDL dạng giao dịch (transactional databases).

3.2.2 Luật kết hợp mờ (fuzzy association rules)

Tuổi	Cholesterol (mg/ml)	Đường trong máu (>120mg/ml)	Bị bệnh tim (có, không)
60	206	0 (<120mg/ml)	2 (có)
54	239	0	2
54	286	0	2
52	255	0	2
68	274	1 (>120mg/ml)	2
54	288	1	1 (không)
46	204	0	1
37	250	0	1
71	320	0	1
74	269	0	1
29	204	0	1
70	322	0	2
67	544	0	1

Bảng 8 - CSDL về khám và chẩn đoán bệnh tim mạch của 13 bệnh nhân

Cho $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ là tập n thuộc tính, i_u là thuộc tính thứ u trong $\mathbf{I}$. $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ là tập m bản ghi, t_v là bản ghi thứ v trong $\mathbf{T}$. $t_v[i_u]$ cho biết giá trị của thuộc tính i_u tại bản ghi t_v. Ví dụ, với CSDL trong bảng 8, $t_5[i_2] = t_5[\text{Cholesterol}] = 274$ (mg/ml). Áp dụng phương pháp mờ hóa thuộc tính ở phần trên, chúng ta gắn với một thuộc tính i_u với một tập các tập mờ F_{i_u} như sau:

$$F_{i_u} = \left\{ f_{i_u}^1, f_{i_u}^2, \ldots, f_{i_u}^k \right\} \quad (3.2)$$

Ví dụ, với CSDL trong bảng 8, chúng ta có:

$$F_{i_1} = \mathrm{F_{Tuổi}} = \{Tuổi\_trẻ, Tuổi\_trung\_niên, Tuổi\_già\} \text{ (với } k = 3)$$

$$F_{i_2} = \mathrm{F_{Cholesterol}} = \{Cholesterol\_thấp, Cholesterol\_cao\} \text{ (với } k = 2)$$

Luật kết hợp mờ [AG00] [KFW98] có dạng:

$$\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \quad (3.3)$$

Trong đó:

- $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}$ là các tập mục (itemset). $\mathbf{X} = \{x_1, x_2, \ldots, x_p\}$, $\mathbf{Y} = \{y_1, y_2, \ldots, y_q\}$. $x_i \neq x_j$ (nếu $i \neq j$) và $y_i \neq y_j$ (nếu $i \neq j$).
- $\mathbf{A} = \{f_{x1}, f_{x2}, \ldots, f_{xp}\}$, $\mathbf{B} = \{f_{y1}, f_{y2}, \ldots, f_{yq}\}$ là tập các tập mờ tương ứng với các thuộc tính trong $\mathbf{X}$ và $\mathbf{Y}$. $f_{xi} \in F_{xi}$ và $f_{yj} \in F_{yj}$.

Chúng ta cũng có thể viết lại luật kết hợp mờ ở một trong hai dạng sau:

$$\mathbf{X}=\{x_1, \ldots, x_p\} \text{ is } \mathbf{A}=\{f_{x1}, \ldots, f_{xp}\} \Rightarrow \mathbf{Y}=\{y_1, \ldots, y_q\} \text{ is } \mathbf{B}=\{f_{y1}, \ldots, f_{yq}\} \quad (3.4)$$

Hoặc:

$$(x_1 \text{ is } f_{x1}) \otimes \ldots \otimes (x_p \text{ is } f_{xp}) \Rightarrow (y_1 \text{ is } f_{y1}) \otimes \ldots \otimes (y_q \text{ is } f_{yq}) \quad (3.5)$$

(với $\otimes$ là phép toán T-norm (T-chuẩn) trong logic mờ)

Một tập thuộc tính mờ trong luật kết hợp mờ không chỉ là $\mathbf{X} \subseteq \mathbf{I}$ mà là một cặp <**X**, **A**> với **A** là tập các tập mờ tương ứng với các thuộc tính trong **X**.

Độ hỗ trợ (fuzzy support) của tập mục <**X**, **A**> ký hiệu là *fs*(<**X**, **A**>) được xác định theo công thức:

$$fs(<X, A>) = \frac{\sum_{v=1}^{m} \left\{ \alpha_{x_1}(t_v[x_1]) \otimes \alpha_{x_2}(t_v[x_2]) \otimes \ldots \otimes \alpha_{x_p}(t_v[x_p]) \right\}}{|T|} \quad (3.6)$$

Trong đó:

- $\mathbf{X} = \{x_1, \ldots, x_p\}$, t_v là bản ghi thứ v trong **T**.
- $\otimes$ là toán tử T-norm (T-chuẩn) trong lý thuyết logic mờ. Nó có vai trò như phép toán logic AND trong logic cổ điển.

- $\alpha_{x_u}(t_v[x_u])$ được xác định theo công thức:

$$\alpha_{x_u}(t_v[x_u]) = \begin{cases} m_{x_u}(t_v[x_u]) & neu\ \ m_{x_u}(t_v[x_u]) \geq w_{x_u} \\ 0 & neu\ nguoc\ lai \end{cases} \quad (3.7)$$

Trong đó: m_{x_u} là hàm thuộc của tập mờ f_{x_u} gắn với thuộc tính x_u, còn w_{x_u} là ngưỡng (xác định bởi người dùng) của hàm thuộc m_{x_u}.

- |**T**| (lực lượng của **T**) là số lượng bản ghi trong **T** và chính là bằng *m*.

Tập mục phổ biến: một tập thuộc tính mờ <**X**, **A**> là phổ biến nếu độ độ hỗ trợ của nó lớn hơn hoặc bằng độ hỗ trợ tối thiểu *fminsup* (fuzzy minumum support) do người dùng nhập vào:

$$fs(<\mathbf{X}, \mathbf{A}>) \geq fminsup \quad (3.9)$$

Độ hỗ trợ của một luật mờ được tính theo công thức:

$$fs(<\mathbf{X} \text{ is } \mathbf{A} => \mathbf{B} \text{ is } \mathbf{Y}>) = fs(<\mathbf{X} \cup \mathbf{Y}, \mathbf{A} \cup \mathbf{B}>) \quad (3.10)$$

Một luật được gọi là phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng *fminsup*, có nghĩa là $fs(<\mathbf{X} \text{ is } \mathbf{A} => \mathbf{B} \text{ is } \mathbf{Y}>) \geq fminsup$.

Độ tin cậy (fuzzy confidence) của một luật kết hợp mờ dạng **X** is **A** => **Y** is **B** được ký hiệu là *fc*(**X** is **A** => **Y** is **B**) và xác định theo công thức sau:

$$fc(\mathbf{X} \text{ is } \mathbf{A} => \mathbf{Y} \text{ is } \mathbf{B}) = fs(<\mathbf{X} \text{ is } \mathbf{A} => \mathbf{B} \text{ is } \mathbf{Y}>) / fs(<\mathbf{X}, \mathbf{A}>) \quad (3.11)$$

Một luật được xem là tin cậy nếu độ tin cậy của nó lớn hơn hoặc bằng độ tin cậy tối thiểu *fminconf* (fuzzy minimum confidence) xác định bởi người sử dụng, có nghĩa là: $fc(\mathbf{X} \text{ is } \mathbf{A} => \mathbf{Y} \text{ is } \mathbf{B}) \geq fminconf$.

Toán tử T-norm (⊗): có nhiều cách lựa chọn phép toán T-norm [PDD99] [DMT03] [LAZ65] [ZHJ91] cho công thức (3.6) như:

- Phép lấy min: $a \otimes b = \min(a, b)$
- Tích đại số: $a \otimes b = ab$
- Tích bị chặn: $a \otimes b = \max(0, a + b - 1)$
- Tích Drastic: $a \otimes b = a$ (nếu b=1), = b (nếu a=1), = 0 (nếu a, b < 1)

- Phép giao Yager: $a \otimes b = 1 - \min[1, ((1-a)^w + (1-b)^w)^{1/w}]$ (với $w > 0$). Khi $w = 1$ thì trở thành tích bị chặn, khi w tiến ra $+\infty$ thì trở thành hàm *min*, khi w tiến về 0 thì trở thành tích Drastic.

Qua thực nghiệm, tôi thấy hai phép toán phù hợp nhất là phép lấy *min* và phép *tích đại số* do chúng thuận tiện cho việc tính toán và thể hiện được mối liên hệ chặt chẽ giữa các thuộc tính trong các tập phổ biến. Khi chọn phép lấy min cho toán tử T-norm, công thức (3.6) trở thành công thức (3.12), còn khi chọn phép tích đại số thì (3.6) sẽ trở thành công thức (3.13) như sau:

$$fs(<X,A>) = \frac{\sum_{v=1}^{m} \min\left\{\alpha_{x_1}(t_v[x_1]), \alpha_{x_2}(t_v[x_2]), \ldots, \alpha_{x_p}(t_v[x_p])\right\}}{|T|} \quad (3.12)$$

$$fs(<X,A>) = \frac{\sum_{v=1}^{m} \prod_{x_u \in X} \left\{\alpha_{x_u}(t_v[x_u])\right\}}{|T|} \quad (3.13)$$

Một lý do khác để sử dụng hai phép toán lấy *min* và phép *tích đại số* cho toán tử T-norm lại liên quan đến ngữ nghĩa của luật kết hợp mờ. Trong logic cổ điển, phép kéo theo ($\rightarrow$ hoặc $\Rightarrow$), liên kết hai mệnh đề P và Q để được $P \rightarrow Q$, là một mệnh đề phức hợp, với nội dung ngữ nghĩa là "nếu P thì Q". Đây là một liên kết logic khá phức tạp, nó nhằm diễn tả một quan hệ nhân quả, tức là chỉ trong trường hợp P và Q có quan hệ phụ thuộc nhân quả với nhau. Nhưng khi hình thức hóa, người ta gán cho $P \rightarrow Q$ một giá trị chân lý như là hàm của các giá trị chân lý của P và của Q, nên không tránh khỏi khiên cưỡng về mặt giải thích ngữ nghĩa [PDD99].

Trong logic mờ, phép kéo theo cho ta các mệnh đề phức hợp dạng "nếu u là P thì v là Q", trong đó P là tập mờ trên tập vũ trụ U và Q là tập mờ trên tập vũ trụ V. Ta có thể xem "nếu u là P thì v là Q" tương đương với việc (u, v) thuộc một tập mờ nào đó trên tập vũ trụ UxV, ta ký hiệu tập mờ đó là $P \rightarrow Q$. Định nghĩa quan hệ kéo theo trong logic mờ có nghĩa là định nghĩa tập mờ $P \rightarrow Q$ (hay xác định hàm thuộc $m_{P \rightarrow Q}$) từ các tập mờ P và Q (hay từ hàm thuộc m_P của P và m_Q của Q).

