BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Thị Thúy Trinh

KHAI PHÁ TẬP MỤC PHỔ BIẾN MỜ DỰA TRÊN CẤU TRÚC CÂY VÀ KỸ THUẬT XỬ LÝ SONG SONG

LUẬN ÁN TIẾN SĨ NGÀNH MÁY TÍNH

Hà Nội - Năm 2023

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Thị Thúy Trinh

KHAI PHÁ TẬP MỤC PHỔ BIẾN MỜ DỰA TRÊN CẤU TRÚC CÂY VÀ KỸ THUẬT XỬ LÝ SONG SONG

LUẬN ÁN TIẾN SĨ NGÀNH MÁY TÍNH
Mã số: 9 48 01 04

Xác nhận của Học viện Khoa học và Công nghệ	**Người hướng dẫn 1** *(Ký, ghi rõ họ tên)*	**Người hướng dẫn 2** *(Ký, ghi rõ họ tên)*

Hà Nội - Năm 2023

LỜI CAM ĐOAN

Các kết quả trình bày trong luận án là công trình nghiên cứu của tôi được hoàn thành dưới sự hướng dẫn của PGS.TS. Nguyễn Long Giang và TS. Trương Ngọc Châu. Những kết quả trình bày là mới và chưa từng được công bố ở các công trình của người khác.

Tôi xin chịu trách nhiệm về những lời cam đoan của mình.

Hà Nội, tháng 5 năm 2023

Nghiên cứu sinh

Trần Thị Thúy Trinh

LỜI CẢM ƠN

Luận án tiến sĩ được hoàn thành tại Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam dưới sự hướng dẫn khoa học của PGS.TS. Nguyễn Long Giang và TS. Trương Ngọc Châu.

Trước tiên tôi xin được bày tỏ lòng biết ơn sâu sắc tới các thầy hướng dẫn PGS. TS. Nguyễn Long Giang và TS. Trương Ngọc Châu. Trong quá trình thực hiện luận án, nghiên cứu sinh đã nhận được nhiều định hướng khoa học, những bài học quý báu, sự hướng dẫn nhiệt tình từ các thầy hướng dẫn. Các thầy cũng đã luôn tận tâm động viên, khuyến khích và chỉ dẫn giúp đỡ nghiên cứu sinh hoàn thành được bản luận án này.

Tôi xin chân thành cảm ơn Học viện Khoa học và Công nghệ và Viện Công nghệ thông tin, Viện Hàn lâm Khoa học & Công nghệ Việt Nam đã tạo điều kiện thuận lợi cho tôi trong suốt quá trình nghiên cứu và thực hiện luận án.

Tôi xin cảm ơn các thầy cô và các đồng nghiệp ở các nơi mà tác giả tham gia viết bài đã có những góp ý thiết thực để tác giả có được những công bố như ngày hôm nay.

Tôi xin cảm ơn Ban Giám hiệu, ban lãnh đạo, tập thể cán bộ, giảng viên Trường Đào tạo Quốc tế và Khoa Công nghệ thông tin, Trường Đại học Duy Tân đã tạo điều kiện giúp đỡ tôi trong suốt thời gian học tập và nghiên cứu.

Cuối cùng, tác giả xin bày tỏ lòng biết ơn tới những người thân, bạn bè đã động viên, tạo động lực để tác giả hoàn thành luận án này.

Hà Nội, tháng 5 năm 2023

Trần Thị Thúy Trinh

MỤC LỤC

Danh mục các thuật ngữ

Tiếng Anh	**Ý nghĩa**
Cellular Automata	Automata di động
Compact Frequent Pattern	Mẫu phổ biến nhỏ gọn
Compressed Fuzzy Frequent Pattern	Mẫu mờ phổ biến nén
Complete Multiple Fuzzy Frequent Itemsets	Tập mục phổ biến mờ phức toàn bộ
Cellular learning automata	Automata di động học
Cellular learning automata Fuzzy Mining	Khai phá mờ bằng automata di động học
Differential Evolution	Tiến hóa vi phân
Expectation maximization	Cực đại hóa kỳ vọng
Expectation maximization coefficient	Biến thiên cực đại hóa kỳ vọng
Fuzzy Association Rules Mining	Khai phá luật kết hợp mờ
Fuzzy Frequent Itemset	Tập mục mờ phổ biến
Fuzzy Frequent Pattern	Mẫu mờ phổ biến
Fuzzy minimum confidence	Độ tin cậy mờ tối thiểu
Frequent Pattern	Mẫu phổ biến
Fuzzy Pre-order Size Coding	Mã mờ duyệt tiền tố - Kích thước
Fuzzy Pre-order Post-order Coding	Mã mờ duyệt tiền tố - hậu tố
Fuzzy Transaction Data-Mining	Khai phá dữ liệu giao dịch mờ
Gaussian mixture model	Mô hình Gaussian hỗn hợp
Irregular learning automata	Tự động học bất quy tắc
Integrated Multiple Fuzzy Frequent Pattern	Mẫu phổ biến mờ phức tích hợp
Multiple Fuzzy Frequent Pattern	Mẫu mờ phổ biến phức
Nodelist Fuzzy Frequent Pattern	Mẫu phổ biến mờ theo Nodelist
Nodelist Pre-order Size Fuzzy Frequent	Mẫu phổ biến mờ theo Nodelist tiền tố, kích thước
Pre-order Post-order Code	Mã tiền tố hậu tố
Transaction ID	Số thứ tự giao dịch

Bảng các ký hiệu, từ viết tắt

Từ viết tắt	**Ý nghĩa**
CA	Cellular Automata
CFP	Compact Frequent Pattern
CFFP	Compressed Fuzzy Frequent Pattern
CMFFP	Complete Multiple Fuzzy Frequent Itemsets
CLA	Cellular learning automata
CLA-F	Cellular learning automata Fuzzy Mining
DE	Differential Evolution
EM	Expectation maximization
EMC	Expectation maximization coefficient
FTDA	Fuzzy Transaction Data-Mining
FFI	Fuzzy Frequent Itemset
FFP	Fuzzy Frequent Pattern
fminconf	Fuzzy minimum confidence
FP	Frequent Pattern
FPOSC	Fuzzy Pre-order Size Coding
FPPC	Fuzzy Pre-order Post-order Coding
GMM	Gaussian mixture model
ICLA	Irregular learning automata
iMFFP	Integrated Multiple Fuzzy Frequent Pattern
MFFP	Multiple Fuzzy Frequent Pattern
MFAR	Mining Fuzzy Association Rules
NFFP	Nodelist Fuzzy Frequent Pattern
NPSFF	Nodelist Pre-order Size Fuzzy Frquent
PPC	Pre-order Post-order Code
TID	Transaction ID
TLL	Total Log Likelihood
UBFFP	Upper Bound Fuzzy Frequent Pattern
UBMFFP	Upper-bound Multiple fuzzy frequent pattern

Danh sách bảng biểu

Danh sách hình vẽ

MỞ ĐẦU

1. Tính cấp thiết của luận án và động lực nghiên cứu

Nghiên cứu gắn với ứng dụng thực tiễn là hoạt động cần nhiều thời gian và công sức không nhỏ của các nhà khoa học. Hơn nữa, trong thời đại công nghệ 4.0, các ứng dụng không chỉ hỗ trợ các tính năng kinh doanh cơ bản mà còn giúp con người đưa ra những dự đoán tương đối chính xác ở thời điểm hiện tại và tương lai. Sự phát triển mạnh mẽ của các hệ thống thông minh này làm tăng nhu cầu ứng dụng thực tế dẫn đến việc tạo ra một lượng lớn dữ liệu hàng ngày. Các công cụ và phương pháp thống kê truyền thống dựa trên nhu cầu ứng dụng, nhưng chúng không có khả năng xử lý lượng dữ liệu khổng lồ có nguồn gốc từ các ứng dụng này. Việc phân tích những dữ liệu như vậy là nhiệm vụ ưu tiên hàng đầu nếu không nó sẽ chuyển sang một hệ thống rất phức tạp và bất lợi. Để khắc phục vấn đề này, khai phá dữ liệu [1]–[3] là một trong những cách tiếp cận có lợi bằng cách hỗ trợ phân tích dữ liệu và tóm tắt dữ liệu thành thông tin hữu ích. Khái niệm khai phá dữ liệu là tạo ra thông tin chưa được xác định trước đó với mức độ liên quan lớn từ cơ sở dữ liệu để ra quyết định. Phụ thuộc vào sự đa dạng của kiến thức, các phương pháp khai phá dữ liệu có thể được chia thành các loại: luật kết hợp [4]–[8], phân loại [7], [9]–[11], phân cụm [12]–[14] và các mẫu tuần tự [15], [16]. Đặc biệt, khai phá luật kết hợp rất quan trọng đối với nghiên cứu khai phá dữ liệu [17]–[19]. Trong các giao dịch kinh doanh phổ biến, luật kết hợp có dạng $A \rightarrow B$ với mục đích tìm kiếm mối quan hệ của các mục trong cơ sở dữ liệu. Điều này giúp doanh nghiệp đưa ra quyết định trong việc hoạch định chiến lược kinh doanh, tiếp thị. Trong giai đoạn thứ nhất của quy trình khai phá luật kết hợp, các tập phổ biến được lấy từ một tập hợp dữ liệu nhất định. Từ các tập mục phổ biến được trích xuất, các luật kết hợp được xây dựng trong giai đoạn thứ hai. Giai đoạn chính của khai phá luật kết hợp là khai phá tập mục phổ biến vì cần rất nhiều nỗ lực để định vị các tập phổ biến trong một tập dữ liệu. Hầu hết các nghiên cứu trong lĩnh vực này đều tập trung vào việc nâng cao hiệu quả khai phá theo nhóm mục phổ biến về mặt thời gian và bộ nhớ.

Các thuật toán khai phá tập mục phổ biến hoặc luật kết hợp truyền thống [20], [21] hầu hết chỉ biểu diễn dữ liệu giao dịch ở dạng giá trị nhị phân, nghĩa là

nó liên quan đến sự xuất hiện của các mục; tuy nhiên, với cách tiếp cận rõ, để khai phá các tập mục phổ biến cho các luật kết hợp trong cơ sở dữ liệu có chứa dữ liệu định lượng là khó. Do tính dễ sử dụng và tương tự với suy luận của con người, lý thuyết tập mờ [22], [23] đang được sử dụng trong các hệ thống thông minh thường xuyên hơn [24]–[27]. Biểu diễn ngôn ngữ làm cho tri thức đơn giản hơn để con người dễ hiểu, do đó nó được sử dụng rộng rãi. Vì vậy, để khai phá các luật kết hợp mờ từ cơ sở dữ liệu định lượng, các miền của thuộc tính định lượng sẽ được chuyển đổi thành một tập mờ được thể hiện trong các biến ngôn ngữ bằng cách sử dụng hàm liên thuộc [28], cách tiếp cận này có thể làm giảm các tính toán. Một số thuật toán khai phá mờ đã được nghiên cứu và phát triển rộng rãi. Srikant và Agrawal [29] đã phát triển một cách tiếp cận để tìm luật kết hợp, tách cơ sở dữ liệu định lượng thành cơ sở dữ liệu nhị phân. Au và Chan đã phát triển F-APACS [30] để khai thác các luật kết hợp mờ bằng cách sử dụng các thuật ngữ ngôn ngữ để biểu diễn các luật. Kuok và cộng sự [31] đã thực hiện một phương pháp khai phá mờ để xử lý các thuộc tính có giá trị định lượng. Hong và cộng sự đã trình bày một thuật toán khai phá sử dụng lý thuyết tập mờ để chuyển đổi giá trị định lượng của mục thành các thuật ngữ ngôn ngữ dựa trên cơ chế giống như Apriori thông thường [32].

Thuật toán cho khai phá luật kết hợp mờ với hiệu suất nhanh và hiệu quả trên các tập dữ liệu lớn được đề xuất bởi Mangalampalli và Pudi [33]. Trong bài báo này, tác giả đã sử dụng phương pháp tidlist để tính tần suất của vị trí đặt, với các tính năng như cấu trúc dữ liệu sử dụng byte-vector biểu diễn tidlist, các thay đổi đối với cấu trúc dữ liệu tidlist được tạo để phù hợp với các giá trị thành viên mờ và giao dịch (TID), danh sách nén đã sử dụng góp phần tăng hiệu suất. Trước đây, Janikow đã kết hợp các cây quyết định tượng trưng trên các hệ thống dựa trên luật để điều khiển mờ [34] bằng cách sử dụng biểu diễn mờ. Watanabe và Fujioka [35], [36] đã định nghĩa sự dư thừa tương đương của các phần tử mờ và các định lý liên quan cho việc khai phá luật kết hợp mờ. Thuật toán được thiết kế đã áp dụng cơ chế giống như Apriori để sử dụng độ dư tương đương của các phần tử mờ dựa trên các khái niệm độ dư để khám phá các luật kết hợp mờ. Tác giả đã định nghĩa tính dư thừa của các luật kết hợp mờ là một khái niệm mới trong khai phá dữ liệu và đề xuất một thuật toán dựa trên việc sử dụng các luật dư thừa. Thuật

toán đã đánh giá các luật trước khi tính toán độ chính xác tối thiểu. Mục tiêu của thuật toán là tinh chỉnh thời gian dành cho việc khai phá luật và đồng thời cắt bỏ các luật thừa trong các ứng dụng khai phá dữ liệu. Tuy nhiên, hầu hết các phương pháp khai phá luật kết hợp mờ áp dụng Apriori [37] để tạo ra các ứng cử viên và kiểm tra sự hỗ trợ của chúng, do đó yêu cầu quét lại cơ sở dữ liệu nhiều lần, vì vậy nó gây ra quá trình chậm và không hiệu quả trong cơ sở dữ liệu lớn. Hơn nữa, với cách biểu diễn mờ trong các thuật toán trên, tập hợp mờ của các thuộc tính định lượng và hàm thành viên của chúng phụ thuộc vào ý kiến chủ quan của chuyên gia hoặc tính sẵn có. Vấn đề này gây ra ranh giới "sắc nét" giữa các khoảng mờ, vì vậy khó có thể xác định mức độ của hàm liên thuộc cho các phần tử gần ranh giới của khoảng. Đây là khoảng trống thứ nhất được xác định trong vấn đề nghiên cứu của luận án.

Thay vì sử dụng cách tiếp cận thông thường theo Apriori, Lin et al. đã triển khai phương pháp cây phổ biến mờ (FFP)-tree [38], [39] để khai phá tập mục phổ biến mờ dựa trên cơ chế phát triển mẫu. Tiếp cận này đã áp dụng cả lý thuyết tập mờ và cấu trúc cây FP (Frequent pattern) để xây dựng cây FFP (Fuzzy Frequent Pattern) có thể được sử dụng cho quá trình khai phá. Các biến ngôn ngữ được chuyển đổi cùng với mức độ thuộc của chúng được sắp xếp theo thứ tự tăng dần của mỗi giao dịch, do đó giữ được tính chất đóng (downward closure property) để xây dựng đệ quy cây điều kiện và khai phá các mục phổ biến mờ cần thiết. Cách tiếp cận này có thể yêu cầu rất nhiều thời gian tính toán khi quy mô giao dịch rất lớn. Thuật toán nén cây phổ biến mờ (CFFP – Compact Fuzzy Frequent Pattern)-tree [40] sau đó được thiết kế để giảm kích thước của cây FFP. Do đó, một mảng được gắn với mỗi nút bằng cách bảo toàn các giá trị mờ cho biến ngôn ngữ được xử lý hiện tại với bất kỳ tập mục tiền tố nào của nó trong đường đi. Mặc dù số lượng nút cây của cây CFFP giảm đáng kể so với thuật toán cây FFP, nhưng cần phải giữ thêm một mảng của mỗi nút để lưu trữ các giá trị thành viên của nút được xử lý hiện tại với bất kỳ biến ngôn ngữ nào của nó trong đường đi. Do đó, nó yêu cầu dung lượng bộ nhớ để lưu giữ những thông tin đó, điều này không hiệu quả trong một cơ sở dữ liệu thưa. Để giải quyết hạn chế này, thuật toán cây mẫu phổ biến mờ giới hạn trên (UBFFPT - upper-bound fuzzy frequent pattern) [41] sau đó được thiết kế để giữ không chỉ cấu trúc cây dày đặc mà còn có thể khai phá

các tập mục phổ biến mờ từ giới hạn bộ nhớ so với cây FFP và thuật toán cây CFFP. Thuật toán cây UBFFPT có thể khai phá các mục phổ biến mờ hiệu quả mà giữ nguyên kích thước của các nút cây như thuật toán cây CFFP, việc sử dụng bộ nhớ và tính toán có thể giảm đáng kể. Các thuật toán trên chỉ sử dụng một thuật ngữ ngôn ngữ duy nhất để biểu diễn mục được xử lý trong cơ sở dữ liệu, do đó thông tin được phát hiện có thể không đầy đủ. Nhiều thuật toán liên quan đến khai phá tập phổ biến mờ kép [42]–[44] được đề xuất nhằm giúp tri thức được khai phá đầy đủ hơn so với các phương pháp truyền thống. Hong và cộng sự [42] sau đó đã phát triển cấu trúc dựa trên cây với ý tưởng tương tự về cây FP và FFPT [38] nhưng duy trì nhiều tập mục phổ biến mờ 1-item với cây MFFP. Do đó, không chỉ biến ngôn ngữ đơn lẻ được giữ để biểu diễn cho một mục mà tất cả các mục có giá trị mờ của chúng không nhỏ hơn ngưỡng hỗ trợ tối thiểu. Vì vậy, thông tin đầy đủ hơn được lưu giữ để ra quyết định hiệu quả. Hơn nữa, ý tưởng tương tự sau đó được áp dụng cho cây CMFFP [43] và cây UBMFFP [44]. Với thông tin đầy đủ hơn về nhiều mẫu phổ biến mờ dẫn xuất, các chiến lược hiệu quả do đó có thể đạt được để ra quyết định. Tuy nhiên, trong các thuật toán này, việc khai phá các tập phổ biến mờ được thực hiện một cách đệ quy từ cấu trúc cây, do đó nó yêu cầu một bộ nhớ lớn để lưu trữ các cây tạm thời. Đây là khoảng trống thứ hai luận án sẽ giải quyết.

Khai phá tập phổ biến từ nhiều tập dữ liệu mờ được đề cập trong bài báo [45]. Trong bài báo, tác giả kết hợp nhiều bảng bằng cách sử dụng lược đồ sao tìm các luật kết hợp đa cấp mờ trong mô hình cơ sở dữ liệu quan hệ, có khả năng xử lý nhiều bảng. Thuật toán sử dụng phép nối và thực thể để nhận ra các tập mục phổ biến. Tuy nhiên, kết quả của bài báo vẫn còn nhiều hạn chế trong việc tính toán hỗ trợ của các tập mục liên quan đến các kết nối khác có chứa thuộc tính mờ. Phương pháp khác như [46] sử dụng thuật toán tiến hóa vi phân (DE) để khai phá các luật kết hợp mờ có ý nghĩa thống kê được tối ưu hóa có số lượng lớn và các giá trị đo lường có nghĩa với sự kiểm soát chặt chẽ đối với rủi ro của các luật suy đoán. DE được đề xuất cũng có thể làm tăng đáng kể số lượng các luật được tạo ra và tính phù hợp của các giá trị đo lường của các luật hơn là thuật toán di truyền. Ngoài ra, thuật toán dựa trên mẫu được đề xuất trong [47] nhằm mục đích tìm các luật kết hợp mờ từ các tập dữ liệu định lượng lớn. Thuật toán được thiết kế để

chuyển đổi các luật kết hợp mờ thành các bản sao nhị phân. Sau đó, phương pháp sử dụng lý thuyết giới hạn trung tâm để lấy mẫu thay thế tập dữ liệu lớn ban đầu và giảm kích thước dữ liệu. Sự đóng góp này đã giúp giảm chi phí thời gian. Hơn nữa, thuật toán có thể hạn chế độ lệch của độ hỗ trợ tập phổ biến mờ trong một phạm vi rất nhỏ với xác suất cao. Nhiều nghiên cứu khác nhau đã được thực hiện không chỉ để cải thiện hiệu suất mà còn cải thiện tốc độ tìm kiếm các luật kết hợp mờ với bảng băm, lược đồ hoặc cấu trúc dữ liệu cây [40], [41], [43], [44]. Thuật toán khai phá tập mục mờ phổ biến FFI-Miner [48] được phát triển để khai phá tập đầy đủ các FFI mà không cần tạo ứng viên. Nó sử dụng cấu trúc danh sách mờ để giữ thông tin cần thiết cho quá trình khai phá sau này. Thuật toán sử dụng chiến lược cắt tỉa hiệu quả cũng được phát triển để giảm không gian tìm kiếm, do đó đẩy nhanh quá trình khai phá để phát hiện trực tiếp các tập mục mờ phổ biến. Các mẫu phổ biến là các tập mục được tìm thấy trong một số lượng đáng kể các giao dịch. Cùng với sự gia tăng kích thước dữ liệu, các loại dữ liệu không đồng nhất và biến thể dữ liệu cực kỳ động. Do đó, việc mở rộng các thuật toán khai phá mờ hiệu quả cho kỷ nguyên dữ liệu lớn là một vấn đề quan trọng việc khai phá bằng cách áp dụng các kỹ thuật xử lý song song đã trở thành một cách khả thi để khắc phục vấn đề thời gian xử lý. Đây là khoảng trống thứ ba được xác định trong luận án.

Tại Việt Nam, khai phá luật kết hợp đã được các nhóm nghiên cứu tại Viện Công nghệ Thông tin thuộc Viện Khoa học và Công nghệ Việt Nam như luận án tiến sĩ của Nguyễn Huy Đức [49] giới thiệu thuật toán FSM là thuật toán nhanh khai phá tất cả các tập mục cổ phần cao trong cơ sở dữ liệu giao tác và đề xuất thuật toán AFSM (*Advanced FSM)* dựa trên các bước của thuật toán FSM với phương pháp mới tỉa hiệu quả hơn các tập mục ứng viên. Luận án tiến sĩ của Nguyễn Long Giang [50] trình bày về phương pháp khai phá dữ liệu sử dụng lý thuyết tập thô. Bài báo của tác giả Nguyễn Công Hào [51] trình bày một phương pháp xử lý luật kết hợp mờ dựa trên đại số gia tử. Nhóm nghiên cứu của PGS. TS. Võ Đình Bảy và GS. TS. Lê Hoài Bắc đưa ra phương pháp khai phá tập mục phổ biến trong cơ sở dữ liệu rõ như [52]–[55], đây có thể được xem là nền tảng cho những nghiên cứu trong luận án.

Luận án này nhằm giải quyết ba khoảng trống được xác định ở trên. Việc nghiên cứu giải quyết những vấn đề đó là thực sự cần thiết không chỉ ở phương diện phát triển lý thuyết mà cả ở phương diện ứng dụng thực tế. Đó là động lực để tác giả luận án thực hiện nghiên cứu đề tài "**Khai phá tập mục phổ biến mờ dựa trên cấu trúc cây và kỹ thuật xử lý song song**" để đưa ra các phương pháp mới hiệu quả về khai phá tập mục phổ biến và khai phá các luật kết mờ dựa trên lý thuyết tập mờ.

2. Mục tiêu, đối tượng và phạm vi nghiên cứu của luận án

a. Mục tiêu nghiên cứu

Mục tiêu của luận án nhằm đề xuất các giải pháp khai phá tập mục phổ biến mờ trong cơ sở dữ liệu định lượng, khắc phục vấn đề "sắc nét" khi phân vùng dữ liệu mờ cho các thuộc tính có giá trị định lượng.

Cụ thể, luận án tập trung đề xuất các giải pháp nhằm:

- Xác định các tập mờ cho mỗi thuộc tính định lượng trong cơ sở dữ liệu thông qua kỹ thuật phân cụm.
- Giảm bộ nhớ lưu trữ trong quá trình khai phá tập mục phổ biến mờ
- Giảm thời gian xử lý trong việc khai phá tập mục phổ biến mờ trong các cơ sở dữ liệu lớn.

b. Đối tượng nghiên cứu

- Các thuật toán khai phá tập mục phổ biến trong cơ sở dữ liệu giao dịch
- Các thuật toán khai phá tập mục phổ biến mờ, khai phá luật kết hợp mờ trong cơ sở dữ liệu định lượng.

c. Phạm vi nghiên cứu

- Luận án nghiên cứu các luật kết hợp mờ, tập mục phổ biến mờ trong cơ sở dữ liệu định lượng.
- Tổng hợp các công bố khoa học liên quan đến các phương pháp khai phá tập mục phổ biến mờ.
- So sánh thực nghiệm với các thuật toán đã có

3. Phương pháp nghiên cứu

Luận án đã sử dụng các phương pháp nghiên cứu sau:

- Tổng hợp và đánh giá các kết quả đã được công bố về các phương pháp khai phá tập mục phổ biến mờ từ nhiều nguồn thông tin thu thập được. Trên cơ sở đó đề xuất các kết quả mới, đánh giá kết quả mới bằng việc cài đặt thử nghiệm một số thuật toán. Áp dụng kết quả để giải quyết một bài toán trong thực tiễn.
- Phương pháp so sánh: được sử dụng để so sánh các kỹ thuật, thuật toán đã được đề xuất để giải quyết những vấn đề nghiên cứu liên quan, từ đó hình thành ý tưởng cho thuật toán mới cho vấn đề nghiên cứu.
- Phương pháp thực nghiệm: Các thuật toán được đề xuất đều được thực nghiệm trên các tập dữ liệu thực để đánh giá sự đúng đắn và tính khả thi của thuật toán.