Việc nghiên cứu cấu trúc và ngữ nghĩa của luật kéo theo trong logic mờ đã được nhiều tác giả nghiên cứu, sau đây là một vài cách xác đinh $m_{P\to Q}$ từ m_P và m_Q [PDD99]:

- Theo logic cổ điển: $\forall(u, v) \in U \times V$: $m_{P\to Q}(u, v) = \oplus(1- m_P, m_Q)$. Trong đó, $\oplus$ là toán tử S-norm (hay còn gọi là T-đối chuẩn). Nếu áp dụng $\oplus$ là phép lấy *max* ta có $m_{P\to Q}(u, v) = \max(1- m_P, m_Q)$ (Dienes). Nếu áp dụng $\oplus$ là tổng xác suất thì $m_{P\to Q}(u, v) = 1- m_P + m_P.m_Q$ (Mizumoto). Còn nếu áp dụng $\oplus$ là tổng bị chặn thì $m_{P\to Q}(u, v) = \min(1, 1- m_P + m_Q)$ (Lukaciewicz) .v.v.
- Chúng ta có thể hiểu kéo theo mờ "nếu u là P thì v là Q" chỉ có giá trị chân lý lớn khi cả hai hàm thuộc ở hai vế đều có giá trị lớn, tức là có thể sử dụng toán tử T-norm: $m_{P\to Q}(u, v) = \otimes(m_P, m_Q)$. Nếu áp dụng phép lấy min cho $\otimes$ thì ta có $m_{P\to Q}(u, v) = \min(m_P, m_Q)$ (Mamdani). Nếu áp dụng phép lấy tích đại số thì $m_{P\to Q}(u, v) = m_P . m_Q$ (Mamdani) [DMT03].

Luật kết hợp mờ cũng là một trong những dạng luật kéo theo mờ, do đó nó cũng phải "tuân thủ" về mặt ngữ nghĩa của dạng luật này. Theo cách hiểu của Mamdani thì chúng ta có thể sử dụng toán tử T-norm cụ thể là với phép lấy min và phép tích đại số. Đây chính là một trong những lý do tại sao tôi chọn phép lấy min và phép tích đại số cho toán tử T-norm ở công thức (3.6).

3.2.3 Thuật toán khai phá luật kết hợp mờ

Thuật toán khai phá luật kết hợp mờ được chia làm hai pha như sau:

- Pha 1: Tìm tất cả các tập thuộc tính mờ phổ biến dạng <**X**, **A**> có độ hỗ trợ lớn hơn độ hỗ trợ cực tiểu của người dùng nhập vào: *fs*(<**X**, **A**>) ≥ *fminsup*.
- Pha 2: Sinh các luật kết hợp mờ tin cậy từ các tập phổ biến đã tìm thấy ở pha thứ nhất. Pha này đơn giản và tốn kém ít thời gian hơn so với pha trên. Nếu <**X**, **A**> là một tập thuộc tính mờ phổ biến thì luật kết hợp được sinh ra từ **X** có dạng $X'\ is\ A' \xrightarrow{fc} X \setminus X'\ is\ A \setminus A'$, với **X'** là tập con khác rỗng của **X**, **X** \ **X'** là hiệu của hai tập hợp, **A'** là tập con khác rỗng của **A** và là tập các tập mờ tương ứng với các thuộc tính trong **X'**, **A** \ **A'** là hiệu hai tập hợp, *fc* là độ tin cậy của luật thỏa mãn *fc* ≥ *fminconf* (do người dùng xác định).

Đầu vào của thuật toán (inputs): CSDL **D** với tập thuộc tính **I** và tập bản ghi **T**, độ hỗ trợ tối thiểu *fminsup* và độ tin cậy tối thiểu *fminconf*.

Đầu ra của thuật toán (outputs): tập tất cả các luật kết hợp mờ tin cậy.

Bảng các ký hiệu (notations):

Ký hiệu	Ý nghĩa
D	CSDL (dạng quan hệ hoặc giao dịch)
I	Tập các mục (thuộc tính) trong **D**
T	Tập các giao dịch (hoặc bản ghi) trong **D**
$\mathbf{D_F}$	CSDL mờ (được tính toán từ CSDL ban đầu thông qua hàm thuộc của các tập mờ tương ứng với từng thuộc tính)
$\mathbf{I_F}$	Tập các mục (thuộc tính) trong $\mathbf{D_F}$, mỗi mục hay thuộc tính đều được gắn với một tập mờ. Mỗi tập mờ *f* đều có môt ngưỡng w_f như trong công thức (3.7)
$\mathbf{T_F}$	Tập các giao dịch (hoặc bản ghi) trong $\mathbf{D_F}$, các giá trị thuộc tính trong mỗi giao dịch hoặc bản ghi đã được chuyển sang một giá trị thuộc khoảng [0, 1] nhờ hàm thuộc của các tập mờ tương ứng với từng thuộc tính.
$\mathbf{C_k}$	Tập các tập mục (thuộc tính) có kích thước *k*
$\mathbf{F_k}$	Tập các tập mục (thuộc tính) phổ biến có kích thước *k*
F	Tập tất cả các tập mục (thuộc tính) phổ biến
fminsup	Độ hỗ trợ tối thiểu
fminconf	Độ tin cậy tối thiểu

Bảng 9 - Bảng các ký hiệu sử dụng trong thuật toán khai phá luật kết hợp mờ

Thuật toán:

```
1   BEGIN
2   (D_F, I_F, T_F) = FuzzyMaterialization(D, I, T);
3   F_1 = Counting(D_F, I_F, T_F, fminsup);
4   k = 2;
5   while (F_{k-1} ≠ ∅) {
6         C_k = Join(F_{k-1});
7         C_k = Prune(C_k);
8         F_k = Checking(C_k, D_F, fminsup);
9         F = F ∪ F_k;
10        k = k + 1;
11  }
12  GenerateRules(F, fminconf);
13  END
```

Bảng 10 - Thuật toán khai phá luật kết hợp mờ

Thuật toán trong bảng 10 sử dụng một số chương trình con sau đây:

- Chương trình con ($\mathbf{D_F}$, $\mathbf{I_F}$, $\mathbf{T_F}$) = **FuzzyMaterialization**(**D**, **I**, **T**): hàm này thực hiện nhiệm vụ chuyển đổi từ CSDL **D** ban đầu sang CSDL $\mathbf{D_F}$ với các thuộc tính được gắn thêm các tập mờ và giá trị các thuộc tính ở các bản ghi trong **T** được ánh xạ thành một giá trị thuộc khoảng [0, 1] thông qua hàm thuộc của các tập mờ tương ứng với các thuộc tính.

  Ví dụ, với CSDL **D** trong bảng 8, sau khi thực hiện hàm này, chúng ta sẽ có:

$\mathbf{I_F}$ = {[*Tuổi*, *Tuổi_trẻ*] (1), [*Tuổi*, *Tuổi_trung_niên*] (2), [*Tuổi*, *Tuổi_già*] (3),
[*Cholesterol*, *Cholesterol_thấp*] (4), [*Cholesterol*, *Cholesterol_cao*] (5),
[*Đường_trong_máu*, *Đường_trong_máu_0*] (6),
[*Đường_trong_máu*, *Đường_trong_máu_1*] (7),
[*Bệnh_tim*, *Bệnh_tim_không*] (8), [*Bệnh_tim*, *Bệnh_tim_có*] (9)}

Như vậy $\mathbf{I_F}$ bao gồm 9 thuộc tính đã được mờ hóa so với 4 thuộc tính ban đầu trong CSDL **D**. Mỗi thuộc tính mới là một cặp nằm trong ngoặc vuông bao gồm tên thuộc tính ban đầu và tên của tập mờ gắn với thuộc tính ấy. Ví dụ, thuộc tính *Tuổi* ban đầu sau khi mờ hóa ta sẽ đươc ba thuộc tính mới là [*Tuổi*, *Tuổi_trẻ*] (1), [*Tuổi*, *Tuổi_trung_niên*] (2), [*Tuổi*, *Tuổi_già*] (3). Ngoài ra chương trình con **FuzzyMaterialization** sẽ ánh xạ giá trị các thuộc tính ban đầu sang các giá trị thuộc khoảng [0, 1] nhờ hàm thuộc của các tập mờ. Ví dụ, bảng sau đây được tính toán dựa trên CSDL **D** ở bảng 8:

T	1	2	3	C	4	5	Đ	6	7	B	8	9
60	0.00	0.41	0.92	206	0.60	0.40	0	1	0	2	0	1
54	0.20	0.75	0.83	239	0.56	0.44	0	1	0	2	0	1
54	0.20	0.75	0.83	286	0.52	0.48	0	1	0	2	0	1
52	0.29	0.82	0.78	255	0.54	0.46	0	1	0	2	0	1
68	0.00	0.32	1.00	274	0.53	0.47	1	0	1	2	0	1
54	0.20	0.75	0.83	288	0.51	0.49	1	0	1	1	1	0
46	0.44	0.97	0.67	204	0.62	0.38	0	1	0	1	1	0
37	0.59	0.93	0.31	250	0.54	0.46	0	1	0	1	1	0
71	0.00	0.28	1.00	320	0.43	0.57	0	1	0	1	1	0
74	0.00	0.25	1.00	269	0.53	0.47	0	1	0	1	1	0
29	0.71	0.82	0.25	204	0.62	0.38	0	1	0	1	1	0
70	0.00	0.28	1.00	322	0.43	0.57	0	1	0	2	0	1
67	0.00	0.32	1.00	544	0.00	1.00	0	1	0	1	1	0

Bảng 11 - T_F - giá trị các thuộc tính tại các bản ghi đã được mờ hóa

Chú ý, các chữ cái trong dòng đầu tiên của bảng trên có nghĩa như sau: **T** (Tuổi), **C** (Cholesterol), **Đ** (Đường trong máu), **B** (Bệnh tim).

Do hàm thuộc của mỗi tập mờ *f* có một ngưỡng w_f nên chỉ chỉ những giá trị nào vượt ngưỡng w_f mới được tính đến, ngược lại những giá trị không vượt ngưỡng được xem bằng 0 (theo công thức 3.7). Ngưỡng w_f phụ thuộc vào mỗi hàm thuộc và từng thuộc tính. Những ô được tô màu trong bảng 11 cho biết giá trị của những ô đó vượt ngưỡng (các thuộc tính trong bảng 11 đều lấy w_f bằng 0.5). Những ô không được tô màu được xem có giá trị bằng 0.

- Chương trình con $\mathbf{F_1}$ = **Counting**($\mathbf{D_F}$, $\mathbf{I_F}$, $\mathbf{T_F}$, ***fminsup***): hàm này sinh ra $\mathbf{F_1}$ là tập tất cả các tập phổ biến có lực lượng bằng 1. Các tập thuộc tính phổ biến này phải có độ hỗ trợ lớn hơn hoặc bằng *fminsup*. Ví dụ, áp dụng công thức (3.6) với toán tử T-norm (⊗) là tích đại số và *fminsup* bằng 46% ta được bảng sau:

Tập thuộc tính	**Độ hỗ trợ**	**Là tập phổ biến? *fminsup* = 46%**
{[*Tuổi*, *Tuổi_trẻ*]} (1)	10 %	Không
{[*Tuổi*, *Tuổi_trung_niên*]} (2)	45 %	Không
{[*Tuổi*, *Tuổi_già*]} (3)	76 %	Có
{[*Cholesterol*, *Cholesterol_thấp*]} (4)	43 %	Không
{[*Cholesterol*, *Cholesterol_cao*]} (5)	16 %	Không
{[*Đường_trong_máu*, *Đường_trong_máu_0*]} (6)	85 %	Có
{[*Đường_trong_máu*, *Đường_trong_máu_1*]} (7)	15 %	Không
{[*Bệnh_tim*, *Bệnh_tim_không*]} (8)	54 %	Có
{[*Bệnh_tim*, *Bệnh_tim_có*]} (9)	46 %	Có

Bảng 12 - C_1 - tập tất cả các tập thuộc tính có lực lượng bằng 1

Như vậy $\mathbf{F_1}$ = {{3}, {6}, {8}, {9}}

- Chương trình con $\mathbf{C_k}$ = **Join**($\mathbf{F_{k-1}}$): hàm này thực hiện việc sinh ra tập các tập thuộc tính mờ ứng cử viên có lực lượng *k* từ tập các tập thuộc tính mờ phổ biến lực lượng *k*-1 là $\mathbf{F_{k-1}}$. Cách kết nối sử dụng trong hàm **Join** được thể hiện thông qua ngôn ngữ SQL như sau:

INSERT INTO C_k
SELECT $p.i_1, p.i_2, \ldots, p.i_{k-1}, q.i_{k-1}$
FROM L_{k-1} p, L_{k-1} q
WHERE $p.i_1 = q.i_1, \ldots, p.i_{k-2} = q.i_{k-2}, p.i_{k-1} < q.i_{k-1}$ AND $p.i_{k-1}.o \neq q.i_{k-1}.o$;

Trong đó, $p.i_j$ và $q.i_j$ là số hiệu của thuộc tính mờ thứ *j* trong p và q, còn $p.i_j.o$ và $q.i_j.o$ là số hiệu thuộc tính gốc của thuộc tính mờ thứ *j* trong p và q.

Ví dụ, $\mathbf{C_2}$ = {{3, 6}, {3, 8}, {3, 9}, {6, 8}, {6, 9}}. Tập thuộc tính {8, 9} là không hợp lệ vì cả (8) và (9) có cùng một thuộc tính gốc ban đầu là *Bệnh_tim*.

- Chương trình con $\mathbf{C_k}$ = **Prune**($\mathbf{C_k}$): chương trình con này sử dụng tính chất "*mọi tập con khác rỗng của tập phổ biến cũng là tập phổ biến và mọi tập chứa tập không phổ biến đều là tập không phổ biến*" (downward closure property) để cắt tỉa những tập thuộc tính nào trong $\mathbf{C_k}$ có tập con lực lượng *k*-1 không thuộc tập các tập thuộc tính phổ biến $\mathbf{F_{k-1}}$.

  Sau khi cắt tỉa, $\mathbf{C_2}$ = {{3, 6}, {3, 8}, {3, 9}, {6, 8}, {6, 9}}.

- Chương trình con $\mathbf{F_k}$ = **Checking**($\mathbf{C_k}$, $\mathbf{D_F}$, ***fminsup***): chương trình con này duyệt qua CSDL $\mathbf{D_F}$ để cập nhật độ hỗ trợ cho các tập thuộc tính trong $\mathbf{C_k}$. Sau khi duyệt xong, **Checking** sẽ chỉ chọn những tập phổ biến (có độ hỗ trợ lớn hơn hoặc bằng ***fminsup***) để đưa vào trong $\mathbf{F_k}$.

  Ví dụ, với $\mathbf{C_2}$ ở trên, sau khi thực hiện **Checking**, ta được $\mathbf{F_2}$ = {{3,6}, {6,8}}.

Tập thuộc tính	Độ hỗ trợ	Là tập phổ biến?
{3, 6}	62 %	Có
{3, 8}	35 %	Không
{3, 9}	41 %	Không
{6, 8}	46 %	Có
{6, 9}	38 %	Không

Bảng 13 - F_2 - tập thuộc tính phổ biến có lực lượng bằng 2

- Chương trình còn **GenerateRules**(**F**, ***fminconf***): sinh luật kết hợp mờ tin cậy từ tập các tập phổ biến **F**.

  Với ví dụ trên, sau pha thứ nhất, ta được tập các tập phổ biến $\mathbf{F} = \mathbf{F_1} \cup \mathbf{F_2}$ = {{3}, {6}, {8}, {9}, {3,6}, {6,8}} ($\mathbf{F_3}$ không có vì $\mathbf{C_3}$ bằng tập rỗng). Dưới đây là bảng liệt kê các luật mờ được sinh ra từ **F**:

STT	Luật	Độ hỗ trợ	Độ tin cậy
1	Người già	76 %	
2	Đường trong máu ≤ 120 mg/ml	85 %	
3	Không bị bệnh tim	54 %	
4	Bị bệnh tim	46 %	
5	Người già => Đường trong máu ≤ 120 mg/ml	62 %	82 %
6	Đường trong máu ≤ 120 mg/ml => Người già	62 %	73 %
7	Đường trong máu ≤ 120 mg/ml => Không bị bệnh tim	46 %	54 %
8	Không bị bệnh tim => Đường trong máu ≤ 120 mg/ml	46 %	85 %

Bảng 14 - Các luật mờ được sinh ra từ CSDL trong bảng 8

Với độ tin cậy cực tiểu là 70%, luật thứ 7 ở bảng trên bị loại.

3.2.4 Chuyển luật kết hợp mờ về luật kết hợp với thuộc tính số

Theo công thức 3.7, mỗi hàm thuộc của một tập mờ f đều có một ngưỡng w_f. Những giá trị nào bé hơn ngưỡng w_f thì xem như bằng 0. Nhờ ngưỡng w_f, chúng ta có thể khử mờ để đưa luật kết hợp mờ về dạng gần giống với luật kết hợp với thuộc tính số (quantitative association rules).

Ví dụ, với luật "*Người già => Đường trong máu ≤ 120 mg/ml*, độ hỗ trợ 62%, độ tin cậy 82%" trong bảng 14, chúng ta có thể đưa về dạng sau "*Tuổi ≥ 46 => Đường trong máu ≤ 120 mg/ml*, độ hỗ trợ 62%, độ tin cậy 82%". Chúng ta thấy, giá trị nhỏ nhất còn vượt quá ngưỡng $w_{\text{Tuổi\_già}}$ (= 0.5) trong thuộc tính [*Tuổi*, *Tuổi_già*] là 0.67. Tuổi tương ứng với giá trị mờ bằng 0.67 chính là 46. Trong thuộc tính này, bất cứu người nào có tuổi lớn hơn hoặc bằng 46 thì đều có giá trị hàm mờ lớn hơn hoặc bằng 0.67. "*Tuổi ≥ 46 => Đường trong máu ≤ 120 mg/ml*, độ hỗ trợ 62%, độ tin cậy 82%" hoàn toàn là một luật kết hợp với thuộc tính số.

Vì đa phần hàm thuộc của các tập mờ có đạo hàm ít thay đổi (thường là hàm đơn điệu hoặc số lần đạo hàm đổi dấu là rất ít) nên việc khử mờ tương đối đơn giản.

3.2.5 Thử nghiệm và kết luận

- Thử nghiệm với kích thước dữ liệu (số bản ghi tăng dần) và thời gian tìm kiếm luật
- Thử nghiệm kết quả bằng cách biến thiên độ hỗ trợ và độ tin cậy
- Thử nghiệm số luật tìm được khi biến thiên các trọng số hàm thuộc của các tập mờ
- Thử nghiệm với các toán tử T-norm khác nhau (phép lấy min và tích đại số)
- Thử nghiệm chuyển từ luật kết hợp mờ sang luật kết hợp với thuộc tính được đánh trọng số

Chương IV. Khai phá song song luật kết hợp mờ

Một trong những bước quan trọng của khai phá luật kết hợp là tìm tất cả các tập thuộc tính phổ biến trong CSDL. Đây là bước tương đối phức tạp và tốn nhiều thời gian của CPU (CPU-bound) lẫn thời gian vào ra (I/O-bound) nên các nhà làm tin học đã bỏ nhiều công sức để cải tiến những thuật toán cũ hoặc tìm ra các thuật toán mới nhằm tăng tốc độ tìm kiếm [AS94] [MTV94] [BCJ01] [PHM01] [ZH99] [PBTL99]. Những thuật toán này đều ở dạng tuần tự (sequential algorithms) và làm việc tương đối tốt với những CSDL có kích cỡ không quá lớn (tiêu chí đánh giá CSDL lớn hay nhỏ phụ thuộc vào số thuộc tính và số bản ghi). Tuy nhiên, những thuật toán này sẽ giảm tính hiệu quả một cách đáng kể khi gặp phải những CSDL lớn (hàng trăm megabyte trở lên) do hạn chế về dung lượng bộ nhớ trong và tốc độ tính toán của một máy tính đơn lẻ.

Với sự phát triển bùng nổ của công nghệ phần cứng, theo đó các hệ máy tính song song có sức mạnh tính toán vượt trội ra đời đã mở ra một hướng tiếp cận mới trong KPDL, đó là KPDL song song. Từ năm 1995 trở lại đây, các nhà nghiên cứu đã không ngừng đề xuất các thuật toán song song và phân tán cho bài toán phát hiện luật kết hợp [AM95] [PCY95] [AS96] [HKK97] [ZHL98] [ZPO01] [DP01]. Những thuật toán song song khá đa dạng do một phần chúng được thiết kế phụ thuộc vào kiến trúc của từng hệ máy tính song song cụ thể.

Trong phần đầu tiên của chương này tôi muốn trình bày sơ lược một số thuật toán song song đã đuợc đề xuất và thử nghiệm. Phần tiếp theo tôi xin đề xuất một thuật toán song song cho bài toán khai phá luật kết hợp mờ chạy trên hệ thống PC-Cluster với cơ chế truyền thông điệp của MPI (Message Passing Interface) [MPIS95] [EMPI97] [JDMPI97]. Đây là một thuật toán khá lý tưởng bởi nó hạn chế tối đa được quá trình đồng bộ hóa và trao đổi dữ liệu trong trong tiến trình song song hóa. Tuy nhiên, hạn chế của thuật toán này là chỉ làm việc được với luật kết hợp mờ và luật kết hợp với thuộc tính số và do đó nó phù hợp với CSDL dạng quan hệ hơn là dạng giao dịch.