4. Các đóng góp chính của luận án

Những đóng góp chính của luận án là đề xuất và giải quyết các vấn đề sau:

- Đề xuất phương pháp xác định các tập mờ cho mỗi thuộc tính định lượng trong cơ sở dữ liệu thông qua kỹ thuật phân cụm. Cụ thể hơn, luận án trình bày kỹ thuật phân cụm EMC. Mục tiêu của các thuật toán này là chia dữ liệu thành các cụm có ý nghĩa. Sau đó, các cụm này được sử dụng để phân loại mỗi thuộc tính định lượng như một tập mờ và xác định các hàm thuộc của chúng. Các bước này được kết hợp thành một thuật toán tối ưu để tìm các tập mờ dựa trên lý thuyết thống kê. [CT2], [CT4].
- Đề xuất phương pháp khai phá luật kết hợp mờ trong cơ sở dữ liệu định lượng sử dụng cấu trúc dữ liệu Node-list. Với đề xuất này, thuật toán chỉ truy cập cơ sở dữ liệu hai lần: lần thứ nhất để chuyển các giá trị định lượng sang các tập mờ được thực hiện tự động thông qua thuật toán phân cụm mờ, lần thứ hai tính toán độ hỗ trợ mờ trong cơ sở dữ liệu mờ và xây dựng cây. Quy trình khai phá tập mục mờ phổ biến dựa trên PP_code hoặc POS_code giúp hạn chế mức tiêu thụ bộ nhớ được yêu cầu. [CT1], [CT2], [CT5].

- Đề xuất một phương pháp xử lý song song để khai phá các tập phổ biến mờ sử dụng phương pháp tiếp cận automata di động học(Cellular learning automata). Theo CLA, không gian được biểu diễn như một mạng, với mỗi phần tử là một ô. Từng dòng một, dữ liệu giao dịch sẽ được đọc và đồng thời được chuyển đến các ô, chúng xử lý song song với nhau. Phương pháp này không sử dụng quy tắc vùng lân cận, một loại tự động dữ liệu được gọi là tự động học di động bất quy tắc (ICLA) được sử dụng để tạo danh sách vùng lân cận cho mỗi ô. Thông qua việc sử dụng các ô dữ liệu tự trị này, việc khai phá các tập mục mờ phổ biến được thực hiện. Quá trình này rút ngắn thời gian thực thi của thuật toán. [CT3].

5. Bố cục luận án

Luận án gồm phần Mở đầu, 03 chương và phần kết luận.

- Phần Mở đầu: Trình bày sự cần thiết và động lực nghiên cứu của đề tài; mục tiêu, đối tượng, phạm vi nghiên cứu; phương pháp nghiên cứu; những đóng góp chính và cấu trúc của luận án.
- Chương 1: Cơ sở lý thuyết

  Chương này trình bày các khái niệm, tổng quan về luật kết hợp; logic mờ; tập mục phổ biến mờ, luật kết hợp mờ, các thuật toán khai phá tập mục phổ biến mờ. Từ đó xác định các tồn tại và xác định các vấn đề cụ thể trong luận án.
- Chương 2: Các phương pháp khai phá tập mục phổ biến mờ dựa trên cấu trúc cây.

  NCS trình bày các phương pháp đề xuất về khai phá tập mục phổ biến mờ; có sử dụng phân vùng dữ liệu mờ cho các thuộc tính có giá trị định lượng.
- Chương 3: Khai phá tập mục phổ biến mờ sử dụng phương pháp xử lý song song.

  NCS trình bày các lý thuyết liên quan đến automata di động học và đề xuất thuật toán xử lý song song trong khai phá luật kết hợp mờ.
- Kết luận và hướng phát triển

Chương 1 CƠ SỞ LÝ THUYẾT

Trong chương này, NCS trình bày các khái niệm cơ bản về luật kết hợp, luật kết hợp định lượng, logic mờ, luật kết hợp mờ và các nghiên cứu liên quan đến luật kết hợp mờ. Từ đó, xác định các vấn đề còn tồn tại cần giải quyết trong chương 2.

1.1 Luật kết hợp

1.1.1 Các khái niệm cơ bản về luật kết hợp [56]

Định nghĩa 1.1 Cơ sở dữ liệu giao tác:

Giả sử $I = \{i_1, i_2, \dots, i_m\}$ là tập các mục. $D = \{T_1, T_2, \dots, T_n\}$ là một tập các giao tác, được gọi là cơ sở dữ liệu giao tác, trong đó mỗi giao tác t trong D có dạng (tid, X) trong đó, mỗi giao tác t có định danh tid và tập mục t-itemset, $t = (tid, t - itemset)$; X được gọi là tập mục itemset nếu $X \subseteq I$.

Ví dụ: CSDL giao tác *D* được mô tả như bảng sau

Bảng 1.1: Cơ sở dữ liệu giao tác

Tid	Items
T1	Bánh mì, Sữa
T2	Bánh mì, Tã, Bia, Trứng
T3	Sữa, Tã, Bia, Nước ngọt
T4	Bánh mì, Sữa, Tã, Bia
T5	Bánh mì, Sữa, Tã, Nước ngọt

Bảng 1.1 biểu diễn cơ sở dữ liệu giao tác, trong đó tập $I = \{Bánh\ mì, Sữa, Tã, Bia, Trứng, Nước\ ngọt\}$ là tập mục tên các mặt hàng (hay gọi là mục) và 5 giao tác. Mỗi giao tác biểu diễn danh sách các mặt hàng đã mua. Ví dụ, giao tác T1 có chứa các mục $\{Bánh\ mì, Sữa\}$.

Định nghĩa 1.2: Độ hỗ trợ của tập mục

Độ hỗ trợ của một tập mục *X* trong cơ sở dữ liệu giao tác *D* ký hiệu là *sup (X)* là số giao dịch chứa tập mục X, được tính bởi công thức sau:

$$sup(X) = |t|\ X \subseteq t, t \in D| \qquad (1.1)$$

Trong đó ký hiệu |.| là số giao tác.

Ví dụ: trong CSDL ở bảng 1.1, độ hỗ trợ của tập mục $\{Bia, Tã, Sữa\}$ là 2 vì có hai giao tác chứa 3 mục trên.

Định nghĩa 1.3: Tập mục phổ biến

Một tập mục X có trong cơ sở dữ liệu giao tác D được gọi là phổ biến nếu độ hỗ trợ của nó ($sup(X)$) lớn hơn hoặc bằng ngưỡng độ hỗ trợ tối thiểu (minsup) cho trước do người dùng định nghĩa. Vì vậy, độ hỗ trợ được xem là tần suất xuất hiện đồng thời của các mục.

Định nghĩa 1.4: Luật kết hợp

Một luật kết hợp là một mệnh đề kéo theo có dạng X →Y, trong đó X và Y là các tập mục thoả mãn điều kiện: $X \subseteq I$, $Y \subseteq I$ và $X \cap Y = \emptyset$. Đối với luật kết hợp X → Y, X được gọi là tiền đề, Y được gọi là kết quả của luật.

Định nghĩa 1.5 : Độ hỗ trợ của một luật

Cho luật kết hợp $r = X \rightarrow Y$, độ hỗ trợ của luật r ký hiệu là sup(r) là tỉ số giữa số lượng các giao tác T ⊆ D có chứa cả tập mục X và tập mục Y với tổng số giao tác trong D được xác định như sau:

$$sup(r) = \frac{|\{T \in D | T \supset X \cup Y\}|}{|D|} \quad (1.2)$$

Định nghĩa 1.6 Độ tin cậy của một luật

Cho luật kết hợp $r = X \rightarrow Y$, độ tin cậy của luật r ký hiệu là conf(r) là tỉ số giữa số lượng các giao tác T ⊆ D có chứa cả tập mục X và tập mục Y với tổng số giao tác trong D chứa tập mục X, được xác định như sau:

$$conf(r) = \frac{|\{T \in D | T \supset X \cup Y\}|}{|\{T \in D | T \supset X\}|} = \frac{sup(X \cup Y)}{sup(X)} \quad (1.3)$$

Ví dụ: Xem xét một luật $\{Diapers, Milk\} \rightarrow \{Beer\}$. Vì độ hỗ trợ của tập mục $\{Beer, Diapers, Milk\}$ là 2 và tổng số giao tác là 5, do đó độ hỗ trợ của luật là $\frac{2}{5} =$ 0.4. Độ tin cậy của luật thu được bởi tỉ số giữa độ hỗ trợ của $\{Beer, Diapers, Milk\}$

và độ hỗ trợ của $\{Diapers, Milk\}$. Vì có 3 giao tác chứa $\{Diapers, Milk\}$ nên độ hỗ trợ của luật sẽ là $\frac{2}{3} = 0.67$.

Định nghĩa 1.7: Luật kết hợp mạnh

Cho luật kết hợp $r = X \rightarrow Y$, nếu luật r thỏa mãn cả hai ngưỡng là độ hỗ trợ tối thiểu (minsup) và độ tin cậy tối thiểu (minconf) được gọi là luật kết hợp mạnh, tức là:

$$sup(r = X \rightarrow Y) = P(X \cup Y) \geq minsup$$

$$conf(r = X \rightarrow Y) = P(X \cup Y) = \frac{sup(X \cup Y)}{sup(X)} \geq minconf$$

Ví dụ: Xem xét CSDL trong bảng 1.1, luật kết hợp $\{Diapers, Milk\} \rightarrow \{Beer\}$ có nghĩa là trong cùng một giao dịch, nếu mặt hàng Diapers và Milk được mua thì mặt hàng Beer cũng được mua.

Phát biểu bài toán: Bài toán luật kết hợp được phát biểu như sau [49]:

Cho một cơ sở dữ liệu giao tác D, độ hỗ trợ tối thiểu minsup, độ tin cậy tối thiểu minconf. Hãy tìm tất cả các luật kết hợp có dạng $X \rightarrow Y$ thỏa mãn độ hỗ trợ $sup(X \cup Y) \geq minsup$ và độ tin cậy $conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)} \geq minconf$

Hầu hết các thuật toán khai phá luật kết hợp đều theo hướng chia bài toán thành hai pha cụ thể:

- Pha 1: Tìm tất cả các tập mục phổ biến từ cơ sở dữ liệu, tức là tìm tất cả tập mục có độ hỗ trợ lớn hơn hoặc bằng độ hỗ trợ tối thiểu ($sup(X) \geq minsup$).
- Pha 2: Sinh tất cả các luật có độ tin cậy từ tập mục phổ biến đã tìm thấy ở pha thứ 1. Nếu X là tập mục phổ biến, thì luật sinh ra từ X có dạng $A \rightarrow B$ trong đó $B \subset X$, và $A = X - B$ nếu độ tin cậy của luật $A \rightarrow B$ có độ tin cậy lớn hơn độ tin cậy tối thiểu cho trước *minconf.*

1.1.2 Luật kết hợp trong cơ sở dữ liệu nhị phân

Luật kết hợp nhị phân đề cập đến các luật cổ điển trong bài toán phân tích giỏ hàng. Ở đây các sản phẩm có thể có trong giao dịch hoặc không, chỉ tạo ra các giá trị kiểu boolean (được biểu diễn bằng 1 và 0). Do đó, mọi mục trong giao dịch có thể

được xác định là một thuộc tính nhị phân với miền [0,1]. Mô hình được định nghĩa trong [56] như sau:

Cho $I = \{i_1, i_2, \dots, i_m\}$ là một tập các thuộc tính nhị phân, gọi là các mục. Cho T là cơ sở dữ liệu giao dịch. Mỗi giao dịch t được biểu diễn như là vecto nhị phân với $t[k] = 1$ nếu giao dịch t có chứa mục i_k và $t[k] = 0$ nếu ngược lại. Cho X là một tập mục chứa trong I, ta nói một giao dịch t thỏa mãn X nếu mọi mục trong X, $i_k \in X, t[k] = 1$.

Bảng 1.2: Ví dụ về cơ sở dữ liệu nhị phân

Tid	A	B	C	D	E
1	1	0	1	1	1
2	1	1	1	0	0
3	0	1	1	0	0
4	1	1	1	0	1
5	1	0	1	1	0
6	0	1	1	1	0

Bảng 1.2 mô tả cơ sở dữ liệu nhị phân, CSDL bao gồm sáu giao tác và năm mục được ký hiệu là A - E. Trong ví dụ này, giao dịch TID =1 có các mục A, C, D, E nên các mục này nhận giá trị 1, còn các mục B không có trong CSDL nên B nhận giá trị 0.

Bài toán khai phá luật kết hợp nhị phân tập trung chủ yếu ở giai đoạn khai phá tập mục phổ biến, vì đây là giai đoạn phức tạp, đòi hỏi nhiều chi phí về thời gian và tính toán. Hai thuật toán điển hình trong khai phá tập mục phổ biến là thuật toán Apriori [5] và FP-growth [57]. Thuật toán Apriori tiêu biểu cho phương pháp sinh ra các tập mục ứng viên rồi duyệt cơ sở dữ liệu kiểm tra độ hỗ trợ của chúng, thuật toán FP-Growth đại diện cho phương pháp không sinh ra các tập mục ứng viên mà nén cơ sở dữ liệu theo cấu trúc cây.

1.1.3 Luật kết hợp trong cơ sở dữ liệu định lượng

Theo dạng luật kết hợp nhị phân này thì các mục chỉ được quan tâm là có hay không xuất hiện trong cơ sở dữ liệu giao tác chứ không quan tâm về mức độ hay tần xuất xuất hiện. Trong thực tế, cơ sở dữ liệu không chỉ chứa các thuộc tính nhị phân

mà còn chứa các thuộc tính định lượng và phân loại mà không thể khai phá bằng kỹ thuật cổ điển. Việc khai phá các luật trong loại dữ liệu như vậy có thể được gọi là bài toán luật kết hợp định lượng [29]. Chiến lược khai phá luật kết hợp định lượng được thực hiện bằng cách chuyển đổi các thuộc tính có giá trị định lượng sang giá trị nhị phân. Trong phương pháp này, mỗi giá trị định lượng/phân loại có dạng $\langle attribute, value \rangle$ được ánh xạ sang giá trị nhị phân. Sau đó, các kỹ thuật khai phá luật kết hợp nhị phân được thực hiện để tìm luật. Tuy nhiên, khi miền giá trị của thuộc tính là quá lớn hoặc liên tục thì phương pháp này không hiệu quả [58]**.** Rời rạc hóa các thuộc tính liên tục thành các khoảng thời gian khác nhau là một cách phổ biến để giải quyết vấn đề này. Sau khi rời rạc hóa, các thuộc tính được coi là thuộc tính phân loại [59]. Chẳng hạn, một thuộc tính x có giá trị từ 20 đến 100 có thể được chia thành các khoảng (20–30, 30–40,…, 90–100). Nếu một giá trị là 62, thì khoảng (60–70) trở thành 1 và khoảng còn lại các khoảng vẫn là 0. Ví dụ, Tuổi ∈ [20,50] và Lương ∈ [10,20] → Số xe ∈ [1,2] là một dạng của luật kết hợp định lượng [60]. Vấn đề chính của sự rời rạc hóa các giá trị là mất thông tin và kết quả kém [61]. Ngoài ra, hiệu quả phụ thuộc vào các khoảng xác định, trong khi việc xác định các khoảng thích hợp là khó [62]. Trong khai phá luật kết hợp định lượng, các thuộc tính có thể là định lượng và phân loại.

1.2 Tổng quan về Logic mờ

1.2.1 Tập mờ

Lý thuyết tập mờ được Zadeh đưa ra vào năm 1965 [22] và rất phù hợp để xử lý các giá trị định lượng và biểu diễn ý nghĩa ngôn ngữ. Biểu diễn ngôn ngữ là phổ biến và dễ hiểu hơn đối với con người. Một biến ngôn ngữ là một biến có giá trị của nó là tập các thuật ngữ mờ được biểu diễn bằng ngôn ngữ tự nhiên và được xác định bởi các hàm thành viên [63].

Cho một tập vũ trụ U với các phần tử ký hiệu bởi u , $U = \{x\}$. Một tập mờ $\tilde{A}$ trên U là tập được đặc trưng bởi một hàm $\mu_A(u)$ mà nó liên kết mỗi phần tử $u \in U$ với một số thực trong đoạn [0,1].

$$\tilde{A} = \{(u, \mu_A(u)) \mid u \in U\} \quad (1.4)$$

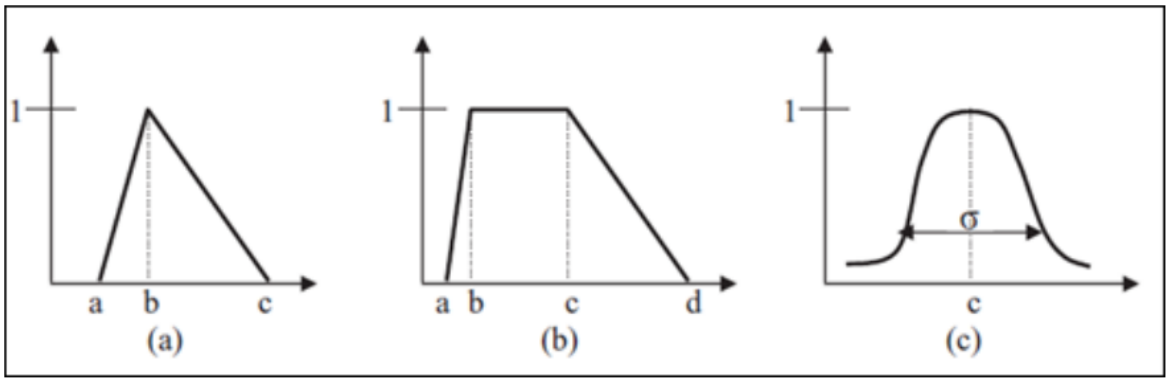
Trong đó $\mu_A(u)$ là một ánh xạ từ U vào [0,1] và được gọi là hàm thành viên của tập mờ $\tilde{A}$.

1.2.2 Hàm thành viên

Hàm thành viên $\mu_A(u)$ định nghĩa cho tập A trên tập vũ trụ U trong khái niệm tập hợp kinh điển chỉ có hai giá trị là 1 nếu $u \in A$ hoặc 0 nếu $u \notin A$. Tuy nhiên trong khái niệm tập mờ thì giá trị hàm thành viên chỉ mức độ thuộc về (membership degree) của phần tử u vào tập mờ A. Khoảng xác định của hàm $\mu_A(u)$ là đoạn [0, 1], trong đó giá trị 0 chỉ mức độ không thuộc về, còn giá trị 1 chỉ mức độ thuộc về hoàn toàn.

$$\mu(A) : U \rightarrow [0,1] \tag{1.5}$$

Kiểu của tập mờ phụ thuộc vào các kiểu hàm thành viên khác nhau. Có nhiều kiểu hàm thành viên khác nhau được đề xuất. Một số kiểu hàm thành viên sử dụng phổ biến trong logic mờ như sau (xem Hình 1.1) [64], [65]:



Hình 1.1: Đồ thị của 3 hàm thành viên phổ biến: (a) tam giác, (b) hình thang, (c) Gauss.

Dạng tam giác (Triangles): Hàm thành viên này được xác định bởi 3 tham số là cận dưới a, cận trên c và giá trị b (ứng với đỉnh tam giác), với $a < b < c$. Hàm thành viên này được gọi là đối xứng nếu nếu giá trị $b - a$ bằng giá trị $c - b$, hay $b = (a + b)/2$. Công thức xác định hàm thành viên tam giác như sau:

$$\text{triangle}(\text{x};\ \text{a},\text{b},\text{c}) = \begin{cases} 0 & x < a \\ (x-a)/(c-a) & a \le x \le c \\ (b-x)/(b-c) & c \le x \le b \\ 0 & x > b \end{cases} \tag{1.6}$$

Dạng hình thang (Trapezoids): Hàm thành viên này được xác định bởi bộ 4 giá trị a, b, c, d, với $a < b < c < d$, theo công thức sau:

$$\text{trapezoid(x; a, b, c, d)} = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \le x < b \\ 1 & b \le x < c \\ (d-x)/(d-c) & c \le x < d \\ 0 & x \ge d \end{cases} \quad (1.7)$$

Dạng Gauss: Hàm thành viên này được xác định bởi 2 tham số, gồm: giá trị c là giá trị trung bình (ứng với giá trị cực đại của hàm thành viên) và σ là độ lệch chuẩn (độ rộng của hàm). Chúng ta có thể điều chỉnh đồ thị hàm thành viên bằng cách thay đổi giá trị tham số σ. Công thức xác định hàm thành viên Gauss như sau:

$$gauss(x; c, \sigma) = exp\left(-\frac{(x-c)^2}{2\sigma^2}\right) \quad (1.8)$$

1.2.3 Biến ngôn ngữ

Biến ngôn ngữ [66] là bộ năm $(X, T(X), U, R, M)$, trong đó X là tên biến, T(X) là tập giá trị ngôn ngữ của biến X, U là không gian tham chiếu của biến cơ sở u, mỗi giá trị ngôn ngữ được xem là một biến mờ trên U kết hợp với biến cơ sở u, R là một quy tắc cú pháp sinh các giá trị ngôn ngữ của $T(X)$, M là quy tắc ngữ nghĩa gán mỗi giá trị ngôn ngữ trong $T(X)$ với một tập mờ trên U.

Ví dụ: Cho X là biến ngôn ngữ có tên TUỔI, biến cơ sở u lấy theo số tuổi của con người có miền xác định là $U = [0,100]$. Tập các giá trị ngôn ngữ $T(TUỔI) = \{rất\ trẻ, trẻ, trung\ niên, già, rấtgià\}$.

1.2.4 Các phép toán logic mờ

Ba phép toán logic mờ cơ bản: phép bù, phép hợp và phép giao thường được sử dụng trong lý thuyết tập mờ, được mô tả dưới đây [22].

Phép bù: Phép toán bù của tập mờ A được ký hiệu là ⌐A. Hàm thành viên của ⌐A có thể được định nghĩa là:

$$\mu_{\neg A}(x) = 1 - \mu_A(x), \quad \forall x \in X \quad (1.9)$$

Phép hợp: Phép hợp của hai tập mờ A và B được ký hiệu là $A \cup B$. Hàm thuộc của $A \cup B$ đối với phép toán chuẩn có thể được định nghĩa như sau:

$$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}, \qquad \forall x \in X \tag{1.10}$$

Phép giao: phép toán giao của hai tập mờ A và B được ký hiệu là $A \cap B$. Hàm thành viên của $A \cap B$ đối với phép toán chuẩn có thể được định nghĩa như sau:

$$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \qquad \forall x \in X \tag{1.11}$$

1.3 Luật kết hợp mờ

Một vấn đề của khai phá luật kết hợp cổ điển là không phải mọi loại dữ liệu đều có thể được sử dụng để khai phá. Các luật chỉ có thể được lấy từ dữ liệu chứa dữ liệu nhị phân, trong đó quan tâm một mục có tồn tại hay không tồn tại trong giao tác. Khi làm việc với một cơ sở dữ liệu định lượng, không có luật kết hợp nào có thể được phát hiện. Để xử lý cơ sở dữ liệu có chứa cả thuộc tính phân loại và thuộc tính định lượng, phương pháp khai phá luật kết hợp định lượng được đề xuất bởi Srikant and Agrawal [29]. Đầu tiên là xác định số lượng phân vùng trên các thuộc tính định lượng, và sau đó chuyển các giá trị định lượng sang các giá trị nhị phân để sử dụng các thuật toán đã có. Các phương pháp khai phá luật kết hợp cổ điển dựa trên logic Boolean để chuyển đổi các thuộc tính số sang thuộc tính boolean bằng cách phân vùng dữ liệu cứng. Vì vậy, số luật sinh ra là thấp. Điều này không hiệu quả trong trường hợp khai phá dữ liệu cỡ lớn. Để giải quyết vấn đề đó, lý thuyết tập mờ được sử dụng trong khai phá luật kết hợp trong thời gian gần đây [31], [67], [68].

1.3.1 Cơ sở dữ liệu giao dịch mờ

Cho $I = \{I_1, I_2, \dots, I_m\}$ là tập n các thuộc tính, i_u là thuộc tính thứ u trong I. $D_Q = \{T_1, T_2, \dots, T_n\}$ là một tập các giao tác với mỗi $T_v \in D_Q$ là một tập con của I chứa các mục có giá trị định lượng và có một định danh duy nhất TID. Một giao tác T được gọi là chứa X nếu $X \subseteq T_q$ trong đó, X là một tập chứa vài mục có trong I. Mỗi thuộc tính I_k có thể kết hợp với tập giá trị mờ được biểu diễn $F_{ik} = \{f_{ik}^1, f_{ik}^2, \dots, f_{ik}^h\}$ với f_{ik}^jlà giá trị mờ thứ j trong F_{ik}. Ví dụ, thuộc tính Lương có thể được biểu diễn như sau: $F_{Lương} = \{Cao,\ Trung\ bình, Thấp\}$. Sử dụng hàm thành

viên liên quan để xác định tập mờ cho các mỗi thuộc tính, cơ sở dữ liệu định lượng D_Q được chuyển thành cơ sở dữ liệu chứa giá trị mờ D_f.