4.1 Một số thuật toán song song khai phá luật kết hợp

Trong phần này, tôi xin trình bày một số thuật toán song song đã được đề xuất và thử nghiệm. Các thuật toán này được thiết kế trên hệ máy tính song song không chia sẻ (shared-nothing architecture) có tính chất như sau:

- Hệ có **N** bộ xử lý (BXL - processor), mỗi BXL $\mathbf{P^i}$ này có bô nhớ trong (RAM) và bộ nhớ ngoài (thường là ổ đĩa) độc lập với các BXL còn lại trong hệ thống.
- **N** BXL này có thể truyền thông với nhau nhờ một mạng tốc độ cao sử dụng cơ chế truyền thông điệp (message passing).

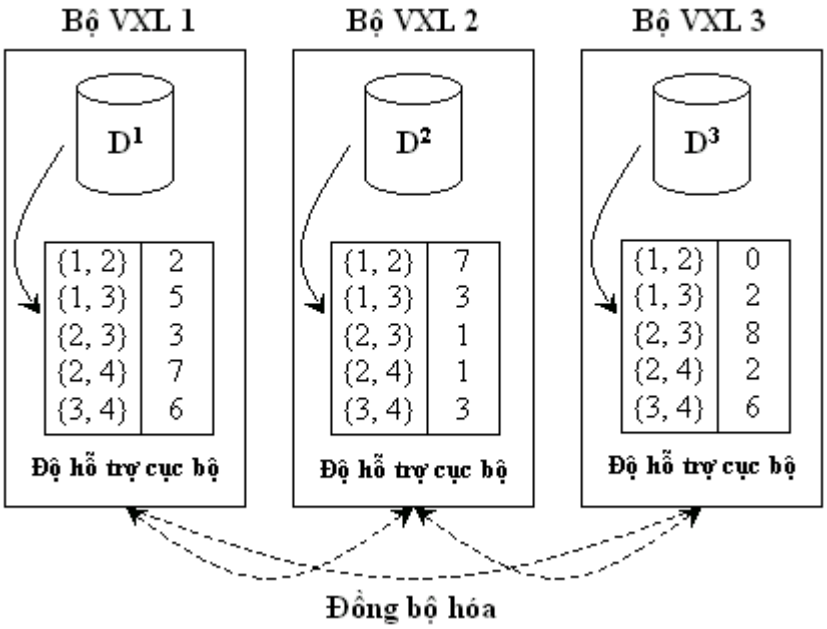
4.1.1 Thuật toán phân phối độ hỗ trợ

Thuật toán song song phân phối độ hỗ trợ (count distribution) dựa trên nền thuật toán Apriori [AS94]. Trong thuật toán này, **N** là số BXL, $\mathbf{P^i}$ là BXL thứ *i*, $\mathbf{D^i}$ là phần dữ liệu được gắn với BXL $\mathbf{P^i}$ (CSDL **D** ban đầu được chia ra làm **N** phần, mỗi phần gắn với một BXL). Thuật toán bao gồm các bước sau:

- (1) Bước 1, với $k = 1$, tất cả **N** BXL đều nhận được $\mathbf{L_k}$ là tập tất cả các tập thuộc tính phổ biến có lực lượng bằng 1.
- (2) Bước 2, với mọi $k > 1$, thuật toán thực hiện lặp đi lặp lại các bước sau:
  - (2.1) Mỗi BXL $\mathbf{P^i}$ tạo ra tập các tập thuộc tính ứng cử viên $\mathbf{C_k}$ bằng cách kết nối các tập thuộc tính phổ biến trong $\mathbf{L_{k-1}}$. Nhớ rằng, tất cả các BXL đều có thông tin về $\mathbf{L_{k-1}}$ giống hệt nhau nên chúng sinh ra $\mathbf{C_k}$ cũng giống hệt nhau.
  - (2.2) Mỗi BXL $\mathbf{P^i}$ duyệt qua CSDL $\mathbf{D^i}$ của riêng nó để cập nhật độ hỗ trợ cục bộ cho các tập thuộc tính ứng cử viên trong $\mathbf{C_k}$. Đây chính là quá trình các BXL thực hiện song song với nhau.
  - (2.3) Sau khi đã cập nhật xong độ hỗ trợ cục bộ cho các tập thuộc tính ứng cử viên trong $\mathbf{C_k}$, các BXL tiến hành truyền thông cho nhau để thu được độ hỗ trợ toàn cục. Ở bước này, các BXL bắt buộc phải đồng bộ hóa với nhau.
  - (2.4) Các BXL căn cứ vào độ hỗ trợ tối thiểu *minsup* để chọn ra tập những tập thuộc tính phổ biến $\mathbf{L_k}$ từ tập các ứng cử viên $\mathbf{C_k}$.

- (2.5) Mỗi BXL có quyền kết thúc tại bước này hoặc tiếp tục thực hiện lặp lại bước 2.1.

Hình sau đây minh họa nguyên lý làm việc của thuật toán này.



Hình 7 - Thuật toán phân phối độ hỗ trợ trên hệ 3 BXL

4.1.2 Thuật toán phân phối dữ liệu

Ưu điểm nổi bật của thuật toán phân phối độ hỗ trợ là không cần truyền dữ liệu giữa các BXL trong quá trình tính toán. Do đó, chúng có thể hoạt động độc lập và không đồng bộ với nhau trong khi duyệt dữ liệu trên bộ nhớ hoặc ổ đĩa cục bộ. Tuy nhiên, nhược điểm của thuật toán này là không khai thác hết sức mạnh tổng hợp của **N** bộ nhớ ứng với **N** BXL của toàn hệ thống. Giả sử mỗi BXL có dung lượng bộ nhớ cục bộ là |**M**| thì số tập thuộc tính ứng cử viên được cập nhật độ hỗ trợ trong mỗi pha bị giới hạn bởi hằng số *m* phụ thuộc |**M**|. Khi số BXL trong hệ thông tăng từ 1 đến **N**, hệ thống sẽ có một bộ nhớ tổng hợp với dung lượng **N** x |**M**|, nhưng với thuật toán phân phối độ hỗ trợ ở trên, chúng ta cũng chỉ đếm được *m* tập thuộc tính ứng cử viên do tính chất của thuật toán là tất cả các BXL đều có tập $\mathbf{C}_k$ giống hệt nhau.

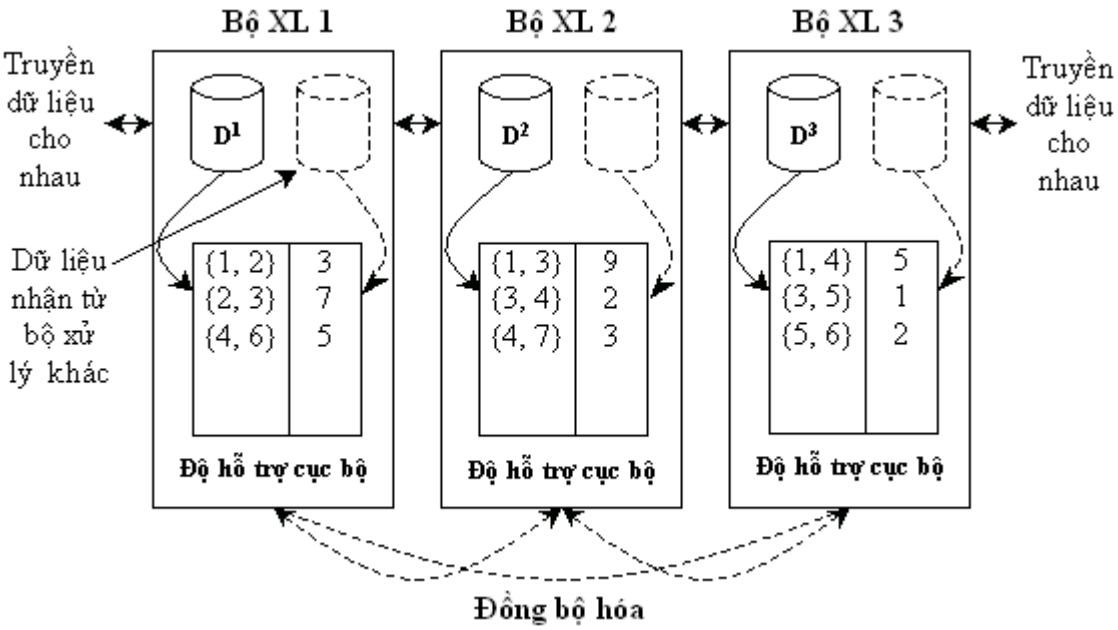
Thuật toán phân phối dữ liệu (data distribution) được thiết kế với mục đích tận dụng được sức mạnh tổng hợp của bộ nhớ hệ thống khi số BXL tăng lên. Trong thuật toán này, mỗi BXL tiến hành cập nhật độ hỗ trợ cho một số các tập thuộc

tính ứng cử viên của riêng nó. Do đó, khi số BXL trong hệ thống tăng lên, thuật toán này có thể cập nhật độ hỗ trợ cho rất nhiều các tập thuộc tính ứng cử viên trong một pha. Nhược điểm của thuật toán này là mỗi BXL phải truyền và nhận dữ liệu ở mỗi pha nên nó chỉ khả thi khi hệ thống có một môi trường truyền thông nhanh và ổn định giữa các nút trong hệ thống.

Thuật toán song song phân phối dữ liệu (data distribution) cũng dựa trên nền thuật toán Apriori [AS94]. Trong thuật toán này, **N** là số BXL, $\mathbf{P^i}$ là BXL thứ *i*, $\mathbf{D^i}$ là phần dữ liệu được gắn với BXL $\mathbf{P^i}$ (CSDL **D** ban đầu được chia ra làm **N** phần, mỗi phần gắn với một BXL). Thuật toán bao gồm các bước sau:

- (1) Bước 1: tương tự như trong thuật toán phân phối độ hỗ trợ
- (2) Bước 2: với $k > 1$:
  - (2.1) Mỗi BXL $\mathbf{P^i}$ tạo tập các tập thuộc tính ứng cử viên $\mathbf{C_k}$ từ tập các tập thuộc tính phổ biến $\mathbf{L_{k-1}}$. Nó không thao tác tất cả trên $\mathbf{C_k}$ mà chỉ giữ lại một phần của $\mathbf{C_k}$ được chia đều cho **N** BXL. Phần được giữ lại cho BXL $\mathbf{P^i}$ được xác định nhờ định danh tiến trình (process identification) mà không cần truyền thông giữ các tiến trình. Các $\mathbf{C_k^i}$ được chia thỏa mãn: $\mathbf{C_k^i} \cap \mathbf{C_k^j} = \varnothing$ (với mọi $i \neq j$) và $\bigcup_{i=1}^{N} C_k^i = C_k$.
  - (2.2) BXL $\mathbf{P^i}$ chỉ đếm độ hỗ trợ cho các tập mục ứng cử viên trong $\mathbf{C_k^i}$ bằng cách sử dụng dữ liệu cục bộ $\mathbf{D^i}$ của nó và dữ liệu nhận được từ các BXL khác trong hệ thống.
  - (2.3) Sau khi đếm xong độ hỗ trợ, mỗi BXL $\mathbf{P^i}$ chọn ra tập những tập thuộc tính phổ biến cục bộ $\mathbf{L_k^i}$ từ $\mathbf{C_k^i}$ tương ứng. Nhớ rằng $\mathbf{L_k^i} \cap \mathbf{L_k^j} = \varnothing$ (với mọi $i \neq j$) và $\bigcup_{i=1}^{N} L_k^i = L_k$.
  - (2.4) Các BXL tiến hành trao đổi $\mathbf{L_k^i}$ cho nhau sao cho tất cả các BXL đều nhận được $\mathbf{L_k}$ để sinh $\mathbf{C_{k+1}}$ cho lần lặp tiếp theo. Bước này cần sự đồng bộ hóa giữa các BXL. Sau khi nhận được bước $\mathbf{L_k}$, mỗi BXL có thể độc lập quyết định ngừng làm việc hoặc tiếp tục thực hiện bước lặp tiếp theo.

Hình sau đây minh họa nguyên lý làm việc của thuật toán này.



Hình 8 - Thuật toán phân phối dữ liệu trên 3 BXL

4.1.3 Thuật toán phân phối tập ứng cử viên

Hạn chế của hai thuật toán trên (count & data distribution) ở chỗ do mọi giao dịch hoặc bản ghi trong CSDL đều có thể hỗ trợ một tập thuộc tính ứng cử viên nào đó nên các giao dịch hay bản ghi phải được đối sánh với tất cả các tập thuộc tính ứng cử viên. Điều này dẫn đến việc thuật toán phân phối độ hỗ trợ phải lưu giữ tập các tập ứng cử viên giống nhau trên mọi BXL và thuật toán phân phối dữ liệu phải gửi dữ liệu cho nhau trong quá trình cập nhật độ hỗ trợ. Hơn nữa, hai thuật toán này phải tiến hành đồng bộ hóa ở cuối mỗi pha thực hiện song song để trao đổi độ hỗ trợ cục bộ hoặc tập các tập phổ biến cho nhau. Yêu cầu đồng bộ hóa trong suốt thời gian thực hiện của thuật toán sẽ làm giảm hiệu suất thực hiện của hệ thống do các BXL hoàn thành công việc sớm phải "chờ đợi" các BXL hoàn thành công việc muộn hơn. Nguyên nhân của vấn đề này là do hai thuật toán trên mới chia công việc cho các BXL một cách "công bằng" chứ chưa chia một cách vừa "công bằng" vừa "khôn ngoan".

Thuật toán phân phối tập ứng cử viên (candidate distribution) cố gắng chia tập ứng cử viên sao cho các BXL có thể độc lập làm việc và hạn chế tối đa công việc đồng bộ hóa. Bắt đầu một pha *l* nào đó (*l* được xác định dựa theo kinh nghiệm),

thuật toán này chia tập thuộc tính phổ biến $\mathbf{L_{l-1}}$ cho các BXL sao cho mỗi BXL $\mathbf{P^i}$ có thể tạo ra tập ứng cử viên $\mathbf{C_m^i}$ ($m \geq l$) độc lập với các BXL khác ($\mathbf{C_m^i} \cap \mathbf{C_m^j} = \varnothing, \ \forall i \neq j$). Đồng thời, dữ liệu cũng được chia lại sao cho mỗi BXL $\mathbf{P^i}$ có thể cập nhật độ hỗ trợ cho các tập ứng cử viên trong $\mathbf{C_m^i}$ độc lập với các BXL khác. Đúng thời gian đó, dữ liệu được phân chia lại sao cho mỗi BXL $\mathbf{P^i}$ có thể cập nhật độ hỗ trợ cho các tập thuộc tính ứng cử viên trong $\mathbf{C_m^i}$ một cách độc lập với các BXL khác. Nhớ rằng, sự phân chia dữ liệu phụ thuộc rất nhiều vào bước phân chia tập ứng cử viên trước đó. Nếu phân chia tập ứng cử viên không "khéo léo" thì chúng ta không thể có một phân hoạch dữ liệu cho các BXL mà chỉ có một phân chia tương đối – nghĩa là có thể có những phần dữ liệu trùng lặp trên các BXL.

Sau khi phân hoạch $\mathbf{L_{k-1}}$, các BXL làm việc độc lập với nhau. Việc cập nhật độ hỗ trợ cho tập các ứng cử viên cục bộ không đòi hỏi các BXL phải truyền thông với nhau. Chỉ có một sự phụ thuộc duy nhất giữa các BXL là chúng phải gửi cho nhau những thông tin cần cho việc cắt tỉa các ứng cử viên không cần thiết. Tuy nhiên, những thông tin này có thể được truyền cho nhau theo chế độ dị bộ và các BXL không cần phải đợi để nhận đầy đủ thông tin này từ các BXL khác. Các BXL cố gắng cắt tỉa được càng nhiều càng tốt nhờ vào những thông tin đến từ các BXL khác. Những thông tin đến muộn sẽ được sử dụng cho lần cắt tỉa tiếp theo. Thuật toán phân phối tập ứng cử viên bao gồm những bước sau:

- (1) Bước 1 ($k < l$): sử dụng một trong hai thuật toán phân phối độ hỗ trợ hoặc phân phối dữ liệu.
- (2) Bước 2 ($k = l$):
  - (2.1) Phân chia $\mathbf{L_{k-1}}$ cho **N** BXL. Chúng ta sẽ xem xét cách phân chia ở phần sau. Quá trình phân chia này là giống hệt nhau và được thực hiện song song trên các BXL.
  - (2.2) Mỗi BXL $\mathbf{P^i}$ sẽ sử dụng $\mathbf{L^i_{k-1}}$ để tạo ra $\mathbf{C_k^i}$ của nó.
  - (2.3) $\mathbf{P^i}$ sẽ cập nhật độ hỗ trợ cho các tập ứng cử viên trong $\mathbf{C_k^i}$ và CSDL sẽ được phân chia lại ngay sau đó.
  - (2.4) Sau đó, $\mathbf{P^i}$ thực hiện trên dữ liệu cục bộ và tất cả dữ liệu nhận được từ các BXL khác. Nó tạo ra **N-1** bộ đệm nhận dị bộ để nhận các

$\mathbf{L_k^j}$ từ các BXL khác. Những $\mathbf{L_k^j}$ này cần thiết cho bước cắt tỉa các tập ứng cử viên trong $\mathbf{C^i_{k+1}}$.

- (2.5) $\mathbf{P^i}$ sinh ra $\mathbf{L_k^i}$ từ $\mathbf{C_k^i}$ và truyền thông lan truyền (broadcast) dị bộ tới **N-1** bộ vi xử lý khác.

- (3 Bước 3 ($k > l$):
  - (3.1) Mỗi BXL $\mathbf{P^i}$ thu thập tất cả những tập phổ biến từ các BXL khác. Thông tin về các tập phổ biến này sẽ được dùng để cắt tỉa. Các tập thuộc tính nhận được từ BXL j sẽ có độ dài k-1, nhỏ hơn k-1 (nếu là BXL chậm), hoặc lớn hơn k-1 (nếu là BXL nhanh).
  - (3.2) $\mathbf{P^i}$ tạo ra $\mathbf{C_k^i}$ dựa vào $\mathbf{L^i_{k-1}}$. Một trường hợp có thể xảy ra là $\mathbf{P^i}$ không nhận được $\mathbf{L^j_{k-1}}$ từ các BXL khác, do đó $\mathbf{P^i}$ cần phải "cẩn thận" trong khoảng thời gian cắt tỉa.
  - (3.3) $\mathbf{P^i}$ thực hiện duyệt dữ liệu để cập nhật độ hỗ trợ cho các tập thuộc tính trong $\mathbf{C_k^i}$. Sau đó nó tính toán $\mathbf{L_k^i}$ từ $\mathbf{C_k^i}$ và truyền dị bộ thông tin về $\mathbf{L_k^i}$ tới **N-1** BXL còn lại trong hệ thống.

Chiến lược phân chia dữ liệu: Chúng ta xem xét cách phân chia dữ liệu của thuật toán này thông qua một ví dụ đơn giản sau đây. Cho $\mathbf{L_3}$ = {ABC, ABD, ABE, ACD, ACE, BCD, BCE, BDE, CDE}. Tiếp đó $\mathbf{L_4}$ = {ABCD, ABCE, ABDE, ACDE, BCDE}, $\mathbf{L_5}$ = {ABCDE} và $\mathbf{L_6} = \varnothing$. Chúng ta xét tập ε = {ABC, ABD, ABE} với các thành viên của nó có chung phân đầu là AB. Nhớ rằng, các tập thuộc tính ABCD, ABCE, ABDE và ABCDE cũng có chung tiền tố AB.

Do đó, giả sử rằng các thuộc tính trong tập thuộc tính được sắp theo thứ tự từ vựng, chúng ta có thể phân chia các tập phổ biến trong $\mathbf{L_k}$ dựa vào tiền tố có độ dài k-1 đầu tiên của các tập, nhờ vậy các BXL có thể làm việc độc lập với nhau.

Cài đặt thuật toán này trong thực tế phức tạp hơn rất nhiều bởi hai lý do. Lý do thứ nhất là một BXL có thể phải nhận các tập thuộc tính phổ biến được tính toán bởi các BXL khác cho bước cắt tỉa tiếp theo. Trong ví dụ trên, BXL được gán tập ứng cử viên ε phải biết BCDE có phải là tập phổ biến hay không mới quyết định được có cắt tỉa tập ABCDE hay không, nhưng tiền tố của BCDE là BC nên BCDE lại thuộc về một BXL khác. Lý do thứ hai là chúng ta phải tính toán cân bằng tải cho các BXL trong hệ thống.

4.1.3 Thuật toán sinh luật song song

Cho một tập phổ biến *l*, chương trình con sinh luật kết hợp sẽ sinh ra luật dạng $a \Rightarrow (l - a)$, trong đó *a* là một tập con khác rỗng của *l*. Độ hỗ trợ của luật chính là độ hỗ trợ của tập phổ biến *l* (tức là $s(l)$), còn độ tin cậy của luật là tỷ số $s(l)/s(a)$. Để sinh luật hiệu quả, chúng ta tiến hành duyệt các tập con của *l* có kích thước lớn trước tiên và sẽ tiếp tục xét các tập con nhỏ hơn khi luật vừa sinh thỏa mãn độ tin cậy tối thiểu (*minconf*). Ví dụ, *l* là tập phổ biến ABCD, nếu luật ABC => D không thỏa mãn độ tin cậy tối thiểu thì luật AB => CD cũng không thỏa mãn do độ hỗ trợ của AB luôn lớn hơn hoặc bằng ABC. Như vậy chúng ta không cần xét các luật mà vế trái là tập con của ABC vì chúng không thỏa mãn độ tin cậy tối thiểu.