Ví dụ về CSDL mờ được thể hiện trong bảng 1.3

Bảng 1.3: CSDL mờ mẫu

TID	**Các mục**
1	$\left(\frac{0.2}{A.Low}+\frac{0.8}{A.Middle}\right),\left(\frac{0.2}{C.Middle}+\frac{0.8}{C.High}\right),\left(\frac{0.8}{D.Low}+\frac{0.2}{D.Middle}\right),\left(\frac{0.4}{E.Middle}+\frac{0.6}{E.High}\right)$
2	$\left(\frac{0.6}{A.Middle}+\frac{0.4}{A.High}\right),\left(\frac{0.8}{B.Low}+\frac{0.2}{B.Middle}\right),\left(\frac{0.6}{C.Low}+\frac{0.4}{C.Middle}\right)$
3	$\left(\frac{0.6}{B.Low}+\frac{0.4}{B.Middle}\right),\left(\frac{0.4}{C.Middle}+\frac{0.6}{C.High}\right)$
4	$\left(\frac{0.8}{A.Middle}+\frac{0.2}{A.High}\right),\left(\frac{0.4}{C.Middle}+\frac{0.6}{C.High}\right),\left(\frac{0.6}{D.Low}+\frac{0.4}{D.Middle}\right)$
5	$\left(\frac{0.2}{A.Low}+\frac{0.8}{A.Middle}\right),\left(\frac{0.8}{B.Low}+\frac{0.2}{B.Middle}\right),\left(\frac{0.2}{C.Low}+\frac{0.8}{C.Middle}\right)$
6	$\left(\frac{0.6}{A.Low}+\frac{0.4}{A.Middle}\right),\left(\frac{0.2}{C.Middle}+\frac{0.8}{C.High}\right),\left(\frac{0.8}{D.Low}+\frac{0.2}{D.Middle}\right),\left(\frac{0.8}{E.Low}+\frac{0.2}{E.Middle}\right)$

1.3.2 Độ hỗ trợ của tập mục mờ

Một tập thuộc tính mờ trong luật kết hợp mờ là một cặp $\langle X, A\rangle$ với A là tập các tập mờ tương ứng với các thuộc tính trong X và $X \subseteq I$.

Độ hỗ trợ của tập mục $\langle X, A\rangle$ ký hiệu là $fsup(\langle X, A\rangle)$ được xác định bởi công thức sau:

$$fsup(\langle X, A\rangle) = \sum_{t\in T} \mu_{x1}(t) \otimes \mu_{x2}(t) \otimes \ldots \otimes \mu_{xp}(t) \qquad (1.12)$$

Trong đó, $\mu_{xp}(t)$ là giá trị mờ của thuộc tính x_p trong giao tác *t*.

$\otimes$ là toán từ T-norm (T-chuẩn). Trong lý thuyết logic mờ, nó có vai trò giống như phép toán AND trong logic cổ điển. Có nhiều cách lựa chọn phép toán T-norm 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: $a \otimes b = \begin{cases} a\ (nếu\ b = 1) \\ b\ (nếu\ a = 1) \\ 0\ (nếu\ a, b < 1) \end{cases}$

Phép giao: $a \otimes b = 1 - min\left[1, ((1-a)^w + (1-b)^w)^{\frac{1}{w}}\right]$ với $(w > 0)$

Phép lấy min và phép tính đại số là hai phép toán phù hợp nhất vì nó thuận tiện cho việc tính toán và thể hiện được mối liên hệ chặt chẽ giữa các thuộc tính trong các tập phổ biến.

Ví dụ: Độ hỗ trợ mờ của tập mục mờ {A.Low} theo công thức (1.12) là 0.2 + 0.2 + 0.6 = 1.0

Khi chọn phép lấy min cho toán tử T-norm, công thức tính độ hỗ trợ của tập mục $\langle X, A\rangle$ trở thành:

$$fsup(\langle X, A\rangle) = \sum_{t \in T} min\{\mu_{x1}(t),\ \mu_{x2}(t), \dots,\ \mu_{xp}(t)\} \qquad (1.13)$$

Khi chọn phép lấy tích đại số cho toán tử T-norm, công thức tính độ hỗ trợ của tập mục $\langle X, A\rangle$ trở thành:

$$fsup(\langle X, A\rangle) = \sum_{t \in T} \prod_{x_p \in X} \{\mu_{xp}(t)\} \qquad (1.14)$$

Ví dụ: Độ hỗ trợ mờ của tập mục mờ {A.Low} và {C.High} xuất hiện trong cùng một giao dịch sẽ là 0.2 + 0.6 = 0.8.

1.3.3 Tập mục phổ biến mờ

Định nghĩa 1.8: (Tập mục phổ biến mờ): [42]

Một tập mục $\langle X, A\rangle$ được gọi là phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng độ hỗ trợ tối thiểu (*fminsup*) do người dùng định nghĩa $fsup(\langle X, A\rangle) \geq fminsup$.

Khai phá tập mục mờ phổ biến là bài toán trích xuất tất cả các tập mục mờ phổ biến có dạng:

$$FFI_k = \{X \mid fsup(X) \geq \delta \times |D_f|\} \quad (1.15)$$

Giả sử độ hỗ trợ tối thiểu trong ví dụ ở bảng 1.3 là 30% thì các mục mờ phổ biến thu được như trong bảng 1.4

Bảng 1.4: Các tập mở phổ biến được khai phá từ bảng 1.3

Tập mục mờ 1-item	**Độ hỗ trợ**
{A. Middle}	3.4
{C.High}	2.8
{C.Middle}	2.4
{B.Low}	2.2
{D.Low}	2.2
Tập mục mờ 2-items	
{A.Middle, C.High}	1.8
{A.Middle, C.Middle}	2.0
{A.Middle, D.Low}	1.8
{C.High, D.Low}	2.2
Tập mục mờ 3-item	
{A.Middle, C.High, D.Low}	1.8

1.3.4 Luật kết hợp mờ

Sau khi có được các khoảng mờ và các hàm thành viên tương ứng của chúng cho mỗi tập mờ của thuộc tính định lượng được, một cơ sở dữ liệu D_F được biến đổi (bằng cách mờ hóa) được tạo ra từ cơ sở dữ liệu gốc. Cho cơ sở dữ liệu mờ $D_F = \{T_1, T_2, \dots, T_n\}$ với các thuộc tính $i_j \in I$ và các tập mờ F_{ij} tương ứng với các thuộc tính trong I. Một luật kết hợp mờ có dạng như sau:

$$If\ X = \{x_1, x_2 \dots, x_p\}\ is\ A = \{a_1, a_2 \dots, a_p\}\ then\ Y = \{y_1, y_2 \dots, y_q\}\ is\ B = \{b_1, b_2 \dots, b_q\}$$

Trong đó: $a_i \in F(x_i)$, $i = 1, \dots, p$ và $b_j \in F(y_j)$, $j = 1, \dots, q$. X và Y là các tập mục con có trật tự của I và phân biệt, không có chung thuộc tính nào. Khi đó, X is A được gọi là tiền đề của luật và Y is B được gọi là hệ quả của luật.

Một ví dụ về luật kết hợp có dạng: *Nếu Tuổi is Trẻ THEN Thu nhập is Thấp*

Định nghĩa 1.9: (Độ hỗ trợ của một luật kết hợp mờ)

Độ hỗ trợ của một luật mờ $X\ is\ A\ \Rightarrow Y\ is\ B$ được xác định theo công thức sau:

$$fsup(\langle X\ is\ A\ \Longrightarrow Y\ is\ B\rangle) = fsup(\langle X \cup Y, A \cup B\rangle) \quad (1.16)$$

Định nghĩa 1.10: (Độ tin cậy của một luật kết hợp mờ)

Độ tin cậy của một luật mờ $X\ is\ A\ \Rightarrow Y\ is\ B$ được xác định theo công thức sau:

$$fconf(\langle X\ is\ A\ \Longrightarrow Y\ is\ B\rangle) = \frac{fsup(\langle X\ is\ A\ \Longrightarrow Y\ is\ B\rangle)}{fsup(\langle X, A\rangle)} \quad (1.17)$$

Định nghĩa 1.11: (Luật mờ phổ biến).

Một luật được gọi là phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng *fminsup*, có nghĩa là $fsup(\langle X\ is\ A\ \Longrightarrow Y\ is\ B\rangle) \geq fminsup$.

Định nghĩa 1.12. (Luật mờ tin cậy). 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) được định nghĩa bởi người dùng, nghĩa là $fconf(\langle X\ is\ A\ \Longrightarrow Y\ is\ B\rangle) \geq fminconf$.

1.4 Các nghiên cứu liên quan

1.4.1 Các nghiên cứu tiếp cận dựa trên Apriori

Chan và Au lần đầu tiên trình bày thuật toán F-APACS [69] để khai phá các luật kết hợp mờ. Các giá trị của các thuộc tính định lượng đầu tiên được chuyển đổi thành biểu diễn của các thuật ngữ ngôn ngữ với các giá trị liên thuộc của chúng theo các hàm liên thuộc được xác định trước. Trong thuật toán F-APACS, các ngưỡng do người dùng chỉ định là không bắt buộc dựa trên phân tích thống kê được thiết kế. Ngoài ra, cả luật kết hợp mờ dương và âm đều có thể được phát hiện thông qua thuật toán F-APACS.

Kuok và cộng sự [31] sau đó đề xuất một cách tiếp cận để xử lý các thuộc tính định lượng để khai phá các luật kết hợp mờ. Một hệ số quan trọng được thiết kế để khai phá tất cả các tập phổ biến (lớn) từ cơ sở dữ liệu định lượng. Nó phản ánh không

chỉ tần suất xuất hiện của các mục trong cơ sở dữ liệu mà còn cả mức độ hỗ trợ của các tập mục. Một hệ số chắc chắn cũng được thiết kế để tạo ra các luật có thể có từ các tập phổ biến.

Đồng thời, Hong và cộng sự thông qua lý thuyết tập mờ đã trình bày một thuật toán FTDA [32] để xử lý các cơ sở dữ liệu định lượng. Thuật toán này dựa trên thuật toán Apriori để khai phá tập phổ biến mờ theo mức độ một cách thông minh để tạo ra các luật kết hợp mờ. Thuật toán FTDA được đề xuất đầu tiên chuyển đổi các giá trị định lượng của các mục thành biểu diễn thuật ngữ ngôn ngữ dựa trên các hàm thuộc được xác định trước. Các lực lượng của các thuật ngữ ngôn ngữ được chuyển đổi sau đó được tính toán. Chỉ một thuật ngữ ngôn ngữ với lực lượng tối đa của mỗi thuộc tính được sử dụng cho quá trình khai phá sau này. Quá trình này có thể giữ nguyên số lượng các mục như số lượng của các thuộc tính ban đầu, do đó giảm chi phí tính toán của các tổ hợp lớn. Sau đó, các tập mục phổ biến mờ còn lại có thể được sử dụng để tạo ra các luật kết hợp mờ. Thuật toán FTDA được mô tả như sau:

Thuật toán 1.1: FTDA

Input: D_Q: Cơ sở dữ liệu định lượng; minsup; minsup: ngưỡng hỗ trợ tối thiểu; minconf: ngưỡng độ tin cậy tối thiểu; MFs: hàm thành viên

Output: Các luật kết hợp mờ

Begin

1: for each transaction t_i in D_Q do

2: for each item (attribute) A_j do

3: Chuyển thuộc tính định lượng q_{ij} sang tập mờ bằng công thức $\left(f_{ij1}/A_j.R_1 + f_{ij2}/A_j.R_2 + \cdots f_{ijn}/A_j.R_n\right)$

4: End for

5: End for

6: Tính $count\left(A_j.R_k\right) = sum\{f_{ijk}\}$

7: $MAXcount\left(A_j.R_k\right) = max\{count\left(A_j.R_k\right)\}$

8: $L_1 \leftarrow \{A_j.R_k | MAXcount\left(A_j.R_k\right) \geq minsup \times |D_Q|\}$

9: r=2;

10: While $L_{r-1} \neq null$ do

11: $C_r \leftarrow \{a \cup b \mid a, b \in L_{r-1}, a \notin b\}$

12: for each transaction t_i in D_Q do

13: $C_{ti} \leftarrow \{z \mid z \in C_r \wedge z \subseteq t_i\}$

14 For each $z \in C_{ti}$ do

15: Tính $count(t_i.z) = \{min(f_{ijx}, f_{ijy}) | x, y \in z, x \notin y\}$

16: End for

17: End for

18: Tính $count(z) = sum\{count(t_i.z)\}$;

19: $L_r \leftarrow \{z \mid count(z) \geq minsup \times |D_Q|\}$

20: $r = r + 1$;

21: End while

22: $C_{FARs} \leftarrow \{L_1 \wedge L_2 \wedge \ldots \wedge L_r \rightarrow L_q \mid q = 1 \; to \; r\}$

23: For each $w \in C_{FARs}$ do

24: Tính $conf(w)$

25: $FARs \leftarrow \{w \mid conf(w) \geq minconf \times |D_Q|\}$

26: End for

27: Return *FARs*

End.

1.4.2 Các nghiên cứu mở rộng từ Apriori

Một số thuật toán biến thể đã được trình bày để khai thác các luật kết hợp mờ. Gyenesei đã đề xuất một quy trình chuẩn hóa mờ bổ sung để khai phá luật kết hợp mờ từ cơ sở dữ liệu định lượng [70]**.** Trong cách tiếp cận của tác giả, bên cạnh độ tin cậy và độ hỗ trợ mờ, một hệ số tương quan mờ mới được định nghĩa như một thước đo mới để khai phá các luật kết hợp mờ. Hong [71] sau đó tăng cường thuật toán FTDA để thiết kế một cách tiếp cận AprioriTid mới để khai phá hiệu quả các luật kết hợp mờ. Yue mở rộng cách tiếp cận FTDA để khai phá luật kết hợp mờ với ràng buộc trọng số [72]**.** Theo cách tiếp cận của họ, mỗi mục được gán một giá trị trọng số trong phạm vi [0, 1] để thể hiện tầm quan trọng của nó. Phương pháp ánh xạ tự tổ chức Kohonen cũng được áp dụng để lấy các tập mờ cho các thuộc tính số. Chen và Wei đã phát triển một khung tổng quát để khai phá các luật kết hợp mờ dựa trên cấu trúc phân loại mờ [73]**.** Hong sau đó thiết kế một quy trình khai phá để trích xuất các luật kết hợp mờ dựa trên độ hỗ trợ ngôn ngữ tối thiểu và ngưỡng tin cậy tối thiểu [74]**.** Sau đó tác giả cũng đã phát triển một thuật toán khai phá mờ nhiều mức để khai phá

các luật kết hợp mờ bằng cách tích hợp các khái niệm tập mờ và phân loại nhiều mức [28].

1.4.3 Các phương pháp nghiên cứu dựa trên cây

Để khai phá luật kết hợp mờ, thuật toán FTDA được đề cập ở trên áp dụng một cơ chế giống như Apriori để khai phá các tập mục phổ biến mờ để tạo ra các luật kết hợp mờ. Cách tiếp cận này yêu cầu quét cơ sở dữ liệu nhiều lần để khai phá các tập phổ biến mờ với việc tính toán tốn nhiều thời gian. Để giải quyết vấn đề này, Papadimitriou đề xuất thuật toán cây mẫu thường xuyên mờ (FFPT- Frequent Fuzzy Pattern Tree) [75]. Lin sau đó trình bày một framework để khai phá mờ khác để tìm ra các mục phổ biến mờ dựa trên cấu trúc cây. Do quá trình xử lý thường phức tạp bởi các toán tử mờ, nên một số thông tin bổ sung được lưu trữ trong các nút của cây để thực hiện chính xác tác vụ. Ba thuật toán là cây phổ biến mờ FP (FFP)-tree [38], cây phổ biến mờ nén (CFFP)-tree [40] và cây mẫu phổ biến mờ giới hạn trên (UBFFP)-tree [41] đã được phát triển để khai phá các tập mục phổ biến mờ từ cơ sở dữ liệu định lượng. Chúng khác nhau chủ yếu ở cấu tạo cây.

1.4.3.1 Thuật toán FP-Tree mờ

Đối với thuật toán cây FFP [38], nó sử dụng cách tiếp cận tương tự với thuật toán FTDA để chuyển đổi thuộc tính số lượng trong cơ sở dữ liệu gốc thành biểu diễn của thuật ngữ ngôn ngữ và thu được tập mục phổ biến mờ 1-item. Sau đó, tập mục phổ biến mờ 1-item được sử dụng để xây dựng chỉ mục Header_Table, chỉ mục này có chức năng tương tự như Header_Table của cấu trúc cây FP. Các giao dịch được chuyển đổi trong cơ sở dữ liệu sau đó được tinh chỉnh lại để chỉ giữ lại các tập mục phổ biến mờ. Chiến lược sắp xếp cục bộ được áp dụng để sắp xếp các mục phổ biến mờ còn lại theo các giá trị thành viên được chuyển đổi của chúng trong mỗi giao dịch. Sau đó, giao dịch được xử lý từng dòng một để xây dựng cây FFP và mỗi nút trong cây giữ giá trị thành viên của các mục phổ biến mờ 1-item được xử lý trong mỗi giao dịch. Một cách tiếp cận tương tự như tăng trưởng FP được sử dụng để thu được các tập phổ biến mờ từ cấu trúc cây FFP được xây dựng.

Thuật toán FFP-tree được mô tả như trong thuật toán 1.2.

Thuật toán 1.2 : FFP-Tree

Input: D_f: Cơ sở dữ liệu mờ; minsup: ngưỡng hỗ trợ tối thiểu

Output: Cấu trúc cây FFP-tree

Begin

1: Tìm L_1 từ CSDL chuyển đổi

2: $Htable \leftarrow L_1$

3: Tạo $root \leftarrow null$

4: $root.next \leftarrow ptr$

5: For each converted t_i in D_f do

6: For each $z \in L_1 \wedge z \subseteq t_i$ do

7: Sắp xếp z theo thứ tự giảm dần $f_i(z)$

8: End for

9: For each $z_k \subseteq t_i \wedge 1 \leq k \leq |t_i|$ do

10: If $ptr == null \vee \forall ptr.name \neq z_k$

11: Tạo node n

12: $(n.name, n.count) \leftarrow (z_k, f_i(z_k))$

13: $n.next \leftarrow ptr$

14 $n \leftarrow Htable.hyper$

15: $n.hyper \leftarrow lptr$

16: Else

17: $n.count \leftarrow n.count + f_i(z_k)$

18: $n.next \leftarrow ptr$

19: Tìm lptr từ Htable

20: $n \leftarrow lptr$

21: $n.hyper \leftarrow lptr$

22: End if

23: **End for**

24: End for

End.

Sau khi cây FFP-tree được xây dựng, các mục mờ phổ biến sẽ được khai phá bằng cách tiếp cận giống như FP-Growth. Thuật toán được thể hiện dưới đây [38].

Thuật toán 1.3: FFP-Growth

Input: FFP-tree; minsup: ngưỡng hỗ trợ tối thiểu;

Output: Tập mục mờ phổ biến

Method: Call FFP-Growth (FFP-tree, null) như sau:

Procedure FFP − Growth (FFP − tree, α)

1: If p là đường đi duy nhất, $p \in FFP - tree$ then

2: **For** each tổ hợp q of nodes in p **do**

3: $z \in (q \cup \alpha)$

4: $z.count \leftarrow min\{n.count \mid n \in q\}$

5: **End for**

6: **Else**

7: **For** each i_j in Head_Table **do**

8: $q \leftarrow (i_j \cup \alpha)$

9: $q.count \leftarrow i_j.count$

10: Xây dựng FFP-tree có điều kiện của q là $Tree_q$

11: **If** $Tree_q \neq null$ **then**

12: Call FFP − Growth ($Tree_q$, q)

13: **End if**

14 **End for**

15: **End if**

End.

Mặc dù thuật toán cây FFP có thể được sử dụng để khai phá hiệu quả các tập phổ biến mờ từ cây FFP đã xây dựng, nhưng cần có nhiều nút hơn do thứ tự được sắp xếp để xây dựng cây FFP dựa trên giá trị thành viên của các mục trong mỗi giao dịch. Do đó, hai giao dịch có cùng thuật ngữ ngôn ngữ có thể có thứ tự khác nhau, do đó tạo ra các đường dẫn khác nhau trong cấu trúc cây FFP. Quá trình này có thể tạo ra nhiều nút bổ sung.

1.4.3.2 Thuật toán CFFP-tree và UBFFP-tree

Để giải quyết vấn đề trên của thuật toán FFP-tree, Lin sau đó trình bày thuật toán nén cây FFP (CFFP – Compact Fuzzy Frequent Pattern) [76] để giảm số nút cây. Chiến lược sắp xếp của thuật toán cây CFFP khác với thuật toán cây FFP. Nó áp dụng chiến lược sắp xếp toàn cục để sắp xếp các tập mục mờ phổ biến 1-item còn lại theo

thứ tự giảm dần về tần suất xuất hiện của chúng trong tất cả các giao dịch. Một mảng bổ sung được gắn vào mỗi nút và được cập nhật trong cấu trúc cây CFFP để giữ các giá trị thành viên của nút hiện đang được xử lý với bất kỳ nút tiền tố nào của nó bằng thao tác giao nhau. Dựa trên mảng đính kèm của mỗi nút, thuật toán CFFP được thực hiện để khai phá các tập mục phổ biến mờ hoàn chỉnh thông qua một phép toán giao đơn giản.

Mặc dù thuật toán cây CFFP sử dụng một mảng được đính kèm trong mỗi nút để giảm số lượng nút cây, nhưng nó tốn kém bộ nhớ để duy trì mảng. Vì mảng đính kèm trong mỗi nút giữ các giá trị thành viên của nút hiện được xử lý với bất kỳ mục tiền tố nào của nó trong đường dẫn, nên độ phức tạp về không gian của mỗi nút sẽ cao nếu kích thước của các giao dịch được xử lý lớn. Lin sau đó đề xuất thuật toán cây phổ biến mờ giới hạn trên (UBFFP – Upper Bound Fuzzy Frequent Pattern) để đánh giá quá cao các giá trị thành viên giới hạn trên của các tập mục phổ biến mờ để giải bài toán vượt quá cấu trúc cây CFFP. Thuật toán xây dựng cây UBFFP sử dụng chiến lược sắp xếp toàn cục giống như thuật toán cây CFFP để xây dựng cây. Sau đó, mỗi mục trong các giao dịch được làm mờ bằng cách chỉ giữ lại thuật ngữ ngôn ngữ có giá trị thành viên cao trong các quy trình sau này, đây là quy trình tương tự như cây FFP. Các giao dịch được truyền sau đó được xử lý theo từng bộ từ giao dịch đầu tiên đến giao dịch cuối cùng để xây dựng cây UFFP. Mỗi nút trong cây giữ một tập phổ biến mờ 1-item với số lượng mờ tích lũy của nó, giống như cấu trúc cây FFP nhưng khác với cấu trúc cây CFFP. Sau khi cây UBFFP được xây dựng, một thuật toán tăng trưởng UBFFP được sử dụng để tìm đệ quy các tập phổ biến mờ từ cấu trúc cây UBFFP. Sau quá trình xây dựng, thuật toán tăng trưởng UBFFP sau đó được thực thi để khai thác các tập phổ biến mờ.

1.4.3.3 Thuật toán MFFP (Multiple Fuzzy Frequent Pattern)

Thuật toán xây dựng cây MFFP (Multiple Fuzzy Frequent Pattern) [42] để giữ nhiều vùng phổ biến mờ của một thuật ngữ (mục). Đầu tiên, thuật toán chuyển đổi các giá trị định lượng trong các giao dịch thành các thuật ngữ ngôn ngữ dựa trên các hàm thành viên được xác định trước. Các vùng mờ được chuyển đổi với lực lượng tương ứng của chúng sau đó được kiểm tra theo độ hỗ trợ tối thiểu được xác định trước cho quá trình khai phá tiếp theo. Mỗi thuật ngữ có thể có nhiều hơn một vùng mờ, điều này làm cho quá trình khai phá trở nên phức tạp hơn nhưng lại đầy đủ để

tạo ra các luật kết hợp mờ. Các tập phổ biến mờ, được biểu diễn bằng các thuật ngữ ngôn ngữ, sau đó được dẫn xuất từ cây MFFP.

Thuật toán 1.4: MFFP-tree

Input: D_Q: Cơ sở dữ liệu định lượng; minsup: ngưỡng hỗ trợ tối thiểu

Output: Cấu trúc cây MFFP-tree

Begin

1: for each giao dịch t_i in D_Q do

2: for each mục (attribute) A_j do

3: Chuyển đổi giá trị định lượng q_{ij} by MFs as $\left(f_{ij1}/A_j.R_1 + f_{ij2}/A_j.R_2 + \cdots f_{ijn}/A_j.R_n\right)$

4: End for

5: End for

6: Tính $count\left(R_{jl}\right) = sum\{f_{ijl}\}$

7: $L_1 \leftarrow \{R_{jl} |\ count\left(R_{jl}\right) \geq minsup\ \times |D_Q|\}$

8: Tạo Header_Table với L_1 được sắp xếp giảm dần theo count

9: For each giao dịch t_i in D_f do

10: Loại bỏ các mục mờ không có trong L_1

11: Sắp xếp các vùng phổ biến mờ còn lại theo thứ tự giảm dần của các giá trị thành viên.