Thuật toán sinh luật tuần tự [AS94] thể hiện ý tưởng trên như sau:

```
// Simple algorithm
Forall frequent itemset l_k, k > 1 do
      Call gen_rules(l_k, l_k);

// The gen_rules generates all valid rules α => (l - α), for all α ⊂ a_m
Procedure gen_rules(l_k : frequent k-itemset, a_m : frequent m-itemset)
Begin
1     A = {(m-1)-itemsets a_m-1 | a_m-1 ⊂ a_m}
2     Forall a_m-1 ∈ A do begin
3           conf = s(l_k)/s(a_m-1);
4           if (conf ≥ minconf) then begin
5                 output the rule a_m-1 => (l_k – a_m-1);
6                 if (m – 1 > 1) then
7                       Call gen_rules(l_k, a_m-1);
8           end
9     end
End
```

Bảng 15 - Thuật toán sinh luật kết hợp tuần tự

Để sinh luật song song, chúng ta chia tập các tập thuộc tính phổ biến cho tất cả các BXL trong hệ thống. Mỗi BXL sinh luật trên các tập phổ biến được phân chia cho nó sử dụng thuật toán trên.

Trong thuật toán sinh luật song song, để tính độ tin cậy của một luật, BXL có thể cần phải tham chiếu đến độ hỗ trợ của một tập phổ biến nằm trên một BXL khác. Vì lý do này, các BXL nên có thông tin về toàn bộ các tập phổ biến trước khi thực hiện thuật toán sinh luật song song.

4.1.4 Một số thuật toán khác

Ngoài ba thuật toán nêu trên, các nhà nghiên cứu trong lĩnh vực này đã đề xuất thêm khá nhiều thuật toán khai phá luật kết hợp song song khác.

Thuật toán phân phối dữ liệu thông minh (Intelligent Data Distribution Algorithm) [HKK97] được đề xuất dựa trên thuật toán phân phối dữ liệu với một bước cải tiến trong việc truyền dữ liệu giữa các BXL trong thời gian tính toán. Thay vì truyền dữ liệu giữa cặp BXL, các BXL trong thuật toán này được tổ chức thành một vòng logic và chúng tiến hành truyền dữ liệu theo vòng tròn này.

Thuật toán MLFPT (Multiple Local Frequent Pattern Tree) [ZHL98] là thuật toán dựa trên FP-growth. Thuật toán này giảm được số lần duyệt qua CSDL, không cần tạo ra tập ứng cử viên và cân bằng tải giữa các BXL trong hệ thống.

Thuật toán khai phá luật kết hợp song song do [ZPO01] đề xuất khác với các thuật toán khác ở chỗ nó làm việc trên hệ thống đa xử lý đối xứng (SMP, còn được gọi là shared-everything system) thay vì trên hệ song song phân tán không chia sẻ tài nguyên (shared-nothing system).

4.2 Thuật toán song song cho luật kết hợp mờ

Các thuật toán song song được đề xuất trước đây thường phải đồng bộ hóa giữa các BXL bởi chúng hoặc phải truyền thông tin về tập ứng cử viên (thuật toán phân phối độ hỗ trợ, thuật toán phân phối ứng cử viên) hoặc phải truyền dữ liệu cho nhau (thuật toán phân phối dữ liệu). Do phải truyền thông và đồng bộ hóa trong suốt quá trình tính toán nên các thuật toán trên không được xem là song song lý tưởng. Với cách tiếp cận luật kết hợp mờ ở phần trên, tôi xin đề xuất một thuật toán song song gần lý tưởng để khai phá dạng luật này. Thuật toán "lý tưởng" ở chỗ các BXL trong hệ thống gần như không phải truyền thông với nhau trong suốt quá trình tính toán, chúng chỉ cần truyền thông với nhau một lần duy nhất khi thuật toán kết thúc để tập hợp các luật khai phá được từ các BXL trong hệ thống.

4.2.1 Hướng tiếp cận

Theo bài toán khai phá luật kết hợp mờ tuần tự trong phần trên, mỗi thuộc tính i_u trong **I** được gắn với một tập các tập mờ F_{i_u} như sau:

$$F_{i_u} = \left\{ f_{i_u}^1, f_{i_u}^2, ..., f_{i_u}^k \right\}$$

Ví dụ, với CSDL trong bảng 8, chúng ta có:

$$F_{i_1} = \mathrm{F_{Tuổi}} = \{Tuổi\_trẻ, Tuổi\_trung\_niên, Tuổi\_già\} \text{ (với } k = 3)$$

$$F_{i_2} = \mathrm{F_{Cholesterol}} = \{Cholesterol\_thấp, Cholesterol\_cao\} \text{ (với } k = 2)$$

Luật kết hợp mờ có dạng:

$$\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}.$$

Trong đó:

- $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}$ là các tập thuộc tính. $\mathbf{X} = \{x_1, x_2, \ldots, x_p\}$, $\mathbf{Y} = \{y_1, y_2, \ldots, y_q\}$. $x_i \neq x_j$ (nếu $i \neq j$) và $y_i \neq y_j$ (nếu $i \neq j$).
- $\mathbf{A} = \{f_{x1}, f_{x2}, \ldots, f_{xp}\}$, $\mathbf{B} = \{f_{y1}, f_{y2}, \ldots, f_{yq}\}$ là tập các tập mờ tương ứng với các thuộc tính trong **X** và **Y**. $f_{xi} \in F_{xi}$ và $f_{yj} \in F_{yj}$.

Mỗi thuộc tính mờ không phải chỉ là tên thuộc tính mà là một cặp bao gồm [<tên thuộc tính> + <tên tập mờ tương ứng>]. Với **I** = {*Tuổi*, *Cholesterol*, *Đường_trong_máu*, *Bệnh_tim*} như trong bảng 8 thì tập các thuộc tính mờ sẽ là:

$\mathbf{I_F}$ = {[*Tuổi*, *Tuổi_trẻ*] (1), [*Tuổi*, *Tuổi_trung_niên*] (2), [*Tuổi*, *Tuổi_già*] (3),
[*Cholesterol*, *Cholesterol_thấp*] (4), [*Cholesterol*, *Cholesterol_cao*] (5),
[*Đường_trong_máu*, *Đường_trong_máu_0*] (6),
[*Đường_trong_máu*, *Đường_trong_máu_1*] (7),
[*Bệnh_tim*, *Bệnh_tim_không*] (8), [*Bệnh_tim*, *Bệnh_tim_có*] (9)}

Bảng 16 - Tập các thuộc tính mờ sau khi mờ hóa từ CSDL ở bảng 8

Như vậy, sau khi mờ hóa $\mathbf{I_F}$ sẽ bao gồm 9 thuộc tính mờ so với 4 thuộc tính ban đầu. Sau khi mờ hóa, giá trị các bản ghi tại các thuộc tính của CSDL ban đầu cũng được chuyển về khoảng [0, 1] nhờ các hàm thuộc tương ứng. Yêu cầu của bài toán là tìm tất cả các luật kết hợp mờ trên tập thuộc tính $\mathbf{I_F}$ và tập các bản ghi $\mathbf{T_F}$.

Như chúng ta đã biết, tập các thuộc tính mờ (cả vế trái lẫn vế phải) của luật kết hợp mờ không chứa bất kỳ hai thuộc tính mờ nào có cùng thuộc tính nguồn (thuộc tính không mờ trong **I**) ban đầu. Ví dụ, những luật "*Tuổi_già* AND *Cholesterol_cao* AND *Tuổi_trẻ* => *Bệnh_tim_có*" hoặc "*Đường_trong_máu* >

120 AND *Bệnh_tim_không* => *Bệnh_tim_có*" là không hợp lệ bởi trong luật thứ nhất *Tuổi_già* và *Tuổi_trẻ* là hai thuộc tính mờ có cùng một nguồn gốc ban đầu là *Tuổi*, còn trong luật thứ hai, *Bệnh_tim_không* và *Bệnh_tim_có* cũng là hai thuộc tính mờ bắt nguồn từ thuộc tính *Bệnh_tim* ban đầu. Có hai lý do để khẳng định điều này. Thứ nhất, các thuộc tính mờ có cùng một nguồn gốc thường có giá trị mờ "loại trừ lẫn nhau" nên nếu chúng cùng xuất hiện trong một tập phổ biến thì độ hỗ trợ của tập phổ biến đó thường là nhỏ và có thể là rất nhỏ trong trường hợp chúng loại trừ nhau thật sự. Ví dụ, giá trị hàm thuộc đối với tập mờ *Tuổi_già* của một đối tượng nào đó mà cao thì giá trị hàm thuộc đối với tập mờ *Tuổi_trẻ* là nhỏ, vì không có người nào lại "vừa già vừa trẻ". Lý do thứ hai là những luật kết hợp mờ như thế thường không được tự nhiên và có ít ý nghĩa.

Như vậy, những luật kết hợp liên quan đến các thuộc tính có chung nguồn gốc là hoàn toàn độc lập với nhau, do đó chúng ta có thể tìm kiếm chúng bằng một thuật toán song song gần lý tưởng. Giả sử trong hệ thống của chúng ta có 6 BXL, chúng ta sẽ tìm cách chia $\mathbf{I_F}$ thành sáu phần cho 6 BXL này như sau:

Với BXL $\mathbf{P^1}$:

$$\mathbf{I_F}^1 = \{[Tuổi, Tuổi\_trẻ] (1),$$
$$[Cholesterol, Cholesterol\_thấp] (4),$$
$$[Đường\_trong\_máu, Đường\_trong\_máu\_0] (6),$$
$$[Đường\_trong\_máu, Đường\_trong\_máu\_1] (7),$$
$$[Bệnh\_tim, Bệnh\_tim\_không] (8), [Bệnh\_tim, Bệnh\_tim\_có] (9)\}$$
$$= \{1, 4, 6, 7, 8, 9\}$$

Với BXL $\mathbf{P^2}$: $\mathbf{I_F}^2 = \{1, 5, 6, 7, 8, 9\}$

Với BXL $\mathbf{P^3}$: $\mathbf{I_F}^3 = \{2, 4, 6, 7, 8, 9\}$

Với BXL $\mathbf{P^4}$: $\mathbf{I_F}^4 = \{2, 5, 6, 7, 8, 9\}$

Với BXL $\mathbf{P^5}$: $\mathbf{I_F}^5 = \{3, 4, 6, 7, 8, 9\}$

Với BXL $\mathbf{P^6}$: $\mathbf{I_F}^6 = \{3, 5, 6, 7, 8, 9\}$

Như vậy, chúng ta đã chia đều được 9 thuộc tính mờ cho 6 BXL, mỗi BXL được 6 thuộc tính. Hai thuộc tính được đưa ra để phân chia là *Tuổi* và *Cholesterol*. Đây là cách chia tối ưu bởi tích giữa số lượng tập mờ gắn với thuộc tính *Tuổi* (là

3) và số lượng tập mờ gắn với thuộc tính *Cholesterol* (là 2) vừa bằng số lượng BXL trong hệ thống (là 6).