12: End for

13: Tạo $root \leftarrow null$

14 **For** each giao dịch t_i trong CSDL được chuyển đổi **do**

15: Thêm t_i vào cây MFFP tree

16: **If** R_{jl} in t_i ở nhánh tương ứng của cây MFFP tree trong giao dịch **then**

17: Thiết lập $node.count \leftarrow node.count + f_{ijl}\left(R_{jl}\right)$

18: **else**

19: Chèn thêm a node của R_{jl} vào bên cuối nhánh tương ứng của cây MFFP

20: Thiết lập $(node.name, node.count) \leftarrow \left(f_{ijl}\left(R_{jl}\right)\right)$

21: Chèn a link từ the node của R_{jl} trong nhánh cuối vào node hiện tại

22: **If** không có nhánh nào với node của R_{jl} **then**

23: Chèn một link từ node vào của R_{jl} trong the Header_Table đến node đã thêm.

24: **End if**

25: **End if**

26: **End for**

End.

Các mục phổ biến mờ (vùng mờ) trong Header_Table được xử lý lần lượt từ dưới lên trên để tạo thành tập phổ biến mờ. Các nút chứa vùng mờ hiện đang được xử lý được tìm thấy đầu tiên từ cây MFFP. Các tập mục phổ biến mờ có thể được suy ra một cách hiệu quả từ các nút bằng cách sử dụng phép toán giao nhau trong các tập mờ, đây là phép toán min. Thuật toán khai phá được mô tả như trong thuật toán 1.5.

Thuật toán 1.5: MFFP-Growth

Input: MFFP-tree**;** minsup: ngưỡng hỗ trợ tối thiểu; Header-table

Output: The fuzzy frequent itemsets

1: Xử lý các mục phổ biến mờ trong Header_Table lần lượt từ dưới lên trên bằng các bước sau. Đặt vùng mờ hiện đang được xử lý là R_{jl} .

2: For each i_j in Head_Table do

3: Tìm tất cả các nút có vùng mờ R_{jl} trong cây MFFP thông qua các link.

4: Trích xuất các mục cha trong đường dẫn của R_{jl} để tạo thành các mẫu mờ có điều kiện.

5: Thực hiện tổ hợp để lấy ra các mục mờ có liên quan đến R_{jl}

Tính tổng các giá trị mờ của các tập mục mờ dẫn xuất giống nhau.

6: End for

7: If tổng các giá trị mờ của các tập mục mờ dẫn xuất lớn hơn minsup

8: Thêm các tập mục mờ dẫn xuất vào tập mục mờ phổ biến

9: End if

End.

1.5 Xác định vấn đề nghiên cứu

Trong các phương pháp khai phá dữ liệu mờ dựa trên Apriori, các giá trị định lượng được chuyển đổi thành các tập mờ theo các hàm thành viên được xác định

trước. Sau đó, tập phổ biến mờ và luật kết hợp mờ có thể được tạo ra dựa trên quá trình thực hiện Apriori. Do thời gian thực hiện của các phương pháp dựa trên Apriori tốn nhiều thời gian nên các phương pháp khai thác dữ liệu mờ dựa trên cây được mô tả để tăng tốc quá trình khai thác. Về cơ bản, các phương pháp này được sửa đổi từ cây FP để xử lý các tập mục mờ. việc khai phá các tập mục mờ phổ biến được thực hiện hoàn toàn trên cây, điều này chiếm nhiều không gian bộ nhớ. Luận án đã đề xuất thuật toán khai phá tập mục phổ biến mờ dựa trên cấu trúc Nodelist nhằm giải quyết vấn đề về không gian bộ nhớ trong công trình [CT1], [CT2], [CT5].

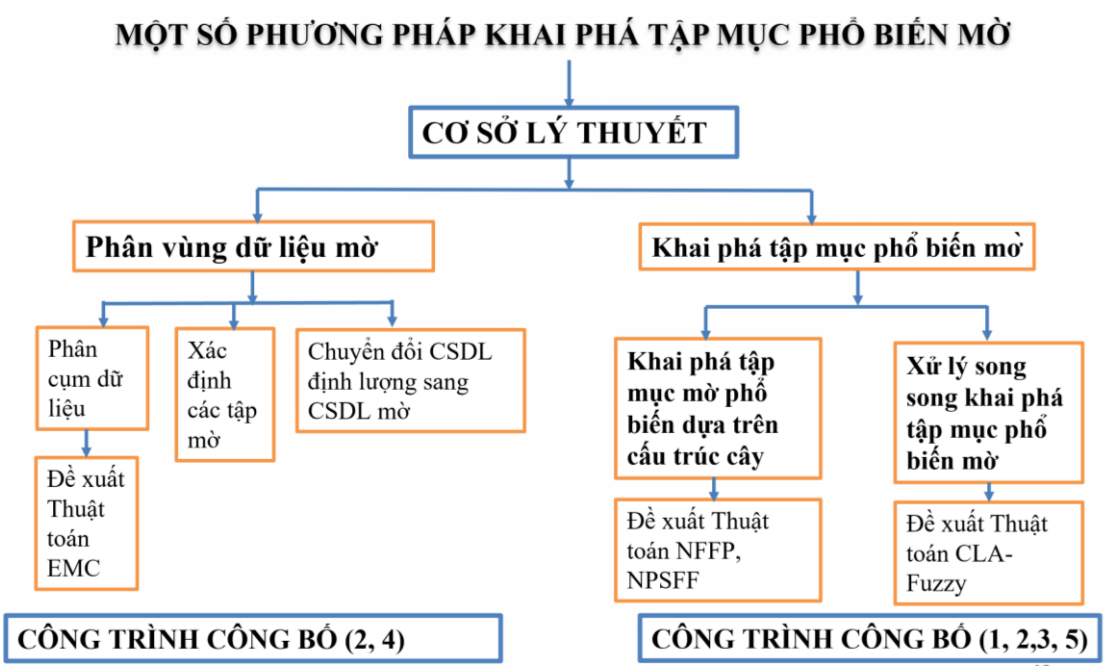
Hơn nữa, các hàm thành viên có thể được đưa ra bởi các chuyên gia. Tuy nhiên, các ý kiến chuyên gia có thể không phải lúc nào cũng có sẵn. Để giải quyết vấn đề này, luận án đã đề xuất phương pháp xác định các tập mờ cho mỗi thuộc tính định lượng trong cơ sở dữ liệu bằng kỹ thuật phân cụm EMC trong công trình [CT2], [CT4].

Cùng với sự gia tăng kích thước của dữ liệu, khai phá dữ liệu trong cơ sở dữ liệu lớn đã trở thành một vấn đề quan trọng. Bên cạnh việc áp dụng điện toán đám mây hoặc các kiến trúc song song và phân phối khác để tăng tốc quá trình khai phá mờ cũng đáng để nghiên cứu. Luận án đề xuất phương pháp xử lý song song quá trình khai phá tập mục phổ biến sử dụng hướng tiếp cận automata di động học [CT3].

1.6 Kết luận chương 1

Trong chương 1, luận án trình bày tóm tắt tổng quan các vấn đề liên quan đến luật kết hợp, logic mờ và luật kết hợp mờ, các phương pháp khai phá dữ liệu mờ khác nhau, bao gồm khai phá dữ liệu mờ dựa trên Apriori, khai phá dữ liệu mờ dựa trên cây rồi từ đó xác định các vấn đề nghiên cứu của luận án.

Hình là sơ đồ thể hiện các nội dung liên quan đến các nghiên cứu của luận án.



Hình 1.2: Các vấn đề liên quan đến nghiên cứu của luận án

Luận án này tập trung trình bày việc giải quyết các bài toán thuộc các nghiên cứu [CT1], [CT2], [CT[3], [CT4], CT5]. Cụ thể, luận án sẽ tập trung nghiên cứu đề xuất và giải pháp giải quyết triệt để 3 vấn đề sau đây:

- Đề xuất phương pháp xác định các tập mờ cho mỗi thuộc tính định lượng trong cơ sở dữ liệu thông qua kỹ thuật phân cụm.
- Đề xuất phương pháp khai phá tập mục phổ biến mờ trong cơ sở dữ liệu định lượng sử dụng cấu trúc dữ liệu Node-list.
- Đề xuất một phương pháp xử lý song song để khai phá các tập phổ biến mờ sử dụng phương pháp tiếp cận automata di động học (Cellular learning automata).

Nội dung hai chương còn lại trong luận án sẽ trình bày giải pháp tương ứng cho ba vấn đề nghiên cứu trên.

Chương 2 KHAI PHÁ TẬP MỤC PHỔ BIẾN MỜ DỰA TRÊN CẤU TRÚC CÂY

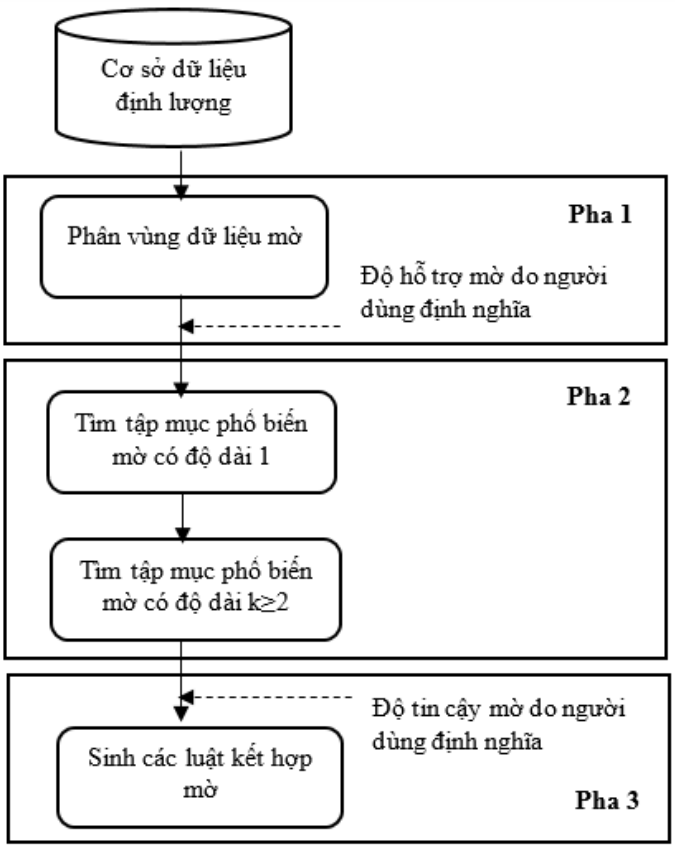
Ở chương này, luận án trình bày vấn đề mờ hóa giá trị định lượng của các mục nhằm giải quyết vấn đề "sắc nét" giữa các khoảng mờ bằng phương pháp phân cụm EMC và xác định các khoảng mờ, kết quả của hai thuật toán này được sử dụng cho bước tiền xử lý dữ liệu sau đó áp dụng các hàm thành viên để chuyển đổi giá trị định lượng sang giá trị mờ. Thứ hai, luận án trình bày hai phương pháp đề xuất khai phá tập mục mờ phổ biến nhằm giải quyết vấn đề không gian bộ nhớ bằng cách sử dụng cấu trúc Nodelist dựa trên cây tiền tố hậu tố (FPPC – Fuzzy Pre-order, Post-order Code) và cây FPOSC (Fuzzy Pre-order Size Code). Kết quả của bước khai phá tập mục mờ phổ biến là cơ sở chính được sử dụng để thực hiện tìm các luật kết hợp mờ.

2.1 Phát biểu bài toán khai phá luật kết hợp mờ

Cho một cơ sở dữ liệu định lượng $D_Q = \{T_1, T_2, \dots, T_n\}$ và tập các mục $I = \{I_1, I_2, \dots, I_m\}$. Cho một tập mờ $A_j = \{A_{j1}, A_{j2}, \dots, A_{jh}\}$ trong đó A_{jk} được xác định là phần tử thứ kth trong tập mờ A_j của mục I_j và $f_{j,k}^{(i)}$ là giá trị mờ (được xác định bởi hàm thành viên) của A_{jk} trong giao tác T_i.

Khai phá luật kết hợp mờ là vấn đề tìm ra tất cả các luật có dạng $A \to B$ thỏa mãn $fsp(A \to B) \geq minsup$ và $fcf(A \to B) \geq minconf$, với minsup và minconf lần lượt là độ hỗ trợ và độ tin cậy tối thiểu do người dùng định nghĩa. Thuật toán khai phá luật kết hợp mờ được thực hiện qua 3 pha chính.

- Pha 1: chuyển đổi cơ sở dữ liệu định lượng sang cơ sở dữ liệu mờ
- Pha 2: Khai phá các tập mục phổ biến mờ có độ hỗ trợ mờ lớn hơn ngưỡng hỗ trợ tối thiểu $FFI_k = \{A | \text{fsup}\ (A) \geq \delta\}$.
- Pha 3: Khai phá tất cả luật kết hợp mờ có độ tin cậy lớn hơn ngưỡng tin cậy tối thiểu từ các mục mờ phổ biến được tìm thấy trong pha 2.



Hình 2.1: Quy trình khai phá luật kết hợp mờ

Các thuật toán đề xuất tập trung vào pha 2, tìm các tập mục phổ biến mờ.

2.2 Thuật toán phân cụm dữ liệu và xác định các khoảng mờ

2.2.1 Các khái niệm cơ bản

2.2.1.1 Phân cụm dữ liệu

Trong khai phá dữ liệu phương pháp tối đa hóa kì vọng Expectation Maximization (EM) [77]–[79] là thuật toán gom nhóm dữ liệu được dùng trong tác vụ khám phá tri thức. Trong thống kê, thuật toán EM lặp và tối ưu hóa khả năng nhìn thấy dữ liệu quan sát thông qua việc ước lượng tham số cho mô hình thống kê cho các biến không quan sát được. Tuy nhiên, Thuật toán EM có những điểm hạn chế như sau: Thứ nhất, EM chạy nhanh ở các vòng lặp ban đầu nhưng chậm hơn ở các vòng lặp sau. Thứ hai, EM không phải lúc nào cũng tìm được tham số tối ưu cho toàn cục, thay vào đó là tối ưu cục bộ.

Từ các nhược điểm của thuật toán EM, luận án đề xuất thuật toán gom cụm cải tiến Expectation Maximization Coefficient (EMC) [CT.4]. Mục tiêu của thuật

toán này là chia dữ liệu thành các cụm có ý nghĩa. Về cơ bản phân nhóm một tập dữ liệu không được gắn nhãn $X = \{x(1), \dots, x(m)\}$ là sự phân chia X thành các nhóm con C $\{1 < C < m\}$ sao cho mỗi nhóm biểu thị một cấu trúc con tự nhiên trong X. Thuật toán EMC, tối ưu hóa phân cụm linh hoạt bằng thuật toán cải tiến cực đại hóa kỳ vọng dựa vào hệ số biến thiên Expectation Maximization Coefficient (EMC). Thuật toán EMC được cải tiến dựa trên thuật toán Expectation Maximization (EM) [77]–[79] bằng cách gia tăng khả năng gom cụm mềm trong qua trình tính toán khoảng cách dựa vào hệ số biến thiên Coefficient (C), đồng thời giảm tối ưu hóa cục bộ và tăng tối ưu hóa toàn cục. Trong bước (C) này, sử dụng hệ số biến thiên để tăng độ mềm trong quá trình phân cụm. Cụ thể hơn, để phân vùng các cụm cũng như tính toán mật độ phân bố giữa các phần tử trong mỗi cụm dựa trên (hệ số biến thiên và tỷ lệ khoảng cách giữa các phần tử trong một cụm). Ngoài ra, thuật toán EMC giảm tối ưu hóa cục bộ và tăng tối ưu hóa toàn cục. Để đánh giá tính hiệu quả của thuật toán phân cụm EMC luận án sử dụng đại lượng (Likelihood) ước lượng hóa tham số đó là mean và variance từ mẫu dữ liệu cho trước.

Định nghĩa 2.1: Hệ số biến thiên C_v được định nghĩa là tỉ số của độ lệch chuẩn σ so với kỳ vọng $\overline{x}$ của cụm i chứa các phần tử $X_i\{x_{i_1}, x_{i_2}, \dots, x_{i_n}\}$ tức là:

$$C_v(X_i) = \frac{\sigma}{\bar{x}} \times 100 \tag{2.1}$$

Định nghĩa 2.2: Phân bố Gaussian đơn biến [77]–[79]

$$N(X|\bar{x}, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X-\bar{x})^2}{2\sigma^2}} \tag{2.2}$$

Định nghĩa 2.3: Phân bố Gaussian đa biến [77]–[79]

$$N(X|\bar{x}, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} exp\left\{-\frac{1}{2}(X-\bar{x})^T \sum^{-1}(X-\bar{x})\right\} \tag{2.3}$$

Phương pháp ước lượng hóa tham số (Maximum Likelihood). Tính log cho phân phối Gaussian. [77]–[79]

$$ln\, p(X|\bar{x}, \Sigma) = -\frac{1}{2} ln(2\pi) - \frac{1}{2} ln|\Sigma| - \frac{1}{2}(X - \bar{x})^T \sum^{-1} (X - \bar{x}) \quad (2.4)$$

Lấy đạo hàm và gắn bằng 0

$$\frac{\delta\, ln\, p(X|\bar{x}, \Sigma)}{\delta \bar{x}} = 0, \bar{x}_{ML} = \frac{1}{N} \sum_{n=1}^{N} X_n$$

$$\frac{\delta\, ln\, p(X|\bar{x}, \Sigma)}{\delta\, \Sigma} = 0, \qquad \Sigma_{ML} = \frac{1}{N} \sum_{n=1}^{N} X_n$$

Trong đó N số lượng các mẫu. Phân phối hỗn hợp tuyến tính của Gaussian

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(X|\bar{x}_k, \Sigma_k) \quad (2.5)$$

Trong đó K là số Gaussian và π_k hệ số pha trộn, với trọng số cho mỗi đơn vị Gaussian $0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1$. Xét log likelihood

$$ln\, p(X|\bar{x}, \Sigma, \pi) = \sum^{N} ln\, p\,(X_n) = \sum^{N} ln \left\{ \sum_{n=1}^{N} \pi_n (X_n|\bar{x}_k, \Sigma_k) \right\} \quad (2.6)$$

2.2.1.2 Xác định các khoảng mờ

Khi thao tác dữ liệu trong cơ sở dữ liệu mờ, vấn đề quan trọng nhất là làm thế nào tìm ra một phương pháp xử lý các giá trị mờ để từ đó xây dựng các quan hệ đối sánh giữa chúng. Các giá trị trong cơ sở dữ liệu mờ rất phức tạp, bao gồm các giá trị ngôn ngữ, giá trị số, giá trị khoảng, ... Có nhiều cách tiếp cận khác nhau để xử lý các giá trị mờ được các tác giả trong và ngoài nước quan tâm nghiên cứu trong những năm gần đây, chẳng hạn như: lý thuyết thuyết tập mờ [22], lý thuyết khả năng [80], [81], quan hệ tương tự [82]. Mỗi một cách tiếp cận có những ưu và nhược điểm riêng và cùng chung một mục đích đó là làm thế nào xử lý thật tốt và thật chính xác những giá trị mờ, xem giá trị rõ là trường hợp riêng của giá trị mờ. Các giá trị khoảng hầu như chuyển về dạng các các số mờ theo các dạng như tam giác, hình thang, hình chuông để xử lý.

2.2.2 Bài toán đặt ra

Cho trước cơ sở dữ liệu chứa các giá trị định lượng D_Q.

Bài toán đặt ra: Xác định tập các tập mờ của các thuộc tính định lượng trong DQ cùng các hàm thành viên tương ứng. Chuyển đổi cơ sở dữ liệu định lượng sang cơ sở dữ liệu mờ.

2.2.3 Thuật toán phân cụm dữ liệu EMC

2.2.3.1 Ý tưởng thuật toán

Thuật toán EMC là một kỹ thuật tối ưu hóa lặp lại được vận hành linh hoạt (Thuật toán được cải thiện để tăng tính linh hoạt cho phân cụm đồng thời giảm tối ưu hóa cục bộ và tăng tối ưu hóa toàn cục).

1) **Bước E:** dựa trên các tham số của mô hình, tính toán các xác suất gán nhãn các điểm dữ liệu vào một nhóm.
2) **Bước M**: cập nhật các tham số của mô hình dựa trên các nhóm gom được từ bước **E**.
3) **Bước C**: Cập nhật các tham số của mô hình dựa trên các biến tiềm ẩn tính theo phương pháp khả năng ước lượng cực đại và tỷ lệ tương tự giữa các đối tượng trong một cụm và đánh giá hệ số biến thiên của các phần tử trong cụm.

Thuật toán EMC bắt đầu bằng các tham số cho mô hình dự đoán. Sau đó thực hiện vòng lặp 5 tiến trình được thể hiện trong Thuật toán 2.1.

2.2.3.2 Thuật toán EMC

Thuật toán EMC được mô tả như trong Thuật toán 2.1

Thuật toán 2.1: EMC (Expectation Maximization Coefficient)

Đầu vào: Khởi tạo giá trị của hệ số biến thiên $C_{v_{value}}$

Đầu ra: Số cụm tối ưu

1: Khởi tạo các tham số kỳ vọng $\bar{x}_j$, hiệp phương sai Σ_j, hệ số pha trộn π_j trong đó $\sum_{j=1}^{j} \pi_j = 1$ và $\pi_j \geq 0 \; \forall j$. Hệ số biến thiên $\mathrm{C_{v_{value}}} = 15\ \%$ đây là giá trị khởi tạo từ người dùng để tính toán tỉ lệ biến thiện giữa các phần tử trong một cụm và các cụm.

2: **Bước E**: Dựa trên các tham số của mô hình, tính toán các xác suất gán nhãn các điểm dữ liệu vào một nhóm

$$\gamma_j(X) = \frac{\pi_k \mathcal{N}(X|\bar{x}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(X|\bar{x}_j, \Sigma_j\right)} \quad (2.7)$$

3: **Bước M**: Cập nhật các tham số của mô hình dựa trên các nhóm gom được từ bước E

$$\bar{x}_j = \frac{\sum_{n=1}^{N} \gamma_j(X_n) X_n}{\sum_{n=1}^{N} \gamma_j(X_n)} \quad (2.8)$$

$$\Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(X_n)\left(X_n - \bar{x}_j\right)X_n - \bar{x}_j{}^T}{\sum_{n=1}^{N} \gamma_j(X_n)} \quad (2.9)$$

$$\pi_j = \frac{1}{N}\sum_{n=1}^{N} \gamma_j(X_n) \quad (2.10)$$

4: Đánh giá log likelihood.

$$\ln p(X|\bar{x}, \Sigma, \pi) = \sum_{n-1}^{N} \ln = \sum^{N} \ln\left\{\sum_{k=1}^{K} \pi_k(X_n|\bar{x}_k, \Sigma_k)\right\} \quad (2.11)$$

5: **Bước C**: Cập nhật thông tin về hệ số biến thiên của các cụm và đánh giá khả năng biến động các phần tử cho từng cụm, cụ thể ta đánh giá hệ số biến thiên của cụm thứ i với C_{v_i} có thảo mãn giá trị biến thiên $C_{v_{value}}$ đã cho hay không.

$$C_{v_i} = \frac{\sum_{n=1}^{N} \gamma_j(X_n) X_n}{\frac{1}{n}\sum_{k=1}^{n} x_k} \quad (2.12)$$

$$C_{v_i} \leq C_{v_{value}} \quad (2.13)$$

6: Nếu không hội tụ và thảo mãn giá trị biến thiên $C_{v_{value}}$ đã cho, quay trở lại bước 2. Nếu likelihood không có nhiều thay đổi thuật toán kết thúc.

Ví dụ: Cho một cơ sở dữ liệu định lượng với thuộc tính định lượng {Số lượng} được biểu diễn trong bảng 2.1 như sau:

Bảng 2.1: Bảng dữ liệu về mặt hàng và số lượng

TID	Tên hàng	Số lượng	TID	Tên hàng	Số lượng
1	Bánh	55	31	Sữa	90
2	Kẹo	74	32	Kẹo	54
3	Bánh	25	33	Sữa	82
4	Bánh	78	34	Bánh	44
5	Bánh	23	35	Bánh	84

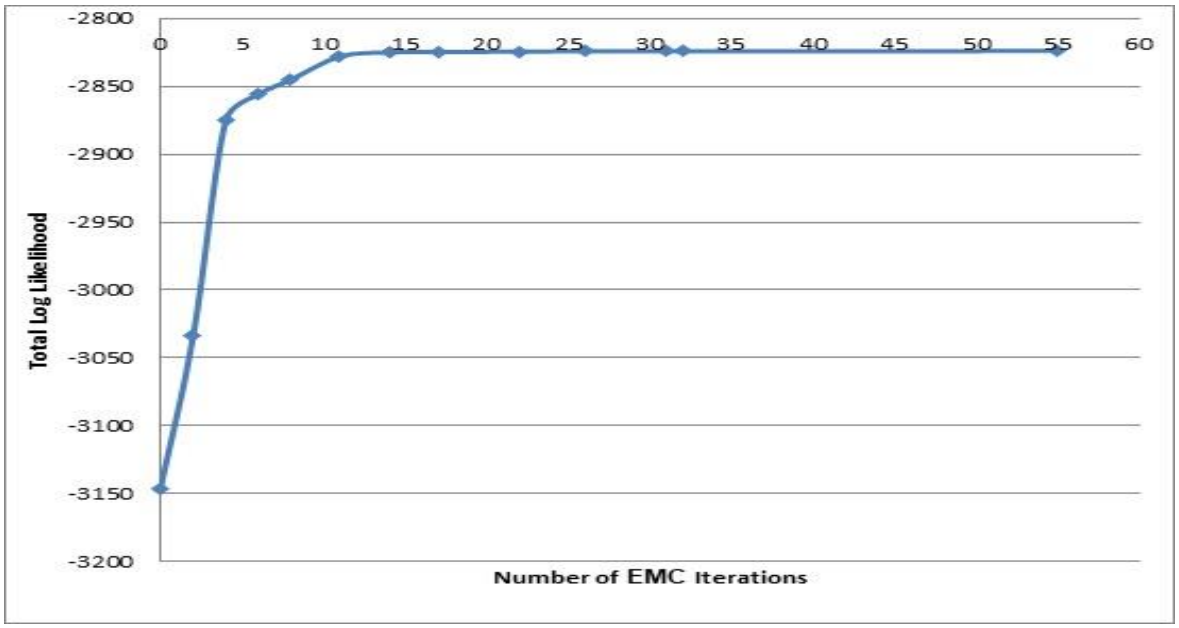
6	Kẹo	64	36	Bánh	88
7	Bánh	24	37	Bánh	53
8	Bánh	89	38	Bánh	72
9	Bánh	25	39	Bánh	48
10	Bánh	82	40	Bánh	44
11	Bánh	55	41	Bánh	83
12	Bánh	53	42	Bánh	77
13	Kẹo	88	43	Bánh	33
14	Sữa	54	44	Bánh	83
15	Bánh	73	45	Kẹo	82
16	Bánh	99	46	Bánh	83
17	Bánh	54	47	Bánh	95
18	Bánh	32	48	Bánh	75
19	Bánh	45	49	Sữa	31
20	Sữa	35	50	Bánh	79
21	Bánh	46	51	Bánh	33
22	Bánh	64	52	Bánh	90
23	Bánh	34	53	Kẹo	23
24	Sữa	67	54	Bánh	30
25	Bánh	53	55	Bánh	47
26	Sữa	97	56	Bánh	84
27	Bánh	97	57	Sữa	22
28	Bánh	77	58	Bánh	96
29	Bánh	44	59	Bánh	94
30	Sữa	72	60	Sữa	98

Giả sử số lượng các mặt hàng đều nằm trong khoảng từ 21 đến 99. Sử dụng thuật toán EMC với dữ liệu đầu vào bảng, số cụm 3 và giá trị $C_{v_{value}} = 15\%$. ta có kết quả của phân cụm như sau:

Bảng 2.2: Kết quả phân cụm của thuật toán EMC

CỤM 1		CỤM 2		CỤM 3	
TID	**Số lượng**	**TID**	**Số lượng**	**TID**	**Số lượng**
57	22	12	53	50	79
30	23	25	53	10	82
53	23	37	53	33	82
7	24	14	54	45	82
3	25	17	54	41	83
9	25	32	54	44	83
54	30	1	55	46	83
49	31	11	55	35	84
18	32	22	64	56	84
43	33	6	64	13	88
51	33			36	88
23	34			8	89
20	35			31	90
29	44			52	90
34	44			59	94
40	44			47	95
19	45			58	96
21	46			26	97
55	47			27	97
39	48			60	98
				16	99
				30	72
				38	72
				15	73
				2	74
				48	75
				28	77
				42	77
				4	78
				24	67

2.2.3.3 Đánh giá thuật toán EMC dựa trên Log Likehood



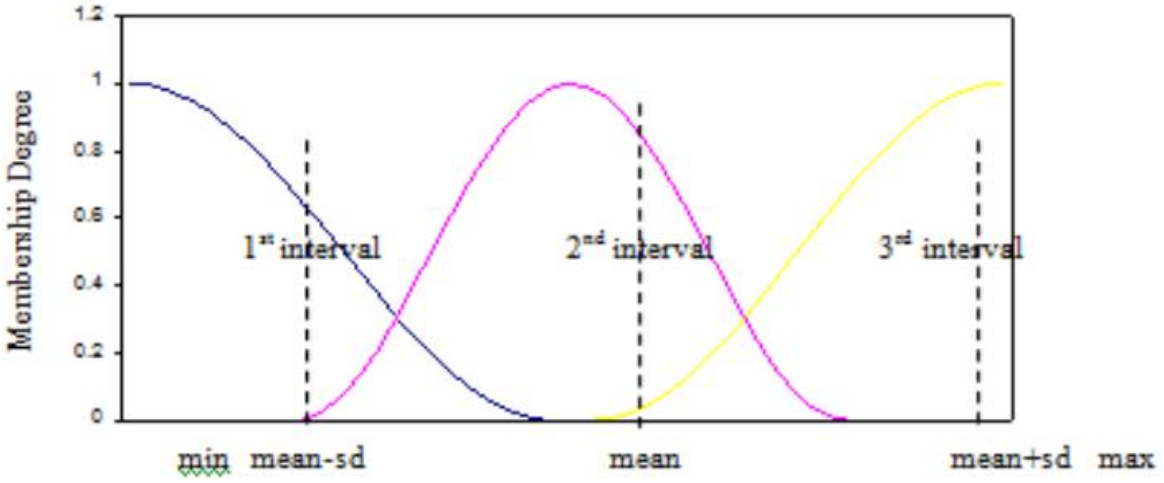
Hình 2.2: Tính tổng Log Likelihood đối với số lần lặp lại của thuật toán EMC

Thông qua kết quả thực nghiệm trong Hình 2.2, trong vùng giá trị (Total Log Likelihood (TLL)> -3150) của TLL, ta có thể tìm thấy kết quả tốt nhất từ tham số cho mô hình GMM. Các giá trị tính toán của C_v khác nhau tương ứng với từng cụm ảnh hưởng đến số lần lặp EMC rất nhiều. Giá trị của một C_v có thể thay đổi linh hoạt, điều này phụ thuộc vào số lượng cụm cũng như kích thước của mỗi cụm. Kết quả thu được từ thuật toán sẽ cho ra các cụm tối ưu và sử dụng chúng để phân loại từng thuộc tính định lượng thành một tập mờ bằng việc xác định các hàm thành viên.

2.2.4 Thuật toán xác định các khoảng mờ

2.2.4.1 Xác định tâm

Trong cơ sở dữ liệu mờ, miền giá trị của các thuộc tính định lượng của mục mờ mà trong đó (các thuộc tính có thể chứa giá trị rõ hoặc mờ) được chia thành hai hoặc nhiều khoảng mờ. Trong các khoảng mờ, một phần tử có thể thuộc nhiều hơn một khoảng với các mức độ khác nhau. Trong phần mục này, giả sử mỗi thuộc tính định lượng được chia thành ba khoảng mờ bằng phương pháp tiếp cận thống kê trong đó sử dụng kỳ vọng $\bar{x}$ (mean) và độ lệch chuẩn (Sd) được minh họa như trong hình 2.3.



Hình 2.3: Các khoảng mờ

Mức độ chồng lấp giữa các đối tượng dữ liệu mờ thuộc về hai hoặc nhiều cụm được định nghĩa như sau:

$$Overlap = \frac{\sum_{j=1}^{n} |C_j|}{\left|\bigcup_j^n C_j\right|} * 100 \quad (2.14)$$

Trong đó C_j là cụm thứ *j*, với *j=1, 2,..., n*

2.2.4.2 Xác định các khoảng mờ

Khoảng thứ nhất (1^{st} interval) Biên dưới (d^-) của khoảng thứ nhất là giá trị nhỏ nhất trong miền của thuộc tính định lượng. Biên trên (d^+) được tính bằng kỳ vọng $\bar{x}$ và độ lệch chuẩn (Sd) của các giá trị của thuộc tính định lượng. Biểu thức toán học của (d^-) và (d^+) được trình bày như sau:

$$\left.\begin{aligned} d^- &= MIN\left(X_{1Cj}, X_{2Cj}, \ldots\ldots, X_{NCj}\right) \\ d^+ &= \bar{x} - \frac{Sd}{2} + \bar{x} \times overlap \end{aligned}\right\} \quad (2.15)$$

Trong đó X_N là giá trị của cụm C_j với $N = 1,2,..,n$ và $j = 1,2,..n$.

Trong khoảng thứ nhất (1^{st} interval) hàm thành viên Z-membership được sử dụng để tính mức thành viên, đó là:

$$f(x)_Z = \frac{1}{2} + \frac{1}{2}\, cos\left(\frac{x - d^-}{d^+ - d^-}\right)\Pi \quad (2.16)$$

Khoảng thứ hai (2^{st} interval) Biên dưới (d^-) và biên trên (d^+) của khoảng thứ hai được tính như sau:

$$\left.\begin{aligned} d^- &= \bar{x} - \frac{Sd}{2} - \bar{x} * overlap \\ d^+ &= \bar{x} + \frac{Sd}{2} + \bar{x} * overlap \end{aligned}\right\} \quad (2.17)$$

Khoảng này sử dụng cả hàm thành viên S-membership và Z-membership, được biểu diễn như sau:

$$\left.\begin{aligned} f(x)_S &= \frac{1}{2} + \frac{1}{2}\, cos\left(\frac{\bar{x} - x}{\bar{x} - d^-}\right)\Pi, \text{với } d^- \le x \le \bar{x} \\ f(x)_Z &= \frac{1}{2} + \frac{1}{2}\, cos\left(\frac{x - \bar{x}}{d^+ - \bar{x}}\right)\Pi, \text{với } \bar{x} \le x \le d^+ \end{aligned}\right\} \quad (2.18)$$

Khoảng thứ ba (3^{st} interval) Biên dưới (d^-) và biên trên (d^+) của khoảng thứ ba được tính như sau:

$$\left.\begin{aligned} d^- &= \bar{x} - \frac{Sd}{2} - \bar{x} * overlap \\ d^+ &= MAX(X_{1Cj}, X_{2Cj}, \ldots., X_{NCj}) \end{aligned}\right\} \quad (2.19)$$

Khoảng này sử dụng hàm thành viên S-Membership có dạng như sau.

$$f(x)_S = \frac{1}{2} + \frac{1}{2}\, cos\left(\frac{d^+ - x}{d^+ - d^-}\right)\Pi \quad (2.20)$$

Ví dụ: Dựa trên kết quả phân cụm của thuật toán EMC, NCS sử dụng các cụm này để phân loại các giá trị của thuộc tính định lượng thành các tập mờ trong bảng 2.3, trong đó giá trị khoảng được xác định từ 21 đến 99.

Bảng 2.3: Tập mờ của thuộc tính định lượng "Số lượng"

Tập mờ	**Khoảng**
Số lượng.Thấp	1 - 48.85
Số lượng.Trung bình	45.58 - 75.50
Số lượng.Cao	72.35 - 100

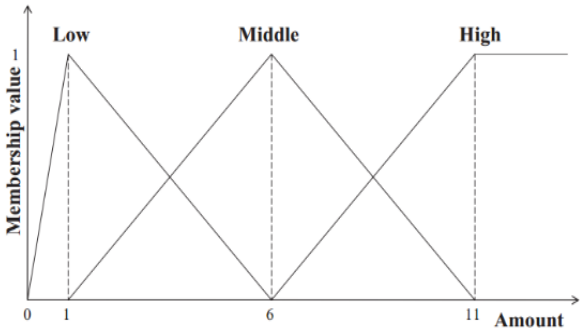
2.2.4.3 Chuyển đổi CSDL định lượng sang CSDL mờ

Sau khi xác định các khoảng mờ, cơ sở dữ liệu định lượng ban đầu được chuyển đổi thành cơ sở dữ liệu mờ, chuẩn bị cho quá trình khai phá luật kết hợp mờ. Đối với mỗi tập mờ mà chúng ta đã xác định trước đó, có một hàng trong cơ sở dữ liệu mới chứa mức độ thành viên của các phần tử đơn lẻ đối với tập cụ thể.

Ví dụ, trong bảng mô tả CSDL bao gồm sáu giao tác và năm mục được ký hiệu là A - E. Ngưỡng hỗ trợ tối thiểu là 30%.

Bảng 2.4: Cơ sở dữ liệu định lượng

TID	Items
1	(*A*:5) (*C*:10) (*D*:2) (*E*:9)
2	(*A*:8) (*B*:2) (*C*:3)
3	(*B*:3) (*C*:9)
4	(*A*:5) (*B*:3) (*C*:10) (*E*:3)
5	(*A*:7) (*C*:9) (*D*:3)
6	(*B*:2) (*C*:8) (*D*:3)



Hình 2.4: Hàm thành viên trong ví dụ

Giả sử rằng các hàm thành viên mờ giống nhau cho tất cả các mục được hiển thị trong Hình 2.4. Trong ví dụ này, số lượng được biểu thị bằng ba vùng mờ: Low, Middle và High.

Bảng 2.5: Cơ sở dữ liệu mờ sau khi chuyển đổi giá trị định lượng thành giá trị mờ.

TID	**Items**
1	$\left(\frac{0.2}{A.Low}+\frac{0.8}{A.Middle}\right),\left(\frac{0.2}{C.Middle}+\frac{0.8}{C.High}\right),\left(\frac{0.8}{D.Low}+\frac{0.2}{D.Middle}\right),\left(\frac{0.4}{E.Middle}+\frac{0.6}{E.High}\right)$
2	$\left(\frac{0.6}{A.Middle}+\frac{0.4}{A.High}\right),\left(\frac{0.8}{B.Low}+\frac{0.2}{B.Middle}\right),\left(\frac{0.6}{C.Low}+\frac{0.4}{C.Middle}\right)$
3	$\left(\frac{0.6}{B.Low}+\frac{0.4}{B.Middle}\right),\left(\frac{0.4}{C.Middle}+\frac{0.6}{C.High}\right)$
4	$\left(\frac{0.2}{A.Low}+\frac{0.8}{A.Middle}\right),\left(\frac{0.6}{B.Low}+\frac{0.4}{B.Middle}\right),\left(\frac{0.2}{C.Middle}+\frac{0.8}{C.High}\right),\left(\frac{0.6}{E.Low}+\frac{0.4}{E.Middle}\right)$
5	$\left(\frac{0.8}{A.Middle}+\frac{0.2}{A.High}\right),\left(\frac{0.4}{C.Middle}+\frac{0.6}{C.High}\right),\left(\frac{0.6}{D.Low}+\frac{0.4}{D.Middle}\right)$
6	$\left(\frac{0.8}{B.Low}+\frac{0.2}{B.Middle}\right),\left(\frac{0.6}{C.Middle}+\frac{0.4}{C.High}\right),\left(\frac{0.6}{D.Low}+\frac{0.4}{D.Middle}\right)$

2.3 Khai phá tập mục phổ biến mờ

2.3.1 Bài toán đặt ra

Cho cơ sở dữ liệu chứa các giá trị mờ D_f và độ hỗ trợ tối thiểu δ

Bài toán đặt ra: Tìm các tập mục phổ biến mờ có dạng:

$$FFI_k \coloneqq \{X | \, sup(X) \geq \delta \times |D_f|\}$$

2.3.2 Khai phá tập mục phổ biến mờ sử dụng cấu trúc cây FPPC-tree

2.3.2.1 Ý tưởng thuật toán

Từ CSDL chứa giá trị mờ D_f, tính độ hỗ trợ của mỗi mục mờ A_{il} trong giao tác T_{q}. Kiểm tra nếu độ hỗ trợ của mục mờ A_{il} lớn hơn độ hỗ trợ tối thiểu δ thì thêm A_{il} vào F_1. Sắp xếp các mục phổ biến mờ trong F_1 theo độ hỗ trợ giảm dần. Các mục mờ không phải là mục phổ biến mờ thì loại khỏi D_f. Xây dựng cây FPPC.

Sau khi xây dựng cây FPPC, bằng cách duyệt qua cây FPPC theo thứ tự pre-order, ta thu được Nodelist của mỗi mục phổ biến mờ (1-item). Với mỗi nút N_i, ta chèn $\langle N_i.pre, N_i.post, N_i.support \rangle$ vào Nodelist của mỗi mục được đại diện bởi N. Cây FPPC được xóa đi sau khi thu được Nodelist nhằm giảm không gian bộ nhớ.

Sau khi có được Nodelist của mỗi mục phổ biến 1-item, ta thực hiện giao Nodelist của các mục phổ biến 1-item để tìm Nodelist của tập mục (k-itemset). với bất kỳ ứng cử viên (k + 1) P_c nào, ta có được độ hỗ trợ của P_c bằng cách tính tổng giá trị độ hỗ trợ của tất cả các FPP_Code trong Nodelist của nó. Dựa vào độ hỗ trợ của P_c, chúng ta có thể đánh giá liệu P_c có phổ biến hay không. Bằng cách lặp lại quy trình trên, ta tìm được tất cả các mẫu mờ phổ biến.

2.3.2.2 Thuật toán xây dựng cây FPPC

Định nghĩa 2.4: Cây FPPC_tree là cấu trúc cây bao gồm:

(1) Một nút gốc được đánh nhãn "null" và tập các cây con tiền tố như là cây con của nút gốc.

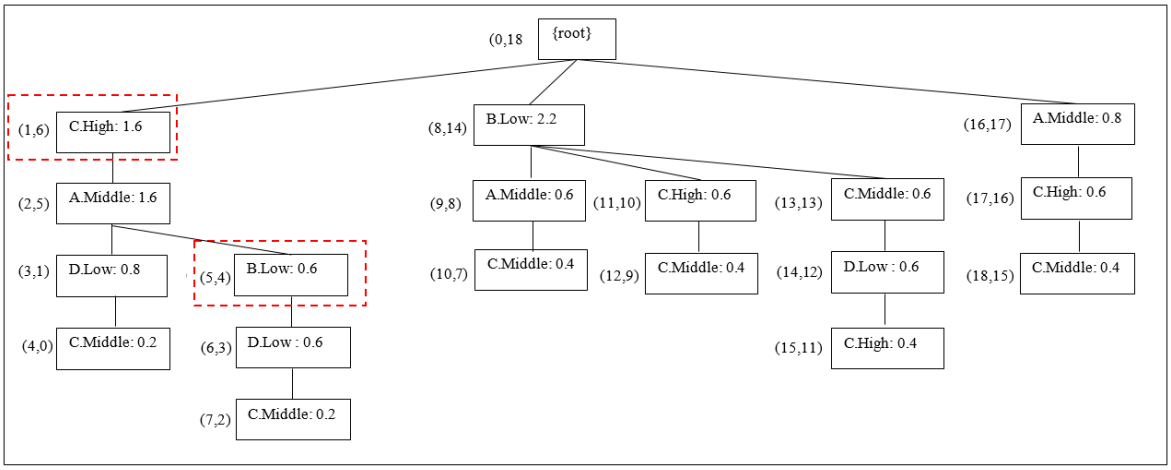
(2) Mỗi nút trong cây con tiền tố gồm 5 thành phần: *f_fuzzy_term, f_support, children_list, fpre_code* và *fpost_code* lần lượt là giá trị mờ của các thuộc tính định lượng, độ hỗ trợ mờ, danh sách các nút con, *pre_code và post_code* của mỗi nút.

Theo định nghĩa 2.6, cây FPPC tương tự như cây FFP_tree [38] hoặc MFFP_tree [42]. Tuy nhiên, cây FPPC_tree có những điểm khác biệt quan trọng so với các cây FFP_tree và MFFP_tree như sau:

(1) Đầu tiên, FFP-tree có trường node-link trong mỗi nút và cấu trúc bảng header để duy trì kết nối của các nút, trong đó FPPC-tree không có cấu trúc như vậy. Vì vậy, cây FPPC là cây tiền tố đơn giản hơn.

(2) Thứ hai, mỗi nút trong cây FPPC có các trường pre-order và post-order trong khi các nút trong cây FFP hoặc cây MFFP không có. Pre-order của một nút được xác định bởi cách duyệt cây theo thứ tự trước. Trong cách duyệt thứ tự trước, một nút N được thăm trước khi tất cả các con của nó được duyệt theo cách đệ quy từ trái sang phải. Theo cùng một cách, post-order của một nút được xác định bằng cách duyệt thứ tự sau. Trong một giao dịch theo thứ tự sau, một nút N được truy cập và được xếp hạng thứ tự sau khi tất cả các con của nó được truyền theo cách đệ quy từ trái sang phải.

(3) Thứ ba, sau khi cây FFP được xây dựng, nó sẽ được sử dụng để khai phá các mục phổ biến mờ trong toàn bộ quá trình thuật toán cây FFP, đây là một quá trình đệ quy và phức tạp. Tuy nhiên, cây FPPC chỉ được sử dụng để tạo mã Pre-Post của mỗi nút. Sau khi hoàn thành toàn bộ tác vụ của mình, cây FPPC có thể bị xóa.



Hình 2.5: Cây FPPC-tree được tạo ra từ CSDL với δ=30%

Thuật toán xây dựng cây FPPC được mô tả như trong Thuật toán 2.2

Thuật toán 2.2: Xây dựng cây FPPC_tree

Input: CSDL chứa giá trị mờ D_f, độ hỗ trợ mờ tối thiểu fminsup δ.

Output: FPPC-tree (FTr), tập mục mờ phổ biến 1-itemset (F_1).

(1) Duyệt qua CSDL mới D_f chứa giá trị mờ để tính độ hỗ trợ của mỗi mục mờ A_{il} trong giao tác T_q theo công thức:

$$sup(A_{il}) = \sum_{A_{il} \subseteq T_q \wedge T_q \in D_f} f_{il}$$

(2) Nếu $sup(A_{il}) \geq minsup\ \delta$, thêm A_{il} vào F_1. Ta có $F_1 = \{A_{il} \mid sup(A_{il}) \geq n \times \delta\}$.

(3) Sắp xếp các mục mờ phổ biến trong F_1 theo độ hỗ trợ giảm dần.

(4) Nếu $A_{il}\ not\ in\ F_1$, xóa A_{il} khỏi tất cả các T_q $(q = 1..n)$.

(5) Tạo nút root của cây FPPC và đánh nhãn "null"

(6) for each T_q in D_f {

(7) Sắp xếp các mục phổ biến còn lại theo độ hỗ trợ giảm dần.

(8) Chèn các mục mờ vào cây FFPC_tree (quy trình tương tự với MFFP_tree [42])

(9) }

(10) Duyệt cây FPPC để sinh PP_Code cho mỗi nút trên cây.

2.3.2.3 Thuật toán xây dựng Nodelist của các mục phổ biến mờ dựa trên cây FFPC

Zhihong Deng and Zhonghui Wang [20] đã đưa ra một số định nghĩa về Nodelist và các tính chất liên quan. Luận án này tổng quát và áp dụng vào các giá trị mờ như sau:

Định nghĩa 2.5: PP_Code

PP_code của mỗi nút trong cây FPPC là $C_i = \langle N_i.fpre\_code, N_i.fpost\_code, N_i.f\_support \rangle$.

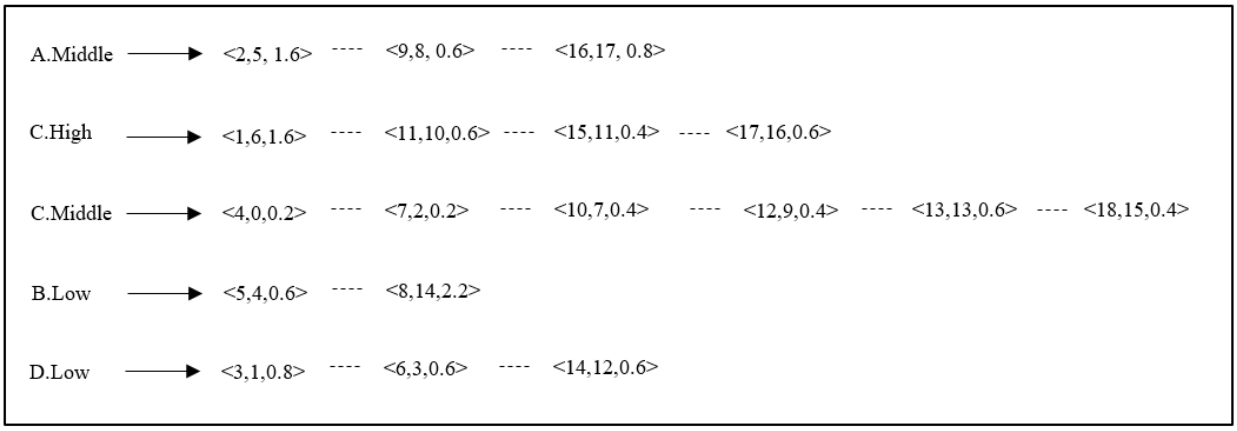
Tính chất 2.1: Node được gọi là nút tổ tiên của nút khi và chỉ khi $N_1.fpre\_code < N_2.fpre\_code$ và $N_1.fpost\_code > N_2.fpost\_code$.

Định nghĩa 2.6: Quan hệ nút tổ tiên – con cháu

Cho C_1 là $\langle pr_1, po_1, s_1\rangle$ và C_2 là $\langle pr_2, po_2, s_2\rangle$, C_1 được gọi là tổ tiên của C_2 khi và chỉ khi $pr_1 < pr_2$ và $po_1 > po_2$.

Định nghĩa 2.7: Nodelist của một mục mờ là một chuỗi PP_code của các nút đại diện cho các mục mờ. PP_codes được sắp xếp theo thứ tự fpre_code tăng dần. Mỗi PP_code trong Nodelist được biểu diễn bởi $\langle N_i.fpre\_code, N_i.fpost\_code, N_i.f\_support\rangle$.