Trong trường hợp chia tối ưu là chúng ta chia đều được tập các thuộc tính mờ cho tất cả các BXL trong hệ thống, tuy nhiên cũng có trường hợp chúng ta sử dụng một giải pháp chia "chấp nhận được" có nghĩa là có một vài BXL trong hệ thống được "nghỉ ngơi". Sau đây tôi xin đề xuất một thuật toán chia tập thuộc tính mờ cho các BXL, thuật toán này dựa trên chiến lược quay lui (backtracking) và sẽ dừng ngay khi tìm được nghiệm đầu tiên. Trong trường hợp không tìm được nghiệm đúng, thuật toán sẽ trả về một nghiệm "chấp nhận được".

Cho CSDL **D** với $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ là tập *n* thuộc tính, $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ là tập *m* bản ghi. Sau khi gắn các tập mờ cho các thuộc tính (còn gọi là quá trình mờ hóa), ta có CSDL $\mathbf{D_F}$ với $\mathbf{T_F}$ là tập các bản ghi mà các giá trị tại các trường thuộc đoạn [0, 1] (tính toán thông qua hàm thuộc của các tập mờ) và tập các thuộc tính mờ $\mathbf{I_F} = \{[i_1, f_{i1}^{1}], \ldots, [i_1, f_{i1}^{k1}], [i_2, f_{i2}^{1}], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^{1}], \ldots, [i_n, f_{in}^{kn}]\}$. Trong đó, f_{ij}^{u} là tập mờ thứ *u* được gắn với thuộc tính i_j và k_j là số lượng tập mờ gắn với thuộc tính i_j. Ví dụ, với CSDL **D** ở bảng 8, chúng ta có **I** = {*Tuổi*, *Cholesterol*, *Đường_trong_máu*, *Bệnh_tim*} và sau khi mờ hóa thì $\mathbf{D_F}$ có $\mathbf{I_F}$ như ở bảng 16. Khi đó, $k_1 = 3$, $k_2 = 2$, $k_3 = 2$, $k_4 = 2$ tương ứng là số lượng tập mờ gắn với 4 thuộc tính trong **I**.

Đặt tập $\mathbf{FN} = \{k_1\} \cup \{k_2\} \cup \ldots \cup \{k_n\} = \{s_1, s_2, \ldots, s_v\}$ ($v \le n$ vì có thể tồn tại những cặp k_i và k_j giống nhau) và **N** là số lượng BXL trong hệ thống, bài toán phân chia tập thuộc tính mờ cho các BXL như sau: *tìm một tập con **Fn** (khác rỗng) của **FN** sao cho tích các phần tử trong **Fn** bằng số lượng BXL (là **N**) trong hệ thống. Trong trường hợp không tìm thấy nghiệm đúng thì thuật toán sẽ trả về một nghiệm "chấp nhận được" tức là tích của các phần tử trong **Fn** là xấp xỉ dưới của **N***. Bài toán này có thể giải quyết bằng chiến lược quay lui. Với ví dụ trên, $\mathbf{FN} = \{3\} \cup \{2\} \cup \{2\} \cup \{2\} = \{3, 2\}$.

Thuật toán:

```
BOOLEAN Subset(FN, N, Idx)
1       k = 1;
2       Idx[1] = 0;
3       S = 0;
4       while (k > 0) {
```

```
5           Idx[k]++;
6           if (Idx[k] <= sizeof(FN)) {
7                 if (S * FN[Idx[k]] <= N) {
8                       if (S * FN[Idx[k]] == N)
9                             return TRUE;
10                      else {
11                            S *= FN[Idx[k]];
12                            Idx[k + 1] = Idx[k];
13                            k++;
14                      }
15                }
16          } else {
17                k--;
18                S /= FN[Idx[k]];
19          }
20    }
21    return FALSE;

FindSubset(FN, N, Idx, Fn)
1     for (n = N; n > 0; n--)
2           If (Subset(FN, n, Idx)) {
3                 Fn = {FN[i] | i ∈ Idx}
4                 return;
5           }
```

Bảng 17 - Thuật toán hỗ trợ việc chia tập thuộc tính mờ cho các BXL

Trong ví dụ trên, sau khi tính toán, **Fn** sẽ bằng {3, 2}. Thuật toán trên cũng đảm bảo việc tìm nghiệm "chấp nhận được" trong trường hợp không tìm được nghiệm đúng do trong hàm **FindSubset** chúng ta đã giảm dần *n* để tìm xấp xỉ dưới của **N**.

4.2.2 Thuật toán song song cho luật kết hợp mờ

Đầu vào (inputs): CSDL **D** với tập thuộc tính **I** và tập bản ghi **T**. Số lượng BXL **N**. Độ hỗ trợ tối thiểu *minsup* và độ tin cậy tối thiểu *minconf*.

Đầu ra (outputs): tập tất cả các luật kết hợp mờ.

Thuật toán song song khai phá luật kết hợp mờ bao gồm các bước sau:

- (1) Mờ hóa CSDL **D** để chuyển về $\mathbf{D_F}$ với tập thuộc tính mờ $\mathbf{I_F}$ và tập bản ghi $\mathbf{T_F}$.

- (2) Dùng thuật toán trong bảng 17 để phân chia tập $\mathbf{I_F}$ cho **N** BXL trong hệ thống.
- (3) Tùy theo việc phân chia tập thuộc tính mờ ở bước (2) để phân chia dữ liệu cho các BXL. Mỗi BXL $\mathbf{P^i}$ chỉ cần những trường liên quan đến tập thuộc tính mờ được phân cho nó.
- (4) Mỗi BXL $\mathbf{P^i}$ sử dụng thuật toán tuần tự tìm luật kết hợp mờ trong bảng 10 để sinh luật. Đây là quá trình các BXL làm việc song song và độc lập với nhau.
- (5) Tập hợp luật sinh được trên tất cả các BXL trong toàn hệ thống chính là đầu ra của thuật toán này.

Thuật toán này không chỉ áp dụng được với luật kết hợp mờ mà còn áp dụng được với luật kết hợp với thuộc tính số và thuộc tính hạng mục (quantitive & categorical association rules).

4.3 Thử nghiệm và kết luận

- Thử nghiệm với số thuộc tính tăng dần và thời gian tìm kiếm luật
- Thử nghiệm với kích thước dữ liệu (số bản ghi tăng dần) và thời gian tìm kiếm luật
- Thử nghiệm với số BXL tăng dần và thời gian tìm kiếm luật

Chương V. Kết luận

Những vấn đề đã được giải quyết trong luận văn này

Với cách tiếp cận dựa trên những đề xuất đã có trong lĩnh vực nghiên cứu về KPDL, bản luận văn này là một sự tổng hợp những nét chính trong *khai phá dữ liệu* nói chung và *khai phá luật kết hợp* nói riêng cùng với một vài đề xuất mới. Sau đây là những điểm chính mà luận văn đã tập trung giải quyết.

Trong chương một, luận văn đã trình bày một cách tổng quan nhất về KPDL - cụ thể là định nghĩa về KPDL và những mục đích, động cơ thúc đẩy các nhà tin học chú trọng vào lĩnh vực nghiên cứu này. Phần này cũng trình bày sơ lược những kỹ thuật chính, những hướng tiếp cận được áp dụng để giải quyết các bài toán nhỏ hơn, cụ thể hơn như bài toán phân lớp, phân loại, .v.v. Nói tóm lại, chương này cung cấp cho người đọc một cái nhìn chung nhất về lĩnh vực nghiên cứu này.

Chương hai phát biểu lại bài toán *khai phá luật kết hợp* co R Agrawal đề xuất năm 1993. Ngoài việc phát biểu các khái niệm một cách hình thức, chương này còn phác họa một số nhánh nghiên cứu cụ thể như luật kết hợp với thuộc tính trọng số, luật kết hợp mờ, khai phá song song luật kết hợp, .v.v. Mục tiêu của chương này là trình bày tất cả những khái niệm cơ bản trong bài toán *khai phá luật kết hợp* và những mở rộng của bài toán này.

Dựa trên những đề xuất của [SA96] [MY98] [AG00] [KFW98], chương ba của luận văn đã trình bày sơ lược về luật kết hợp với thuộc tính trọng số cùng với những ưu, nhược điểm của nó. Tuy nhiên, mục tiêu chính của phần này là trình bày về luật kết hợp mờ, một dạng luật kết hợp mở rộng mềm dẻo hơn, gần gũi hơn của dạng luật kết hợp cơ bản trong chương hai. Những nội dung trình bày trong [AG00] [KFW98] quá vắn tắt và chưa nói lên hết được ý nghĩa của luật kết hợp mờ và đặc biết là mối quan hệ "tế nhị" giữa luật kết hợp mờ và phép kéo theo trong logic mờ. Luận văn lý giải được tại sao lại sử dụng hoặc phép lấy *min* hoặc phép tích đại số cho toán tử T-norm (T-chuẩn) trong công thức (3.6). Phần này cũng nêu lại thuật toán tìm luật kết hợp mờ trong [AG00] [KFW98] dựa trên thuật toán Apriori cùng với một vài sửa đổi nhỏ. Cuối chương này là một đề xuất về cách chuyển đổi từ luật kết hợp mờ sang luật kết hợp với thuộc tính trọng số. Đề

xuất này làm nổi bật ưu điểm của luật kết hợp mờ là khi cần thì nó cũng có thể được chuyển về dạng luật kết hợp thông thường một cách dễ dàng.

Chương bốn của luận văn đề xuất một thuật toán song song mới áp dụng cho bài toán khai phá luật kết hợp mờ. Với thuật toán này, các bộ xử lý trong hệ thống giảm được tối đa công việc truyền thông và đồng bộ hóa trong suốt quá trình tính toán. Sở dĩ thuật toán hoạt động khá "lý tưởng" như vậy là nhờ cách chia tập thuộc tính ứng cử viên một cách vừa công bằng vừa khôn khéo. Công bằng ở chỗ tập ứng cử viên được chia đều cho các bộ xử lý, còn khôn khéo ở chỗ các tập ứng cử viên sau khi chia cho từng bộ xử lý là hoàn toàn độc lập với nhau. Nhược điểm của thuật toán này là chỉ áp dụng cho luật kết hợp với thuộc tính số và luật kết hợp mờ cũng như chỉ thực hiện trên các hệ thống song song không chia sẻ (shared-nothing systems).