Ví dụ: The Node list của A1.Middle gồm 3 nút: $\langle 2,5,1.6\rangle$, $\langle 9,8,0.6\rangle$, $\langle 16,17,0.8\rangle$. *Hình 2.6* thể hiện Nodelist của mỗi mục phổ biến mờ (1-item).



Hình 2.6: Nodelist của các mục mờ phổ biến

Tính chất 2.2

Cho $N_1 \neq N_2$ và $N_1.f\_item = N_2.f\_item$, nếu $N_1.fpre\_code < N_2.fpre\_code$ thì $N_1.fpost\_code < N_2.fpost\_code$.

Chứng minh:

Nếu $N_1.fpre\_code < N_2.fpre\_code$, theo cách duyệt cây theo pre-order thì nút N_1 sẽ được duyệt trước nút N_2. Nhưng N_1 không phải là nút tổ tiên của nút N_2 bởi vì chúng có cùng f_item, vì vậy N_1 nằm bên nhánh trái của N_2 trên cây. Theo cách duyệt ngược lại post-order, ta cũng duyệt từ trái sang phải, vì vậy $N_1.fpos\_code < N_2.fpos\_code$.

Tính chất 2.3

Nếu Nodelist của mục mờ A_i là $NL(A_i) = \{\langle pr_1, po_1, s_1\rangle, \langle pr_2, po_2, s_2\rangle, \dots, \langle pr_n, po_n, s_n\rangle\}$ thì độ hỗ trợ của mục mờ A_i là $s_1 + s_2 + \cdots + s_n$.

Ví dụ Nodelist of *A.Middle* gồm $\langle 2, 5, 1.6\rangle$, $\langle 9,8,0.6\rangle$, $\langle 16,17,0.8\rangle$ và độ hỗ trợ của A.*Middle* là *3.0.* Tất cả các mục mờ còn lại đều đúng với tính chất này.

Định nghĩa 2.8: (Quan hệ $\prec$)

Cho 2 mục mờ phổ biến fi_1 và fi_2 $(fi_1, fi_2 \in F_1)$. $fi_1 \prec fi_2$ khi và chỉ khi fi_1 đứng trước fi_2 trong F_1.

Thuật toán xây dựng Nodelist của mỗi mục phổ biến mờ (1-item) được mô tả trong thuật toán 2.3

Thuật toán 2.3: Nodelist_Construction

Input: FPPC-tree (R) and $L1$ (Tập các mục mờ phổ biến 1-item)

Output: NL_1 (Tập các Node list của L_1)

1: Tạo NL_1, $NL_1[k]$ là Nodelist của $L_1[k]$

2: **for each** node N_i in R duyệt theo tiền thứ tự **do**

3: **if** $N_i.f\_item = L_1[k].f\_item$ **then**

4: insert <*N.pre*, *N.post*, *N. support*> into NL_1[k]

5: **end if**

6: **end for**

7: **return** $NL_1 = \bigcup_k NL_1[k]$;

- ***Giao của Nodelist***

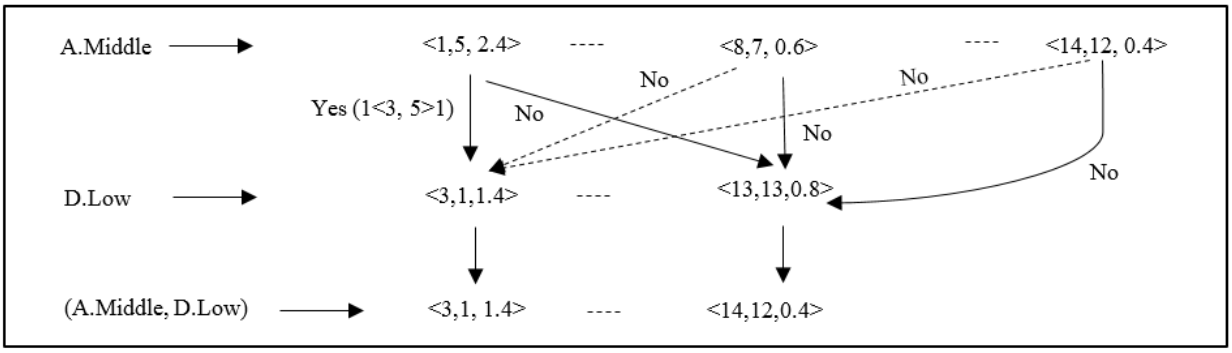
Trong tài liệu [20] tác giả đề xuất phương pháp Code_intersection để xác định Nodelist của k-itemsets chỉ phù hợp với trường hợp giá trị rõ, ở đó cây PPC_tree được xây dựng theo các tập mục được sắp xếp theo thứ tự độ hỗ trợ giảm dần. Nếu i_1 đứng trước i_2 trong L_1 thì tất cả các nút của i_1 luôn là tổ tiên của các nút của i_2, vì vậy tác giả chỉ cần kiểm tra xem Nodelist của nút i_1 có phải là tổ tiên của Nodelist của i_2

không. Đối với trường hợp giá trị mờ, thì mục mờ i_1 có thể là tổ tiên của i_2 trong nhánh này của cây nhưng có thể là nút con cháu trong nhánh khác của cây. Luận án đề xuất phương pháp cải tiến Nodelist_intersection của các tập mục mờ phổ biến k-itemsets để thu được Nodelist của tập phổ biến mờ (k+1) itemsets.

Định nghĩa 2.9: (Nodelist của 2 tập phổ biến mờ (1-item)).

Giả sử có 2 mục mờ phổ biến fi_1, fi_2 trong đó fi_1 đứng trước fi_2 trong F_1 và Nodelist lần lượt là $NF(fi_1)$ và $NL(fi_2)$. Cho mỗi PP_code $C_{1p} =< pr_{1p}, po_{1p}, s_{1p} > \in NL(fi_1)$ và $C_{2q} =< pr_{2q}, po_{2q}, s_{2q} > \in NL(fi_2)$. Nếu C_{1p} và C_{2q} có quan hệ ancestor desscendant (quan hệ phân cấp) của PP_code thì chèn PP_Code con (cháu) vào Nodelist của $fi_1 fi_2$.

Ví dụ: Nodelist của 2 mục mờ A.Middle và D.Low được thể hiên trong hình dưới



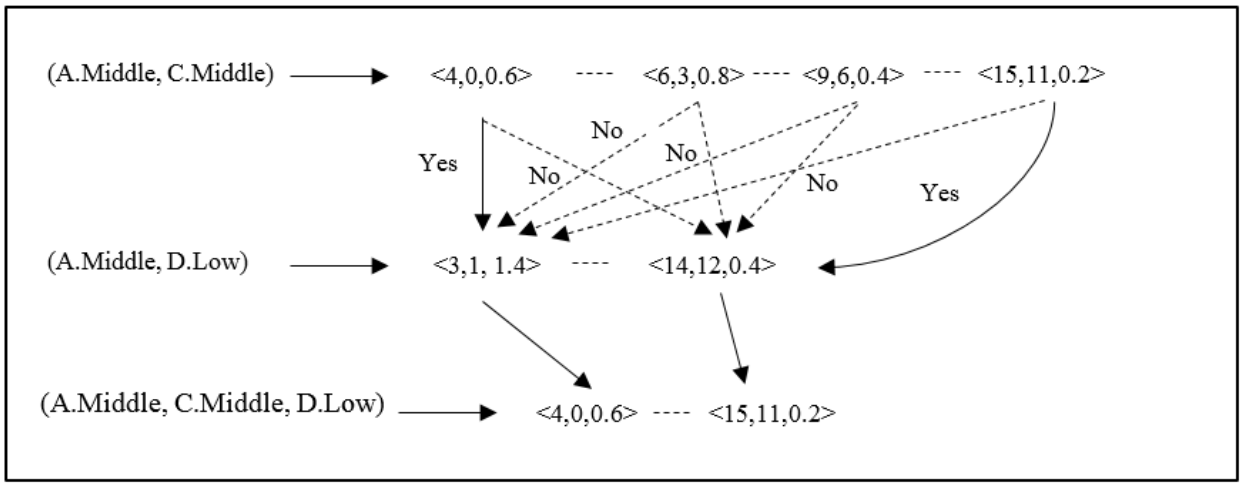
Hình 2.7: Nodelist của A.Middle và D.Low trong ví dụ

Dựa vào định nghĩa trên, NCS tổng quát khái niệm Nodelist của tập mục mờ k-itemsets ($k \geq 3$) như sau:

Định nghĩa 2.10: (Nodelist của 2 tập phổ biến mờ có độ dài k >=3)

Cho $P = i_1 i_2 \dots i_{(k-2)} i_u i_v$ là tập phổ biến mờ, Node list của $P_1 = i_1 i_2 \dots i_{(k-2)} i_u$ là $(\langle pr_{11}, po_{11}, s_{11}\rangle, \langle pr_{12}, po_{12}, s_{12}\rangle, \dots, \langle pr_{1m}, po_{1m}, s_{1m}\rangle)$, the Node list of $P_2 = i_1 i_2 \dots i_{(k-2)} i_v$ be $(\langle pr_{21}, po_{21}, s_{21}\rangle, \langle pr_{22}, po_{22}, s_{22}\rangle, \dots, \langle pr_{2n}, po_{2n}, s_{2n}\rangle)$. Với bất kỳ PP_code $C_{1p} =< pr_{1p}, po_{1p}, s_{1p} > \in NL(P_1)$ và $C_{2q} =< pr_{2q}, po_{2q}, s_{2q} > \in NL(P_2)$. Nếu C_{1p} và C_{2q} có quan hệ ancestor desscendant (quan hệ phân cấp) của PP_code thì chèn PP_Code con (cháu) vào Nodelist của $i_1 i_2 \dots i_{(k-2)} i_u i_v$.

Ví dụ, ta đã có Nodelist của (A.Middle, C.Middle) và (A.Middle, D.Low) lần lượt là <4,0,0.6><6,3,0.8>, <9,6,0.4>, <15,11,0.2> và <3,1,1.4>, <14,12,0.4>. Vậy Nodelist của tập mục mờ (A.Middle, C.Middle, D.Low) được thể hiện trong hình dưới:



Hình 2.8: Nodelist của tập mục mờ (A.Middle, C.Middle, D.Low)

Tính chất 2.4

Với bất cứ Nodelist của tập mờ có độ dài k $P = i_1 i_2 \dots i_k$ được biểu diễn $\{\langle pr_1, po_{1,} s_1\rangle, \langle pr_2, po_{2,} s_2\rangle, \dots, \langle pr_k, po_{k,} s_k\rangle\}$ thì độ hỗ trợ của P là $s_1 + s_2 + \cdots + s_k$.

Chứng minh:

- Với k=1, theo tính chất 2.3, kết luận là đúng.
- Với $k \geq 2$, thuật toán được đề xuất đã sắp xếp các mục mờ theo thứ tự giảm dần của độ hỗ trợ trong mỗi giao tác để xây dựng cấu trúc cây FPPC, và sử dụng τ-norm trong tập mục là $a\ \tau\ b$ $(min(a, b))$ như toán tử giao, vì vậy ta thu được giá trị nhỏ nhất của các mục mờ từ độ hỗ trợ của các nút con. Với bất kỳ tập mục mờ độ dài k $P = i_1 i_2 \dots i_k$ có Nodelist là $NL(P) = \{\langle pr_1, po_{1,} s_1\rangle, \langle pr_2, po_{2,} s_2\rangle, \dots, \langle pr_k, po_{k,} s_k\rangle\}$, theo định nghĩa 2.9, 2.10 thì $\forall C_i \in NL(P)$ là các nút con. Vì vậy độ hỗ trợ của P theo tính chất 2.3 là $s_1 + s_2 + \cdots + s_k$.

Theo tính chất 2.4 thì độ hỗ trợ của tập mục mờ (A.Middle, C.Middle, D.Low) là 0.8.

Thuật toán thực hiện giao Nodelist của 2 tập phổ biến mờ có độ dài k được mô tả trong thuật toán 2.4.

Thuật toán 2.4: Thuật toán FNodelist_Intersection

Input: NL_1 và NL_2 trong đó NL_1 , NL_2 là các Nodelist của 2 tập phổ biến mờ có độ dài k.

Output: NL_3 là Nodelist của tập phổ biến mờ có độ dài (k+1) .

(1) for $(i = 0; i < NL_1.Size(); i++)$ do
(2) for $(j = 0; i < NL_2.Size(); j++)$ do
(3) if $(NL_1[i].fpre\_code < NL_2[j].fpre\_code)$ then
(4) if $(NL_1[i].fpos\_code > NL_2[j].fpos\_code)$ then
(5) Thêm $NL_2[j]$ vào NL_3;
(6) End if
(7) else
(8) if $(NL_1[i].fpos\_code < NL_2[j].fpos\_code)$ then
(9) Thêm $NL_1[i]$ vào NL_3;
(10) End if
(11) End if
(12) End for
(13) return NL_3
(14) End for

2.3.2.4 Thuật toán NFFP

Thuật toán NFFP được trình bày hai đóng góp chính: (1) cải thiện thuật toán tìm Nodelist của các mục mờ phổ biến (k-itemsets) từ [20], (2) đề xuất cách tiếp cận mới bằng cách sử dụng cấu trúc Nodelist để tìm các mục mờ phổ biến, cách tiếp cận này giúp giảm bớt các yêu cầu sử dụng bộ nhớ, vì lưu trữ cây FPPC trong quá trình khai phá các mục mờ phổ biến là không cần thiết.

NCS áp dụng phương pháp tương tự Apriori để khai phá các mẫu phổ biến. Đầu tiên, ta tạo ra Nodelist của các ứng cử viên (k + 1) bằng cách thực hiện giao các Nodelist của các mẫu có độ dài k phổ biến. Thứ hai, đối với bất kỳ ứng cử viên (k + 1) P_c nào, ta có được độ hỗ trợ của P_c bằng cách tính tổng giá trị độ hỗ trợ của tất cả các FPP_Code trong Nodelist của nó. Dựa vào độ hỗ trợ của P_c, chúng ta có thể đánh

giá liệu P_c có phổ biến hay không. Bằng cách lặp lại quy trình trên, ta tìm được tất cả các mẫu mờ phổ biến.

Thuật toán NFFP được mô tả như trong Thuật toán 2.5

Thuật toán 2.5: Thuật toán khai phá tập mục mờ phổ biến NFFP

Input: độ hỗ trợ mờ tối thiểu *fminsup* (δ), tập phổ biến mờ (1-item) (L_1), Nodelist của $L1$ ($NL1$);

Output: Tập mục mờ phổ biến (FFIs)

(1) For $(k = 2; L_{k-1} \neq \emptyset; k++)$ do begin
(2) For each $p = i_1 i_2 \dots i_{k-2} i_x \in L_{k-1}$ and $q = i_1 i_2 \dots i_{k-2} i_y \in L_{k-1}$, do
(3) If $i_x \succ i_y$ then
(4) $l = i_1 i_2 \dots i_{k-2} i_x i_y$
(5) If each k-1 subsets l in L_{k-1} then begin
(6) l.Node-list = NL_Intersection (p.Node-list, q.Node-list);
(7) Tính $l.support$; // Sử dụng tính chất 2.4
(8) If $(l.support \geq n \times \delta)$ then begin
(9) $L_k = L_k \cup \{l\}$;
(10) $NL_k = NL_k \cup \{l.Nodelist\}$;
(11) end if
(12) end if
(13) end if
(14) end for
(15) Xóa NL_{k-1};
(16) end for
(17) $FFIs = \bigcup_k L_k$

Bảng 2.6 cho thấy kết quả cuối cùng của các mục mờ phổ biến trong ví dụ minh họa.

Bảng 2.6 Các tập mục mờ phổ biến trong ví dụ

Fuzzy frequent 1-itemsets	**Support**
{A. Middle}	3.4
{C.High}	2.8
{C.Middle}	2.4
{B.Low}	2.2
{D.Low}	2.2
Fuzzy frequent 2-itemsets	**Support**
{A.Middle, C.High}	1.8
{A.Middle, C.Middle}	2.0
{A.Middle, D.Low}	1.8
{C.High, D.Low}	2.2
Fuzzy frequent 3-itemsets	**Support**
{A.Middle, C.High, D.Low}	1.8

2.3.3 Khai phá tập mục phổ biến sử dụng cấu trúc cây FPOSC-tree

2.3.3.1 Ý tưởng thuật toán

Từ CSDL chứa giá trị mờ D_f, tính độ hỗ trợ của mỗi mục mờ A_{il} trong giao tác T_{q}. Kiểm tra nếu độ hỗ trợ của mục mờ A_{il} lớn hơn độ hỗ trợ tối thiểu δ thì thêm A_{il} vào F_1. Sắp xếp các mục phổ biến mờ trong F_1 theo độ hỗ trợ giảm dần. Các mục mờ không phải là mục phổ biến mờ thì loại khỏi D_f. Xây dựng cây FPOSC.

Trong khi xây dựng cây FPOSC có thể thêm một số nút con mà không cần phải duyệt lại cây và pre-order được tính toán cùng lúc với việc xây dựng Node-list của các mục mờ phổ biến. Với mỗi nút N_i, ta chèn $\langle N_i.pre, N_i.size, N_i.f\_sup\rangle$ vào Nodelist của mỗi mục được đại diện bởi N. Cây FPOSC được xóa đi sau khi thu được Nodelist nhằm giảm không gian bộ nhớ.

Sau khi có được Nodelist của mỗi mục phổ biến 1-item, ta thực hiện giao Nodelist của các mục phổ biến 1-item để tìm Nodelist của tập mục (k-itemset). với bất kỳ ứng cử viên (k + 1) P_c nào, ta có được độ hỗ trợ của P_c bằng cách tính tổng giá trị độ hỗ trợ của tất cả các FPP_Code trong Nodelist của nó. Dựa vào độ hỗ trợ của

P_c, chúng ta có thể đánh giá liệu P_c có phổ biến hay không. Bằng cách lặp lại quy trình trên, ta tìm được tất cả các mẫu mờ phổ biến.

2.3.3.2 Thuật toán xây dựng cây FPOSC (Fuzzy Pre-order Size Coding)

Định nghĩa 2.11

Cây FPOSC thực chất là một cải tiến của cây FPPC [CT1], sự khác biệt quan trọng giữa cây FPPC và FPOSC là cấu trúc dữ liệu được thay thế từ (pre-order, post-order) thành (pre-order, size). Cấu trúc của cây FPOSC chứa: nút gốc có nhãn "null" và tập hợp cây con. Mỗi nút trên cây bao gồm: tên nút (f_item), độ hỗ trợ mờ của nút (f_sup), danh sách các nút con (sub_nodes), pre-order và size được xác định bởi các nút con).

- Tên nút đại diện cho mục mờ tham chiếu tới một nút.
- Danh sách các nút con chứa các nút con bên trong chúng.
- f_sup đại diện cho tổng độ hỗ trợ mờ của mục trong giao dịch có cùng tên với nút có cùng đường dẫn.
- pre-order về cơ bản là một thứ tự được sử dụng để gán mọi nút thông qua việc chuyển cây theo thứ tự trước.
- size là kích thước của node được tính tổng số nút con cộng 1.

Trong [CT1], pre-order và post-order được thực hiện sau khi cây được xây dựng hoàn chỉnh. Với cây FPOSC, trong quá trình xây dựng, có thể thêm một số nút con mà không cần phải duyệt lại cây và pre-order được tính toán cùng lúc với việc xây dựng Node-list của các mục mờ phổ biến (length = 1). Do đó, nó giúp giảm thời gian xây dựng cây. Thuật toán xây dựng cây FPOSC được xác định bằng cách điều chỉnh cấu trúc của cây FPPC [CT1], được trình bày trong thuật toán 2.6.

Thuật toán 2.6: FPOSC-Tree_Construction

Input: Fuzzy Database D_f, fminsup δ

Output: FT_r (FPOSC-tree), F_1 (frequent fuzzy itemsets (length=1)

Begin

1: Duyệt D_f tính độ hỗ trợ của mỗi mục mờ A_{jk} trong giao dịch T_i

2: If $fsup(A_{jk}) \geq \delta$ then

3: Thêm A_{jk} vào F_1;

4: End if

5: If A_{jk} not in F_1 then

6: Xóa A_{jk} ra khỏi tất cả T_i $(i = 1 \dots n)$

7: End if

8: Tạo FT_r NodeRoot=null

9: Đặt Flist là danh sách chứa các mục mờ còn lại trong mỗi T_i

10: For each T_i in D_f

11: Sắp xếp FList theo thứ tự fsup giảm dần

12: $e = FList[0]$; e là phần tử đầu tiên trong Flist

13: $List_r = List[size - 1]$

14 Insert_tree $([e| List_r], FT_r)$

15: End for

/* Thủ tục Insert_Tree được sử dụng để gọi đệ quy việc xây dựng cây POSC. Trong đó, e là phần tử đầu tiên của Flist và Flist là danh sách còn lại */

Procedure Insert_tree $([e| List_r], FT_r)$

1: Gọi N là một nút tương ứng với một nhánh trong FT_r

2 If $e.fitem == N.fitem$ then

3: Cộng giá trị mờ $f_{j,k}^{(i)}$ của e vào fsup của N

4: Else

5: Tạo nút mới N có fsup là $f_{j,k}^{(i)}$ và thêm N vào cuối nhánh tương ứng.

6: $N.size = 1$

7: If $List_r$ is nonempty then

8: Gọi Insert_Tree ($List_r$,N) recursively

9: End if

10: End if

11: $N.size = N.countChild + 1$

End procedure

Để minh họa, trong ví dụ này, NCS sử dụng cơ sở dữ liệu định lượng (D_Q) được trình bày trong Bảng 2.7 với sáu giao dịch và năm mục được chỉ định từ I_1 đến I_5 và minsup δ được thiết lập là 30%. Giả sử thuộc tính số lượng của mỗi mặt hàng được biểu diễn một trong các thuật ngữ mờ sau: Low, Medium và High. Sau khi

chuyển đổi từ giá trị rõ sang giá trị mờ, chúng ta có cơ sở dữ liệu cập nhật (D_f) như trong Bảng 2.8.

Bảng 2.7: Cơ sở dữ liệu định lượng trong ví dụ

TID	**Items**
1	(I_1:3) (I_3:9) (I_4:3) (I_5:9)
2	(I_1:8) (I_2:3) (I_3:3)
3	(I_2:3) (I_3:10)
4	(I_1:7) (I_2:2) (I_3:9) (I_5:3)
5	(I_1:9) (I_3:7) (I_4:3)
6	(I_2:3) (I_3:8) (I_4:2)

Bảng 2.8: Cơ sở dữ liệu mờ được chuyển đổi từ bàng 2.7

TID	**Items**
1	$\left(\frac{0.6}{I_1.Low}+\frac{0.4}{I_1.Middle}\right), \left(\frac{0.4}{I_3.Middle}+\frac{0.6}{I_3.High}\right), \left(\frac{0.6}{I_4.Low}+\frac{0.4}{I_4.Middle}\right), \left(\frac{0.4}{I_5.Middle}+\frac{0.6}{I_5.High}\right)$
2	$\left(\frac{0.6}{I_1.Middle}+\frac{0.4}{I_1.High}\right), \left(\frac{0.6}{I_2.Low}+\frac{0.4}{I_2.Middle}\right), \left(\frac{0.6}{I_3.Low}+\frac{0.4}{I_3.Middle}\right)$
3	$\left(\frac{0.6}{I_2.Low}+\frac{0.4}{I_2.Middle}\right), \left(\frac{0.2}{I_3.Middle}+\frac{0.8}{I_3.High}\right)$
4	$\left(\frac{0.8}{I_1.Middle}+\frac{0.2}{I_1.High}\right), \left(\frac{0.8}{I_2.Low}+\frac{0.2}{I_2.Middle}\right), \left(\frac{0.4}{I_3.Middle}+\frac{0.6}{I_3.High}\right), \left(\frac{0.6}{I_5.Low}+\frac{0.4}{I_5.Middle}\right)$
5	$\left(\frac{0.4}{I_1.Middle}+\frac{0.6}{I_1.High}\right), \left(\frac{0.8}{I_3.Middle}+\frac{0.2}{I_3.High}\right), \left(\frac{0.6}{I_4.Low}+\frac{0.4}{I_4.Middle}\right)$
6	$\left(\frac{0.6}{I_2.Low}+\frac{0.4}{I_2.Middle}\right), \left(\frac{0.6}{I_3.Middle}+\frac{0.4}{I_3.High}\right), \left(\frac{0.8}{I_4.Low}+\frac{0.2}{I_4.Middle}\right)$

Bảng 2.9 biểu diễn độ hỗ trợ của các tập hợp mục mờ phổ biến (độ dài = 1) sau khi loại bỏ tất cả các mục không thỏa mãn δ.

Bảng 2.9: Độ hỗ trợ của tập phổ biến mờ 1-item

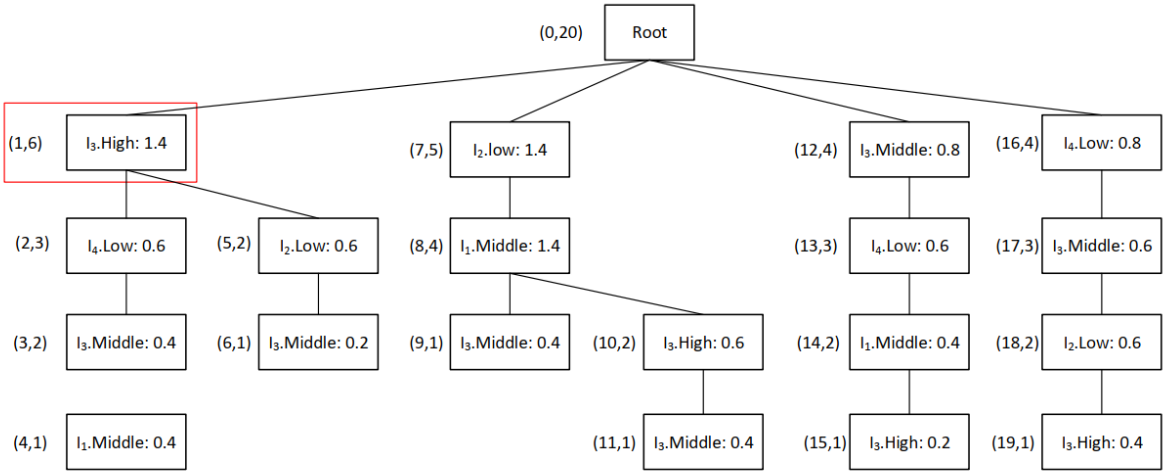
Frequent fuzzy 1-itemset	**Fuzzy Support**
I_3.Middle	2.8
I_2.Low	2.6
I_3.High	2.6
I_1.Middle	2.2
I_4.Low	2.0

Sau khi tìm được tập mục phổ biến mờ (với độ dài 1), thuật toán sẽ loại bỏ các mục không thỏa mãn độ hỗ trợ tối thiểu ra khỏi tất cả các giao tác và các giao tác được cập nhật được hiển thị trong bảng 2.10.

Bảng 2.10: Giao dịch sau khi được cập nhật có chứa các tập hợp mục mờ

TID	**Items**
1	$\left(\frac{0.6}{I_3.High}\right), \left(\frac{0.6}{I_4.Low}\right), \left(\frac{0.4}{I_3.Middle}\right), \left(\frac{0.4}{I_1.Middle}\right)$
2	$\left(\frac{0.6}{I_2.Low}\right), \left(\frac{0.6}{I_1.Middle}\right), \left(\frac{0.4}{I_3.Middle}\right)$
3	$\left(\frac{0.8}{I_3.High}\right), \left(\frac{0.6}{I_2.Low}\right), \left(\frac{0.2}{I_3.Middle}\right)$
4	$\left(\frac{0.8}{I_2.Low}\right), \left(\frac{0.8}{I_1.Middle}\right), \left(\frac{0.6}{I_3.High}\right), \left(\frac{0.4}{I_3.Middle}\right)$
5	$\left(\frac{0.8}{I_3.Middle}\right), \left(\frac{0.6}{I_4.Low}\right), \left(\frac{0.4}{I_1.Middle}\right), \left(\frac{0.2}{I_3.High}\right)$
6	$\left(\frac{0.8}{I_4.Low}\right), \left(\frac{0.6}{I_3.Middle}\right), \left(\frac{0.6}{I_2.Low}\right), \left(\frac{0.4}{I_3.High}\right)$

Để tạo FPOSC-tree, tương ứng với mỗi giao dịch là một nhánh của cây, ta lần lượt chèn các mục mờ trong các giao dịch vào nhánh. Nếu phần tử đầu tiên của giao dịch trùng tên với nút của một nhánh hiện có, thì chúng ta cộng thêm độ hỗ trợ mờ của phần tử đó vào hỗ trợ mờ của nút hiện có.



Hình 2.9: Cây FPOSC

2.3.3.3 Thuật toán xây dựng Nodelist của các mục phổ biến mờ dựa trên cây FPOSC

Định nghĩa 2.12: POS-code

Đối với mọi nút trong cây FPOSC, POS-code có dạng là $\langle pre, size, f\_sup \rangle$.

Ví dụ: node N_1 trong Hình 2.9 có POS-code là $POSC_1$= <1,6,1.4>

Tính chất 2.5

Giả sử có hai nút phân biệt N_1 và N_2, N_1 được gọi là nút cha/tổ tiên của nút N_2 khi và chỉ khi $N_1.pre < N_2.pre < (N_1.pre + N_1.size)$.

Thuộc tính này đã được chứng mình trong [83].

Định nghĩa 2.13: Nút cha/tổ tiên

Cho $POSC_1$ là $\langle pre_1, size_1, f\_sup_1 \rangle$ và $POSC_2$ là $\langle pre_2, size_2, f\_sup_2 \rangle$. $POSC_1$ là cha/tổ tiên của $POSC_2$ khi và chỉ khi $pre_1 < pre_2 < pre_1 + size_1$.

Định nghĩa 2.14: Nodelist theo POS-code

Cho cây POSC_tree, Node-list của tập mục mờ là một chuỗi các POS-codes đại diện cho mục mờ trong các nhánh khác nhau của cây POSC. Các POS-codes được sắp xếp theo thứ tự pre-order. Mỗi POS-code có dạng: $\langle N_i.pre, N_i.size, N_i.f\_sup \rangle$.

Thuật toán xây dựng Node-list của tập mục phổ biến mờ (length=1) F_1 được trình bày trong Thuật toán 2.7.

Thuật toán 2.7: FNode_List_Gen

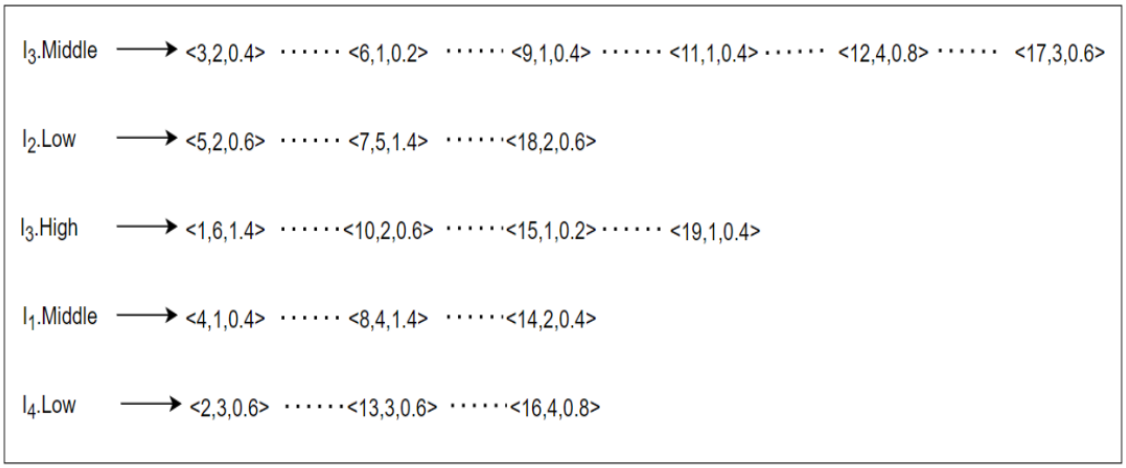
Input: POSC-tree (FT_r), tập mờ phổ biến length=1 (F_1)
Output: Node-list của F_1 (NL_1)

Begin

1: for each N_i in FT_r (được duyệt theo thứ tự trước trong FT_r)
2: Gọi NL_1[k] là Node-list của mục k^{th} trong F_1.
3: If $N_i.f_{item} == F_1[k].f\_item$ then
4: Thêm $\langle N_i.pre, N_i.size, N_i.fsup \rangle$ vào NL_1[k];
5: Return $NL_1 = \cup_k NL_1[k]$

End.

Ví dụ: Node-list của $I_3.High$ gồm có 4 nodes: ⟨1,6,1.4⟩, ⟨10,2,0.6⟩, ⟨15,1,0.2⟩ và ⟨19,1,0.4⟩. Hình 2.10 thể hiện Node-list của tất cả các mục mờ phổ biến (F_1) trong ví dụ minh họa.



Hình 2.10: The Node-list của các mục mờ phổ biến 1-item

Tính chất 2.6

Cho Node-list của mục mờ A_i là $NL(A_i) = \{\langle pre_1, size_1, fsup_1\rangle, \langle pre_2, size_2, fsup_2\rangle, \dots, \langle pre_n, size_n, fsup_n\rangle\}$ thì độ hỗ trợ của A_i là $fsup_1 + fsup_2 + \cdots + fsup_n$.

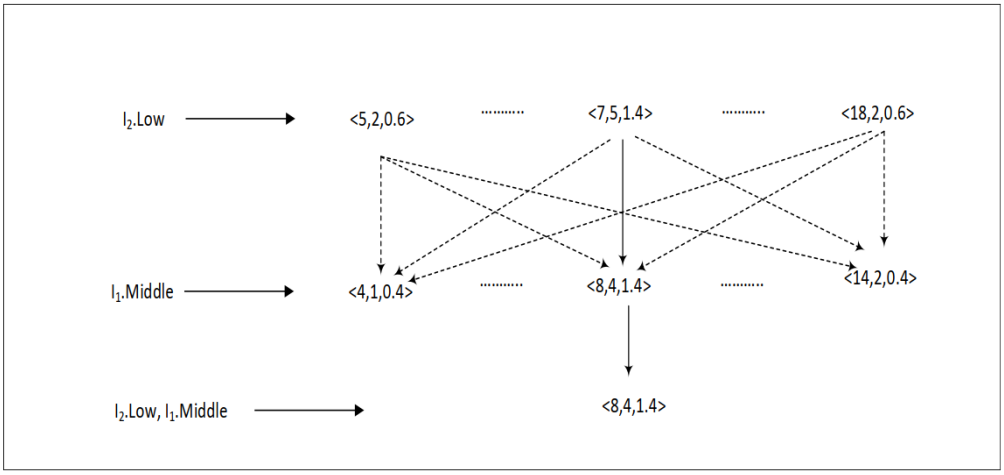
Ví dụ Node-list của $I_3.High$ gồm có 4 nodes: ⟨1,6,1.4⟩, ⟨10,2,0.6⟩, ⟨15,1,0.2⟩ và ⟨19,1,0.4⟩ và độ hỗ trợ của $I_3.High$ là 2.6. Điều này đúng với tất cả các mục mờ phổ biến trong Bảng 2.9.

Quy trình thực hiện giao Node-list được thực hiện như trong [CT2] chi tiết như sau:

(1) Giả sử Node-list của hai mục mờ phổ biến khác nhau fi_1 và fi_2 lần lượt là: $NL(fi_1)$ và $NL(fi_2)$. Mỗi POS-code $C_{1p} = \langle pre_{1p}, size_{1p}, fsup_{1p}\rangle \subset NL(fi_1)$ và $C_{2q} = \langle pre_{2q}, size_{2q}, fsup_{2q}\rangle \subset NL(fi_2)$. Nếu C_{1p} và C_{2q} có quan hệ ADRs (theo định nghĩa 3) nghĩa là $pre_{1p} < pre_{2q} < pre_{1p} + size_{1p}$ hoặc $pre_{2q} < pre_{1p} < pre_{2q} + size_{2q}$ thì đặt POS-code con vào Node-list của fi_1 fi_2.

(2) Giả sử $P = i_1 i_2 \dots i_{k-2} i_u i_v$ là tập mờ phổ biến với $(k \geq 3)$, Node-list của $P_1 = i_1 i_2 \dots i_{k-2} i_u$ là $(\langle pre_{11}, size_{11}, fsup_{11}\rangle, \langle pre_{12}, size_{12}, fsup_{12}\rangle, \dots, \langle pre_{1m}, size_{1m}, fsup_{1m}\rangle)$, Node-list của $P_2 = i_1 i_2 \dots i_{k-2} i_v$ là $(\langle pre_{21}, size_{21}, fsup_{21}\rangle, \langle pre_{22}, size_{22}, fsup_{22}\rangle, \dots, \langle pre_{2n}, size_{2n}, fsup_{2n}\rangle)$. Với mỗi POS-code $C_{1p} = \langle pre_{1p}, size_{1p}, fsup_{1p}\rangle \subset NL(P_1)$ và $C_{2q} = \langle pre_{2q}, size_{2q}, fsup_{2q}\rangle \subset$

$NL(P_2)$, nếu C_{1p} và C_{2q} có quan hệ ADRs thì đặt POS-code con vào Node-list của $P = i_1 i_2 \dots i_{k-2} i_u i_v$.



Hình 2.11: Giao Nodelist của I_2.Low và I_1.Middle

Theo Hình 2.11, Node-list của mục mờ phổ biến $I_2.Low$ là $\{\langle 5,2,0.6\rangle, \langle 7,5,1.4\rangle, 18,2,0.6\}$ và Node-list của mục mờ phổ biến $I_1.Middle$ là $\{\langle 4,1,0.4\rangle, \langle 8,4,1.4\rangle, 14,2,0.4\}$. Để xây dựng Node-list của $(I_2.Low, I_1.Middle)$ NCS kiểm tra ADRs POS-code trong Node-list của $I_2.Low$ và POS-code trong Node-list của $I_1.Middle$. Trong ví dụ, ADRs giữa $\langle 7,5,1.4\rangle$ và $\langle 8,4,1.4\rangle$ thỏa với tính chất 1. Vì vậy, NCS thêm $\langle 8,4,1.4\rangle$ vào Node-list của $(I_2.Low, I_1.Middle)$.

Tính chất 2.7

Với mọi $P = i_1 i_2 \dots i_k$ là tập mờ phổ biến k-itemset và $\{\langle pre_1, size_1, fsup_1\rangle, \langle pre_2, size_2, fsup_2\rangle, \dots, \langle pre_k, size_k, fsup_k\rangle\}$ là Node-list của P, thì độ hỗ trợ của P là $fsup_1 + fsup_2 + \cdots + fsup_k$.

Tính chất này đã được chứng minh trong [CT1]. Phương thức xây dựng giao của hai Node-list được thực hiện trong thuật toán POS Node-list Intersect.

Thuật toán 2.8: POS_Node-list_Intersect

Input: NL_{k1}, NL_{k2} trong đó NL_{k1}, NL_{k2} là Node-list của 2 tập mục mờ k-itemsets
Output: NL_{k1+1} – Node-list của tập mục mờ (k+1) itemsets

Begin

1: for $i = 0; i < NL_{k1}.length; i++$ do
2: For $j = 0; j < NL_{k2}.length; j++$ do
3: If $NL_{k1}[i].pre < NL_{k2}[j].pre$ then
4: If $NL_{k2}[j].pre < NL_{k1}[i].pre + NL_{k1}[i].size$ then
5: Thêm $NL_{k2}[j]$ vào NL_{k1+1};
6: End if
7: else
8: If $NL_{k1}[i].pre < NL_{k2}[j].pre + NL_{k2}[j].size$ then
9: Thêm $NL_{k1}[i]$ vào NL_{k1+1};
10: End if
11: End if
12: End for
13: End for
End.

2.3.3.4 Thuật toán NPSFF

Như đề cập ở trên, cấu trúc dữ liệu PP-code trong thuật toán NFFP được cải tiến trong thuật toán được đề xuất NPSFF sử dụng cấu trúc FPOSC. Thuật toán NPSFF được sử dụng để khai phá tất cả các mục mờ phổ biến sử dụng Node-list và POS-code. Giống như thuật toán NFFP, NPSFF cũng được thực hiện qua bốn pha: (1) xây dựng cây POSC-tree và tìm ra các mục mờ phổ biến với length=1 (F_1); (2) sinh Node-list của F_1 ; (3) xây dựng Node-list của tập mục mờ k-itemsets; (4) tìm tất cả các tập mục phổ biến mờ với NPSFF. Thuật toán NPSFF được thực hiện tương tự với thuật toán NFFP được mô tả như trong thuật toán 2.9.

Thuật toán 2.9: Thuật toán khai phá tập mục mờ phổ biến NPSFF

Input: độ hỗ trợ mờ tối thiểu *fminsup* (δ), tập phổ biến mờ (1-item) (L_1), Nodelist của $L1$ ($NL1$);

Output: Tập mục mờ phổ biến (FFIs)

(1) For $(k = 2; L_{k-1} \neq \emptyset; k++)$ do begin
(2) For each $p = i_1 i_2 \dots i_{k-2} i_x \in L_{k-1}$ and $q = i_1 i_2 \dots i_{k-2} i_y \in L_{k-1}$, do
(3) If $i_x \succ i_y$ then
(4) $l = i_1 i_2 \dots i_{k-2} i_x i_y$
(5) If each k-1 subsets l in L_{k-1} then begin
(6) l.Node-list = POS_Node-list_Intersect (p.Node-list, q.Node-list);
(7) Tính $l.support$; // Sử dụng tính chất 2.4
(8) If $(l.support \geq n \times \delta)$ then begin
(9) $L_k = L_k \cup \{l\}$;
(10) $NL_k = NL_k \cup \{l.Nodelist\}$;
(11) end if
(12) end if
(13) end if
(14) end for
(15) Delete NL_{k-1};
(16) end for
(17) $FFIs = \bigcup_k L_k$

2.4 Thuật toán khai phá luật kết hợp mờ

Trong quá trình tạo luật kết hợp từ các mẫu mờ phổ biến, bất kỳ mục nào trong các mẫu phổ biến được hình thành là tiền đề và kết quả của luật kết hợp mờ. Nếu giá trị độ tin cậy mờ lớn hơn độ tin cậy tối thiểu đã cho, thì luật kết hợp mờ được thiết lập, ngược lại thì không.

Từ kết quả của các bước thực hiện trên, NCS tổng hợp quy trình khai phá luật kết hợp mờ bằng thuật toán MFAR thể hiện ở Thuật toán 2.10.

Thuật toán 2.10: MFAR

Input: CSDL định lượng (D_Q), ngưỡng độ hỗ trợ tối thiểu δ, độ tin cậy tối thiểu minfc

Output: Tất cả các luật kết hợp mờ FRs

Begin

1: Chuyển đổi D_Q sang D_f

2: FPOSC _Tree_ Construction (*Df*, δ) ; // để sinh cây FPOSC (FTr), F_1

3: FNode-list Gen (FTr, F_1);

4: NPSFF (δ, L_1, NL_1); // để tìm ra tất cả mục mờ phổ biến

5: $FRs = \emptyset$;

6: **For each** $X \in FFIs$ **do**

7: **For** each $Y \subset X$ && $Y \neq \phi$ **do**

8: $fr = X \setminus Y \rightarrow Y$;

9: $fc(fr) = \frac{sup(XY)}{sup(X)}$;

10: **If** $fc(fr) \geq mincf$ then

11: $FRs = FRs \cup \{fr\}$;

12: **End if**

13: **End for**

14: **End for**

15: **Return** FRs

Bảng 2.11 Các luật kết hợp mờ trong ví dụ thỏa mãn độ tin cậy tối thiểu 80%

Luật kết hợp	**Độ tin cậy**
D.Low → A.Middle	82%
D.Low → C.High	100%
D.Low → A.Middle, C.High	81.82%
A.Middle, C.High → D.Low	100%
C.High, D.Low → A.Middle	81.82%
A.Middle, D.Low → C.High	100%

2.5 Thực nghiệm

Trong thử nghiệm, NCS sử dụng tập dữ liệu thu được từ tập dữ liệu để khai phá tập mục phổ biến được gọi là Foodmart, Chess và Chain [84]. Mỗi giao dịch trong tập dữ liệu này bao gồm tất cả các mặt hàng mà khách hàng có được trong một lần. Mô tả về tập dữ liệu được trình bày trong bảng 2.12. Để xử lý cơ sở dữ liệu định lượng, NCS đã chỉ định số ngẫu nhiên cho tất cả các mục trong tập dữ liệu này với phân phối trong phạm vi giá trị từ 1 đến 100.

Thuật toán được đề xuất và các thuật toán được so sánh trong luận án này đã được chạy và thử nghiệm trong môi trường lập trình tích hợp IDE (integrated development environment), JDK8 (Java Development Kit) và ngôn ngữ lập trình hướng đối tượng JAVA trên máy tính chạy Windows 10 x64 được trang bị bộ vi xử lý Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz Intel 2,8 GHz và RAM 16 GB. Độ hỗ trợ mờ tối thiểu được thiết lập là 30%.

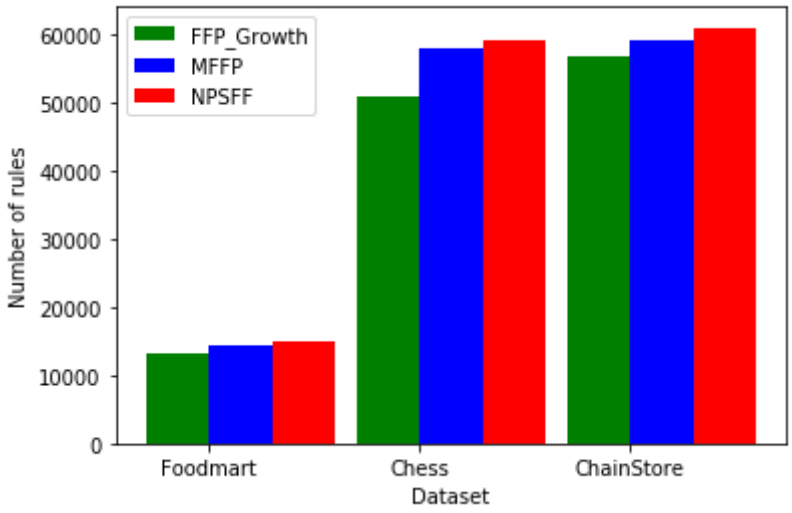
Bảng 2.12: Mô tả tập dữ liệu cho thực nghiệm

Tập dữ liệu	**Số giao dịch**	**Số mục**	**Số lượng mục trung bình trên mỗi giao dịch**
Foodmart	4,141	1,559	4.42
Chess	3,196	75	37
ChainStore	111,294	46,086	7.23

Số đo chỉ ra các luật mờ được sinh ra bởi các thuật toán FFP Growth, MFFP và NPSFF. Kết quả được thể hiện trong Bảng 2.13 và trong Hình 2.12, bảng 2.13 đại diện cho số lượng các luật cho các tập dữ liệu khác nhau. Để đánh giá số lượng luật, FFP Growth tìm ra ít luật hơn trong bộ dữ liệu Foodmart, Chess và ChainStore so với MFFP và NPSFF. (xem trong Hình 2.12).

Bảng 2.13: Số luật kết hợp trong các thuật toán

Dataset	**Number of rules in algorithms**		
	FFP_Growth	**MFFP**	**NPSFF**
Foodmart	13,021	14,321	14,989
Chess	50,846	57,901	59,023
Chain store	56,612	59,111	60,867

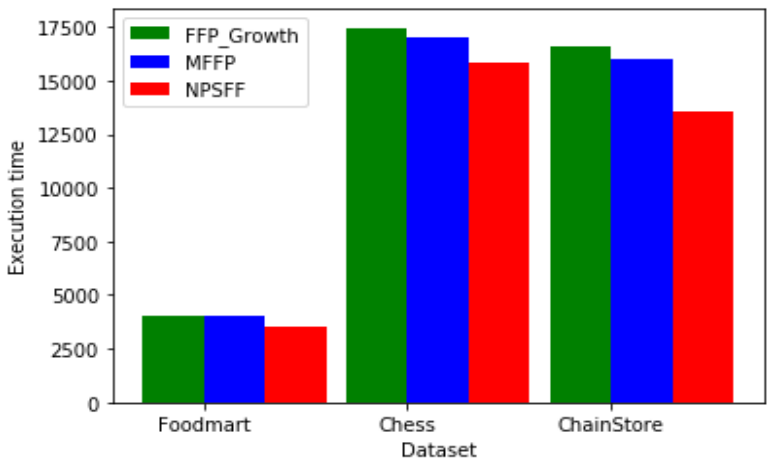


Hình 2.12: Số luật sinh ra từ 3 thuật toán

Thời gian thực hiện NPSFF trong Food ít hơn so với FPP_Growth and MFFP. Thời gian thực thi của bộ dữ liệu Foodmart, Chess và ChainStore cho NPSFF lần lượt là 3503, 14012 và 14712 (xem trong Bảng 2.14). So với FPP_Growth và MFFP cho Foodmart, Chess và ChainStore lần lượt là 2045: 2024, 6135: 6072 và 6544: 6555 (xem trong Hình 2.13).