Trong quá trình thực hiện luận văn cũng như trong thời gian trước đó, tôi đã cố gắng tập trung nghiên cứu bài toán này cũng như đã tham khảo khá nhiều tài liệu liên quan. Tuy nhiên, do thời gian và trình độ có hạn nên không tránh khỏi những hạn chế và thiếu sót nhất định. Tôi thật sự mong muốn nhận được những góp ý cả về chuyên môn lẫn cách trình bày của luận văn từ bạn đọc.

Công việc nghiên cứu trong tương lai

Khai phá luật kết hợp là bài toán được khá nhiều nhà nghiên cứu quan tâm bởi nó được ứng dụng rộng rãi trong các lĩnh vực cũng như chứa đựng nhiều hướng mở rộng khác nhau. Ngay trong luận văn này, tôi cũng chỉ chọn một hướng nhỏ để nghiên cứu. Trong thời gian tới, chúng tôi sẽ mở rộng nghiên cứu của mình ra một số hướng sau:

- Khai phá luật kết hợp mờ với thuộc tính được đánh trọng số. Mục đích của bài toán này là tìm cách gắn trọng số cho các thuộc tính để biểu thị mức độ quan trọng của chúng đối với luật. Ví dụ, khi khai phá luật kết hợp liên quan đến bệnh tim mạch thì những thông tin về *huyết áp*, *lượng đường trong máu* và *cholesterol* quan trọng hơn là thông tin về *trọng lượng* và *tuổi tác*, do đó chúng được gắn trọng số lớn hơn. Bài toán này thực ra không mới mẻ mà đã được một vài người đề xuất, tuy nhiên nó chưa được giải quyết thuấu đáo.

- Thuật toán khai phá dữ liệu song song ở trên chỉ áp dụng cho hệ thống song song không chia sẻ (shared-nothing systems). Trong thời gian tới, chúng tôi sẽ nghiên cứu để cài đặt nó trên hệ thống song song chia sẻ như hệ đa xử lý đối xứng chẳng hạn.
- Mặc dù bài toán khai phá luật kết hợp là độc lập với cơ sở dữ liệu mà nó thao tác, tuy nhiên tôi mong muốn ứng dụng nó vào một cơ sở dữ liệu cụ thể để có thể tinh chỉnh và đưa ra được thông số tối ưu.

Tài liệu tham khảo

Tài liệu tiếng Việt:

[1]. [PDD99] Phan Đình Diệu. *Lô Gích trong Các Hệ Tri Thức*. Khoa Công nghệ, Đại học Quốc gia Hà Nội. Hà Nội - 1999.

[2]. [DMT03] Đinh Mạnh Tường. *Trí tuệ nhân tạo*. Khoa Công nghệ, Đại học Quốc gia Hà Nội. Hà Nội – 2003.

Tài liệu tiếng Anh:

[3]. [AR95] Alan Rea. Data Mining – An Introduction. The Parallel Computer Centre, Nor of The Queen's University of Belfast.

[4]. [AG00] Attila Gyenesei. A Fuzzy Approach for Mining Quantitative Association Rules. Turku Centre for Computer Science, TUCS Technical Reports, No 336, March 2000.

[5]. [AM95] Andreas Mueller. Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison. Department of Computer Science, University of Maryland-College Park, MD 20742.

[6]. [LHM99] Bing Liu, Wynne Hsu, and Yiming Ma. Mining Association Rules with Multiple Minimum Supports. In *ACM SIGKDD* International Conference on KDD & Data Mining (KDD-99), August 15-18, 1999, San Diego, CA, USA.

[7]. [KV01] Boris Kovalerchuk and Evgenii Vityaev. Data Mining in Finance – Advances in Relational and Hybrid Methods. Kluwer Academic Publishers, Boston – Dordrecht - London. 2001.

[8]. [MHT02] Bui Quang Minh, Phan Xuan Hieu, Ha Quang Thuy. Some Parallel Computing Experiments with PC-Cluster. In *Proc. of Conference on IT of Faculty of Technology, VNUH*. Hanoi 2002.

[9]. [KFW98] Chan Man Kuok, Ada Fu, and Man Hon Wong. Mining Fuzzy Association Rules in Databases. Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong.

[10]. [THH02] Do Van Thanh, Pham Tho Hoan, and Phan Xuan Hieu. Mining Association Rules with different supports. In *Proc. of the National Conference on Information Technology*, Nhatrang, Vietnam, May 2002.

[11]. [BCJ01] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: *A Maximal Frequent Itemset Algorithm for Transactional Databases*. Department of Computer Science, Cornell University.

[12]. [HKK97] Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Scalable Parallel Data Mining for Association Rules. Department of Computer Science, University of Minnesota, 4-192 EECS Building, 200 Union St. SE, Minneapolis, MN 55455, USA.

[13]. [PHM01] Jian Pei, Jiawei Han, and Runying Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. Intelligent Database Systems Research Lab, School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada.

[14]. [HK02] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. University of Illinois, Morgan Kaufmann Publishers 2002.

[15]. [HF95] Jiawei Han and Yongjian Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *Proc. of the 21st International Conference on Very Large Dadabases,* Zurich, Switzerland, Sep 1995.

[16]. [LZDRS99] Jinyan Li, Xiuzhen Zhang, Guozhu Dong, Kotagiri Ramamohanarao, and Qun Sun. Efficient Mining of High Confidence Association Rules without Support Thresholds. Department of CSSE, The University of Melbourne, Parkville, Vic, 3052, Australia.

[17]. [HG00] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. *ACM SIGKDD*, July 2000, Volume 2, Issue 1 – page 58 – 64.

[18]. [PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. Efficient Parallel Data Mining for Association Rules. In *Fourth International Conference on Information and Knowledge Management,* Baltimore, Maryland, Nov 1995.

[19]. [PYC98] Jong Soon Park (Sungshin Women's Univ, Seoul, Korea), Philip S. Yu (IBM T.J. Watson Res. Ctr.), and Ming-Syan Chen (National Taiwan Univ., Taipei, Taiwan). Mining Association Rules with Adjustable Accuracy.

[20]. [MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *KDD-1994: AAAI Workshop on Knowledge Discovery in Databases,* pages 181-192, Seattle, Washington, July 1994.

[21]. [LAZ65] L. A. Zadeh. Fuzzy sets. Informat. Control, 338-353, 1965.

[22]. [KMRTV94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *Proc. 3rd International Conference on Information and Knowledge Management,* pages 401-408, Gaithersburg, Maryland, November 1994.

[23]. [MM00] Manoel Mendonca. Mining Software Engineering Data: A Survey. University of Maryland, Department of Computer Science, A. V. Williams Building #3225 College Park, MD 20742. 2000.

[24]. [ZH99] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An Efficient Algorithm for Closed Association Rules Mining. RPI Technical Report 99-10, 1999.

[25]. [ZO98] Mohammed J. Zaki and Mitsunori Ogihara. Theoretical Foundations of Association Rules. In *3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery,* June 1998.

[26]. [ZPO01] Mohammed J. Zaki, Srinivasan Parthasarathy, and Mitsunori Ogihara. Parallel Data Mining for Association Rules on Shared-Memory Systems. In *Knowledge and Information Systems,* Vol. 3, Number 1, pages 1-29 February 2001.

[27]. [MPIS95] *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum, June 12, 1995.

[28]. [EMPI97] *MPI-2: Extensions to the Message-Passing Interface*, Message Passing Interface Forum, July 18, 1997

[29]. [JDMPI97] *MPI-2 Journal of Development*, Message Passing Interface Forum, July 18, 1997.

[30]. [PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. Laboratoire d'Informatique, Université Blaise Pascal – Clermont-Ferrand II, Complexe Scientifique des Cézeaux.

[31]. [ZHL98] Osmar R. Zaiane, Mohammad El-Hajj, and Paul Lu. Fast Parallel Association Rule Mining Without Candidacy Generation. University of Alberta, Edmonton, Alberta, Canada.

[32]. [DP01] Qin Ding and William Perrizo. Using Active Networks in Parallel Mining of Association Rules. Computer Science Department, North Dakota State University, Fargo ND 58105-5164.

[33]. [AY98] R. Agrawal and P. Yu. Online Generation of Association Rules. In *IEEE International Conference on Data Mining,* February 1998.

[34]. [AS96] Rakesh Agrawal and John Shafer. Parallel mining of association rules: Design, implementation and experience. Research Report RJ 10004, IBM Almaden Research Center, San Jose, California, February 1996.

[35]. [AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th International Conference on Very Large Databases,* Santiago, Chile, Sep 1994.

[36]. [AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of theACM SIGMOD Conference on Management of Data,* pages 207-216, Washington, D.C., May 1993.

[37]. [SA95] Ramakrishnan Srikant and Rekesh Agrawal. Mining Generalized Association Rules. In *Proc. of the 21st International Conference on Very Large Databases,* Zurich, Switzerland. Sep 1995.

[38]. [SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables. IBM Almaden Research Center, San Jose, CA 95120.

[39]. [MY98] R. J. Miller and Y. Yang. Association Rules over Interval Data. Department of Computer & Information Science, Ohio State University, USA.

[40]. [PMPIP] *RS/6000 SP: Practical MPI Programming*, Yukiya Aoyama & Jun Nakano, International Technical Support Organization, www.redbooks.ibm.com

[41]. [MS00] T. Murai and Y. Sato. Association Rules from a Point of View of Modal Logic and Rough Sets. In proceeding of the forth Asian Fuzzy Symposium, May 31, June 3, 2000, Tsukuba, Japan, pp, 427-432.

[42]. [HHMT02] Tran Vu Ha, Phan Xuan Hieu, Bui Quang Minh, and Ha Quang Thuy. A Model for Parallel Association Rules Mining from The Point of View of Rough Set. In *Proc. of International Conference on East-Asian Language Processing and Internet Information Technology*, Hanoi, Vietnam, January 2002.

[43]. [FSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI Press / The MIT Press 1996.

[44]. [WYY01] Wei Wang, Jiong Yang, and Philip S. Yu. Efficient Mining of Weighted Association Rules (WAR). IBM Watson Research Center.

[45]. [WMPPP] *Writing Message-Passing Parallel Programs with MPI*, Neil MacDonald, Elspeth Minty, Tim Harding, Simon Brown, Edinburgh Parallel Computing Centre, The University of Edinburgh.

[46]. [ZKM01] Zijian Zheng, Ron Kohavi, and Llew Mason. Real World Performance of Association Rule Algorithms. Blue Martini Software, 2600 Campus Drive, San Mateo, CA 94403, USA.

[47]. [ZHJ91] Zimmermann H. J. Fuzzy Set Theory and Its Applications. Kluwer Academic Publishers, 1991.