Bảng 2.14: Thời gian thực thi các thuật toán

Dataset	Execution time in algorithms (ms)		
	FFP_Growth	MFFP	NPSFF
Foodmart	4045	4001	3503
Chess	16810	16004	14012
Chain store	17393	17204	14712

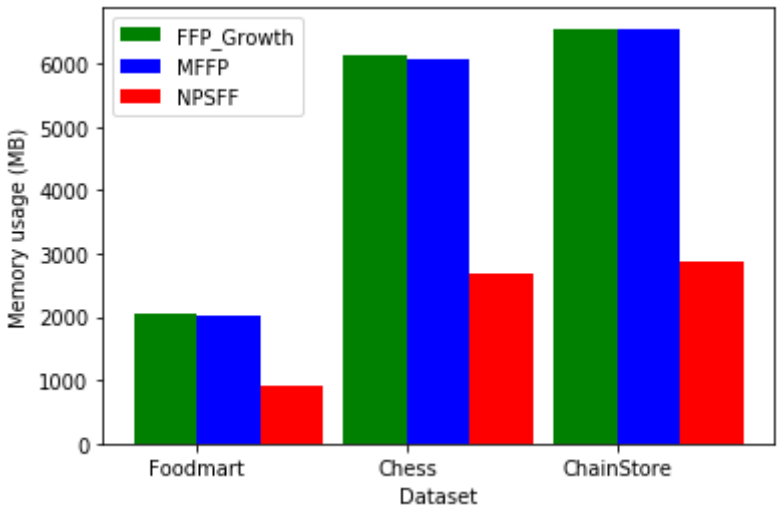


Hình 2.13: Thời gian thực thi của các thuật toán

Theo Bảng 2.15, mức sử dụng bộ nhớ của FFP_Growth và MFFP cao hơn so với NPSFF. Việc sử dụng bộ nhớ của NPSFF cho Foodmart, Chess và Chain store lần lượt là 896, 2688 và 2867 (xem Hình 2.14). Hiệu suất của thuật toán NPSFF cao so với FFP_Growth và MFFP về số lượng luật, thời gian thực thi và sử dụng bộ nhớ. Lý do chính là thuật toán được đề xuất chỉ duyệt qua cơ sở dữ liệu hai lần. Hơn nữa, cây FPOSC được sử dụng để tạo mã POS. Quy trình khai phá các tập phổ biến mờ dựa trên mã POS giúp hạn chế mức tiêu thụ bộ nhớ được yêu cầu.

Bảng 2.15: Bộ nhớ sử dụng trong các thuật toán

Dataset	**Memory usage in algorithms (MB)**		
	FFP_Growth	**MFFP**	**NPSFF**
Foodmart	2045	2024	896
Chess	6135	6072	2688
Chain store	6544	6555	2867



Hình 2.14: Đánh giá bộ nhớ sử dụng của các thuật toán trong các tập dữ liệu khác nhau

Hiệu quả của thuật toán NPSFF được chứng minh là hiệu quả về cả thời gian xử lý và mức tiêu thụ bộ nhớ dựa trên ba đề xuất nêu trên. 1) Áp dụng cấu trúc Node-list để tăng tính linh hoạt cho việc duyệt dữ liệu trong cây và cho phép biểu diễn các giá trị mờ trên đó. 2) Đề xuất thuật toán xây dựng dạng cây FPOSC để tạo mã POS được sử dụng trong việc tìm kiếm các tập hợp mục mờ phổ biến. Cây FPOSC sẽ bị xóa sau khi hoàn thành nhiệm vụ của nó. Do đó, thời gian xử lý và bộ nhớ lưu trữ

giảm đáng kể. 3) Tiến hành thử nghiệm thuật toán NPSFF trên nhiều tập dữ liệu thực và tổng hợp. Về cơ bản, thuật toán NPSFF được đề xuất tốt hơn thuật toán FFP Growth về thời gian thực hiện. Trong các nghiên cứu sắp tới, NCS sẽ thực hiện thuật toán phân cụm AP đã đề xuất ở trên để phân chia không gian mờ cho dữ liệu động theo thời gian và thực hiện khai phá các luật kết hợp mờ trong cơ sở dữ liệu không đầy đủ.

2.6 Kết luận chương 2

Trong chương này, NCS đưa ra giải pháp giải quyết các vấn đề liên quan đến ranh giới "sắc nét" giữa các khoảng mờ bằng cách đề xuất thuật toán phân cụm dữ liệu EMC. Kết quả của thuật toán này [CT4] được sử dụng trong giai đoạn tiền xử lý dữ liệu, phân vùng dữ liêu để chuyển đổi cơ sở dữ liệu định lượng sang cơ sở dữ liệu mờ. Thứ hai, luận án giải quyết vấn đề không gian bộ nhớ bằng cách đưa ra 2 phương pháp khai phá tập mục phổ biến mờ dựa trên cấu trúc dữ liệu Nodelist là NFFP [CT1] và NPSFF [CT2]. NCS đề xuất hai thuật toán NFFP, NPSFF sử dụng cây FPPC_tree, cây POSC-tree để lưu trữ cơ sở dữ liệu định lượng với các giá trị thành viên theo thứ tự giảm dần. Dựa trên cây được xây dựng, Nodelist của từng mục mờ phổ biến được tạo ra. Sau đó, thuật toán NFFP, NPSFF thu được Nodelist của các mục mờ phổ biến (k + 1) bằng cách giao với Nodelist của các mục k mờ phổ biến và sau đó trích xuất các tập tin mờ (k + 1) phổ biến. Ưu điểm của thuật toán này là cây FPPC_Tree cũng như cây POSC-tree được sử dụng để tạo mã FPP_Code hoặc POS-code cho mỗi nút để lấy Nodelist của từng mục mờ phổ biến và sau đó nó sẽ bị xóa để có thể giảm yêu cầu sử dụng bộ nhớ.

Với sự gia tăng kích thước dữ liệu, các loại dữ liệu là không đồng nhất và dữ liệu động. Việc mở rộng các thuật toán khai phá mờ hiệu quả cho kỷ nguyên dữ liệu lớn là một vấn đề quan trọng việc khai phá bằng cách áp dụng các kỹ thuật xử lý song song đã trở thành một cách khả thi để khắc phục vấn đề thời gian xử lý. Vấn đề này được trình bày ở chương 3.

Chương 3 KHAI PHÁ TẬP MỤC PHỔ BIẾN MỜ SỬ DỤNG KỸ THUẬT XỬ LÝ SONG SONG

Trong chương này, NCS trình bày kỹ thuật xử lý song song để khai phá các tập mục mờ phổ biến nhằm giải quyết vấn đề thời gian tính toán bằng cách sử dụng phương pháp tiếp cận tự động học di động (Cellular Learning Automata). Theo CLA, không gian được biểu diễn như một mạng, với mỗi phần tử là một ô, từng dòng một, dữ liệu giao dịch sẽ được đọc và đồng thời được chuyển đến các ô, chúng sẽ cộng tác với nhau song song. Với việc không sử dụng quy tắc vùng lân cận, một loại tự động dữ liệu được gọi là tự động học di động bất thường (ICLA) được sử dụng để tạo danh sách vùng lân cận cho mỗi ô. Thông qua việc sử dụng các automata dữ liệu di động này, việc khai thác tập phổ biến mờ được thực hiện. Quá trình này rút ngắn thời gian thực thi của thuật toán.

3.1 Giới thiệu

Trong những năm gần đây, nhiều thuật toán đã được phát triển để nghiên cứu các vấn đề về khai phá song song cho các luật kết hợp, phân loại, phân cụm và các tác vụ khác. Agrawal và cộng sự. đã đề xuất thuật toán khai phá song song đầu tiên luật kết hợp [85]–[88], trong khi Wang đã nghiên cứu các thuật toán khai phá luật kết hợp song song khác [89]–[91]. Trong số các kiến trúc song song, kiến trúc chủ-tớ (master-slave) thường được sử dụng. Hướng tiếp cận này mang lại những lợi ích đáng kể về hiệu suất [92]. Bộ xử lý chính phân bổ các nhiệm vụ cho các bộ xử lý phụ và thu thập kết quả từ chúng. Trong nghiên cứu [93] việc đánh giá độ chính xác trong khai phá dữ liệu mờ di truyền thường rất tốn thời gian. Trong bài báo [94] bằng cách sử dụng kiến trúc song song master-slave để điều chỉnh động các hàm thành viên và sử dụng chúng để xử lý các giao dịch định lượng trong khai thác dữ liệu mờ. Việc thiết kế một thuật toán khai phá mờ GA song song dựa trên kiến trúc chủ-tớ là rất tự nhiên và phù hợp. Bộ xử lý chính sử dụng một tập hợp đơn lẻ như một thuật toán di truyền đơn giản, và phân phối các nhiệm vụ đánh giá tính phù hợp của các chức năng thành viên và tập phổ biến lớn đối với bộ xử lý phụ. Các quá trình tiến hóa, chẳng hạn như trao đổi chéo, đột biến và sản xuất được thực hiện bởi bộ xử lý chính. Ngoài

ra, nghiên cứu [95] trình bày một thuật toán song song có thể mở rộng để khai thác các luật kết hợp mờ phù hợp với các tập dữ liệu dày đặc.

Trong nghiên cứu [96], tối ưu hóa đám hạt song song có điều khiển mờ (FCP-PSO) được sử dụng để giải các bài toán điều phối kinh tế không lồi lớn. Jin và cộng sự. đã giải quyết vấn đề phủ sóng trong các trò chơi truy đuổi bằng cách sử dụng PPSO dựa trên giao diện truyền thông báo (MPI) [97]. Andrew và cộng sự. đề xuất một PSO song song dựa trên thu nhỏ bản đồ để đánh giá khối lượng thông tin [98]. Trong lĩnh vực trích xuất luật kết hợp và PSO, các nhà nghiên cứu đã đề xuất nhiều thuật toán tính toán song song. Đối với một lượng lớn dữ liệu thử nghiệm, một thuật toán PSO song song được áp dụng để trích xuất luật kết hợp là một giải pháp khả thi.

Ngoài ra, thuật toán iMFFP [99] đề xuất tích hợp các cây MFFP [42] khác nhau từ các cơ sở dữ liệu nhánh và được tích hợp vào cây iMFFP theo trình tự. Sau đó, Header_table được tạo ra và quá trình khai phá tập mục phổ biến được thực hiện. Với phương pháp này, việc tính toán độ hỗ trợ mờ của các mục mờ sẽ không chính xác, không đầy đủ thông tin do cơ sở dữ liệu bị phân rã. Hơn nữa, việc xây dựng từng nhánh cây MFFP và tích hợp dần vào cây iMFFP hoàn chỉnh sẽ gây tốn không gian bộ nhớ.

Trong chương này, NCS trình bày một phương pháp xử lý song song để khai phá các tập mục mờ phổ biến, một giai đoạn quan trọng trong khai phá luật kết hợp mờ bằng cách sử dụng phương pháp tiếp cận tự động học di động (Cellular Learning Automata). Trong chiến lược này, cơ sở dữ liệu định lượng ban đầu được chuyển thành cơ sở dữ liệu mờ trong bước tiền xử lý. Sau khi trích xuất tập phổ biến mờ 1-item từ tập dữ liệu, các mục mờ không phổ biến sẽ bị loại bỏ. Môi trường CA sẽ bắt đầu hoạt động sau giai đoạn tiền xử lý và tạo các ô CA khớp với từng mục mờ phổ biến 1-item. Mỗi dòng dữ liệu trong cơ sở dữ liệu nén được đọc và gửi đến các ô đồng thời, sau đó chúng hoạt động song song. Trong phương pháp này, chế độ LRI (Linear Reward Inaction) được sử dụng. Với mô hình này sẽ không có hình phạt và không có quy tắc cho vùng lân cận giữa các ô. ICLA đề cập đến cụm ô được tạo bởi các giao dịch tập dữ liệu. Danh sách lân cận cho các láng giềng được duy trì bởi các ô và được cập nhật mỗi khi nhận được giao dịch từ môi trường. Ở bước cuối cùng, mỗi ô kiểm tra danh sách các vùng lân cận của nó và loại bỏ các vùng lân cận có mức hỗ trợ thấp hơn ngưỡng tối thiểu của người dùng. Sau đó, mỗi ô thu thập các mục mờ thường

xuyên được sắp xếp theo danh sách lân cận, gửi chúng vào môi trường, v.v. Với việc sử dụng danh sách vùng lân cận trong CLA giúp hạn chế mức tiêu thụ bộ nhớ được yêu cầu.

3.2 Một số khái niệm liên quan về automata di động học (Cellular learning automata)

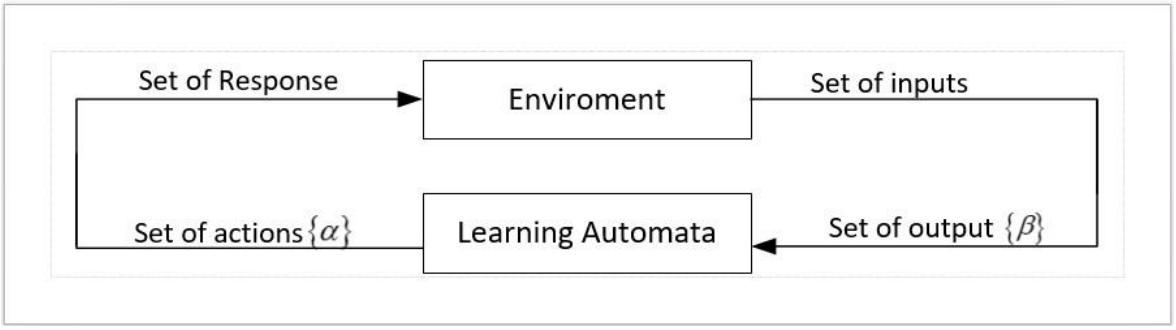
3.2.1 Automata học LA (Learning Automata)

Khái niệm về automat học được Tsetlin đề xuất lần đầu tiên vào năm 1973 [100]. Các nghiên cứu khác được đưa ra liên quan đến các vấn đề này là nhằm tìm ra một hành động tối ưu giữa các hành động được phép trong tự động ngẫu nhiên. Các ứng dụng của LA được sử dụng trong nhận dạng mẫu, phân vùng đồ thị và lập kế hoạch đường đi [101]. Automat học có thể được hình dung như một đối tượng với các nhóm hành động hạn chế. LA chọn ngẫu nhiên một trong các hành động trong số các hành động của nó và gửi nó đến môi trường. Các LA có thể cập nhật hành động xác suất của nó dựa trên các phản hồi của môi trường và quy trình được lặp lại [102]. Một LA bao gồm hai phần:

1. Một automata ngẫu nhiên với số lượng hành động hạn chế và một môi trường ngẫu nhiên.

2. Thuật toán học: thuật toán mà automata sẽ học hành động tối ưu bằng cách sử dụng hành động đó.

Mỗi hành động được chọn bởi môi trường tiềm năng sẽ được đánh giá và câu trả lời được đưa ra cho một dữ liệu tự động học. LA sẽ sử dụng câu trả lời này và chọn hành động của nó cho giai đoạn tiếp theo. Hình 3.1 cho thấy mối quan hệ giữa dữ liệu tự động học và môi trường [101].



Hình 3.1: Môi trường, LA và mối quan hệ giữa chúng

3.2.1.1 Môi trường

Công thức của môi trường (Environment) được đưa ra từ công thức: $E = \alpha, \beta, c$ trong đó $\alpha = \alpha_1, \alpha_2, \dots, \alpha_m$ là tập các giá trị đầu vào, $\beta = \beta_1, \beta_2, \dots, \beta_m$ là tập các giá trị đầu ra và $c = c_1, c_2, \dots, c_m$ là tập các xác suất phạt. [15]

Ngoài ra, công thức của môi trường còn có thể chia thành ba mô hình:

- Khi β có hai giá trị, môi trường được gọi là P-model. $\beta = 1$ khi bị phạt, và . $\beta = 0$ khi được thưởng.
- Khi môi trường là Q-model, $\beta(n)$ là tập giá trị đầu ra hữu hạn với hơn hai giá trị thuộc $[0,1]$.
- Khi môi trường là S-model, $\beta(n)$ là các biến ngẫu nhiên liên tục trong $[0,1]$.

c_i là xác suất của một câu trả lời không mong muốn. Trong môi trường tĩnh, các giá trị c_i không thay đổi; trong khi trong môi trường không tĩnh, các giá trị này thay đổi theo thời gian. Automata học được phân loại thành hai nhóm với các cấu trúc thay đổi hoặc cấu trúc được sắp xếp.

3.2.1.2 Automata học ngẫu nhiên

Automata học ngẫu nhiên được hiển thị với bộ $\{\alpha, \beta, F, G, \varphi\}$ trong đó $\alpha = \alpha_1, \alpha_2, \dots, \alpha_m$ là tập các hành động của automata, $\beta = \beta_1, \beta_2, \dots, \beta_m$ là tập các đầu vào Automata , F là hàm sản xuất và trạng thái mới của Automata. G là một hàm đầu ra ánh xạ trạng thái hiện tại và nhập vào đầu ra hiện tại. φ là tập hợp các trạng thái bên trong của automata. Tập hợp α chứa hành động Automata và Automata chọn một hành động giữa hành động của nó và gửi nó đến môi trường. β xác định tập hợp các đầu vào Automata. F và G ánh xạ trạng thái hiện tại của đầu vào Automata vào đầu ra tiếp theo. Automata học ngẫu nhiên được chia thành hai nhóm, Automata có cấu trúc cố định và Automata có cấu trúc biến đổi. Trong cấu trúc cố định, automata là xác suất xác định của hành động Automata; nhưng trong cấu trúc biến đổi, Automata là xác suất cập nhật của hành động Automata và mỗi lần lặp lại dựa trên phản ứng của môi trường mà không có độ trễ cập nhật được thực hiện bằng thuật toán Learning.

3.2.1.3 Automata học ngẫu nhiên có cấu trúc thay đổi

Tự động học có cấu trúc biến đổi được hiển thị với bộ bốn α, β, P, T trong đó $\alpha = \alpha_1, \alpha_2, \dots, \alpha_m$ là một tập các hành động tự động hóa, $\beta = \beta_1, \beta_2, \dots, \beta_m$ là một

tập hợp đầu vào tự động dữ liệu, $p = p_1, p_2, \dots, p_m$ là chuỗi xác suất để lựa chọn từng hành động và thuật toán học có thể được hiển thị như sau:

$$P(n+1) = T[\alpha(n), \beta(n), p(n)] \quad (3.1)$$

3.2.1.4 Mô hình học P-model

Trong hoạt động đầu tiên, automata (r là số lượng hành động tự động), tất cả xác suất của các hành động trong bất kỳ dạng tự động sẽ bằng nhau. Đối với một tự động hóa hành động r, xác suất của hành động được cho bởi $p_i(n) = 1/r$ mà trong mỗi lần lặp lại sẽ thay đổi và cập nhật giá trị của xác suất này (dựa trên phần thưởng hoặc hình phạt). Trong loại Automata này, nếu hành động α_i được chọn giữa các hành động khác trong giai đoạn này thì $P_i(\alpha_i)$ nhận được câu trả lời mong muốn và xác suất của nó sẽ tăng lên và xác suất ngược lại sẽ giảm. Tuy nhiên, đối với các câu trả lời không mong muốn, khi xác suất của Pi (n) giảm thì các xác suất còn lại sẽ tăng lên. Tuy nhiên, các thay đổi được thực hiện sao cho tổng $p_i(n) = 1$. Giả định này đã được chỉ ra bên dưới cho cả câu trả lời mong muốn và công thức câu trả lời không mong muốn của thuật toán học trong dữ liệu tự động học biến trong P-Model [19].

Với câu trả lời mong muốn

$$\begin{aligned} P_i(n+1) &= P(n) + a[1 - P_i(n)] \\ P_j(n+1) &= (1-a)P_j(n) \quad \forall j, j \neq i \end{aligned} \quad (3.2)$$

Với câu trả lời không mong muốn

$$\begin{aligned} P_i(n+1) &= (1-b)P_i(n) \\ P_j(n+1) &= (b/r - 1) + (1-b)P_j(n) \quad \forall j, j \neq i \end{aligned}$$

Trong đó, a là biến thưởng, b là biến phạt và r là số các hành động.

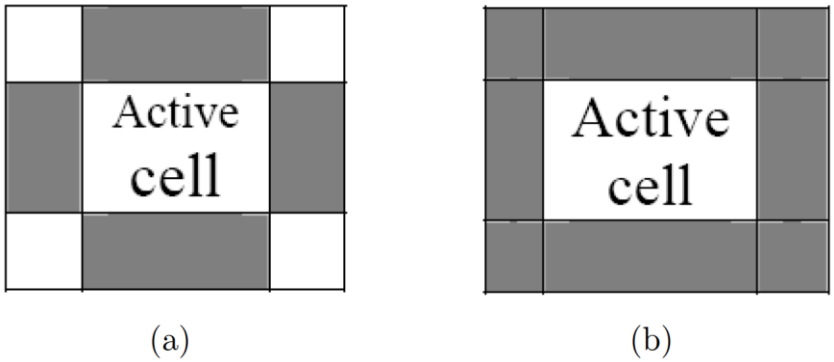
Ba trạng thái được xem xét theo giá trị của a và b là:

- L_{RP} (Linear Reward Penalty): khi $a = b$
- $L_{R\&P}$ (Linear Reward Epsilon Penalty): khi b nhỏ hơn a ($a \gg b$)
- L_{RI} (Linear Reward Inaction) when $b = 0$

3.2.2 Automata di động (CA – Cellular Automata)

Automata tự động CA là một mô hình toán học cho hệ thống trong đó số lượng các thành phần đơn giản hợp tác để tạo ra các mẫu phức tạp. Thực tế, một CA là một hệ thống động lực học rời rạc và thông tin liên lạc giữa các ô của nó bị hạn chế dựa

trên sự tương tác cục bộ. CA bao gồm mạng tinh thể của các ô và tập hợp các luật [35]. Mỗi ô vuông được gọi là ô với mỗi ô có hai trạng thái là màu đen và màu trắng. Các quy tắc của automata di động xác định cách các trạng thái thay đổi.



Hình 3.2: Mô hình láng giềng theo Moore và Von Neumann

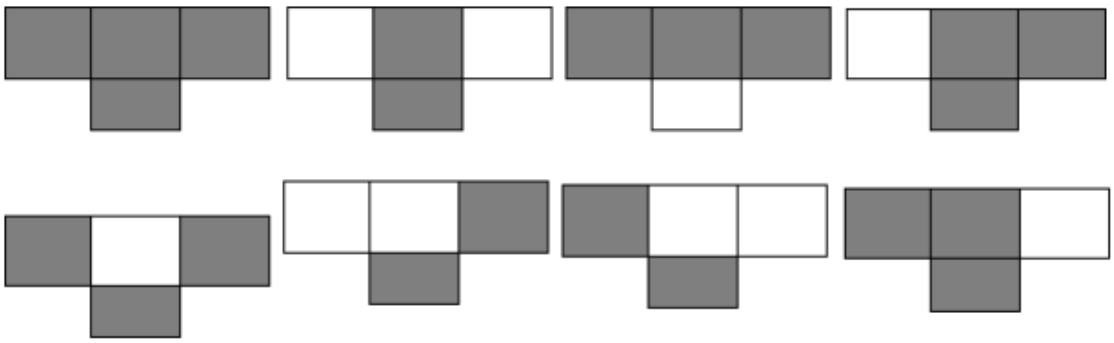
Đối với mỗi ô, có một tập hợp các ô xung quanh ô đang hoạt động được gọi là láng giềng. Trong Hình 3.2 (a) "láng giềng" của một ô là tám ô vuông được tương tác và trong Hình 3.2 (b), nó được khai báo là láng giềng = (trên, dưới, trái, phải). Automata di động dựa trên hành vi của láng giềng và các hành động trong quá khứ. Láng giềng được định nghĩa là một tập hợp các ô xung quanh ô đang hoạt động với khoảng cách là hai hoặc nhỏ hơn hai. CA dựa trên một số quy tắc cục bộ và các quy tắc của Automata thường có thể được thiết kế bởi người dùng. Các ô riêng lẻ được tạo ra bởi các quy tắc của láng giềng cục bộ của nó. Trạng thái mới được tạo ra theo các quy tắc địa phương này và các tiểu bang láng giềng của nó. Một Automaton được cập nhật dựa trên các quy tắc cục bộ. Giá trị của láng giềng sẽ tác động đến việc xác định trạng thái mới của mỗi ô [38]. Về mặt hình thức, một CA có thể được định nghĩa như sau:

Một automata di động d-chiều là một cấu trúc $A = (Z^d, \Phi, N, F)$ trong đó:

- Z^d là một mạng lưới d-tuples các số nguyên mà các mạng này có thể là vô hạn, hữu hạn hoặc bán hữu hạn.
- $\Phi = \{1, \dots, m\}$ là một tập hữu hạn các đỉnh
- $N = \{x_1, x_2, \dots, x_m\}$ là tập con hữu hạn Z^d gọi là vector láng giềng ($x_i \epsilon\ Z^d$).
- F là quy tắc cục bộ của automata di động. Quy tắc này có thể được xác định bởi người dùng. Quy tắc các automata di động xác định các trạng thái thay đổi và cách tập hợp các ô làm láng giềng của nhau.

❖ Automata di động là phương pháp học tăng cường có các đặc điểm sau:

- Các automata di động là các không gian rời rạc,
- Thời gian thực hiện rời rạc
- Mỗi ô bao gồm một số trạng thái giới hạn
- Tất cả các ô đều ở cùng vị trí
- Tất cả các ô được cập nhật cùng một lúc
- Quy tắc của mỗi ô phụ thuộc vào giá trị của các bên xung quanh nó
- Quy tắc cho giá trị mới của mỗi bên chỉ phụ thuộc vào giá trị của số lượng giới hạn của các trạng thái trước đó.



Hình 3.3: Quy tắc tạo các ô

3.2.3 Automata di động học – Cellular learning automata

Automata di động học là sự kết hợp của hai mô hình LA và CA. Ý tưởng chính của CLA là một lớp con của CA ngẫu nhiên sử dụng LA để điều chỉnh xác suất chuyển đổi trạng thái của CA ngẫu nhiên. CLA là một CA trong đó LA là một hoặc nhiều hơn một được chỉ định cho mọi ô. Các quy tắc và các hành động được lựa chọn bởi LA của láng giềng xác định tín hiệu tăng cường đến LA nằm trong mỗi ô. Theo các quy tắc đó và các trạng thái trước đó của các ô láng giềng, giao dịch mới của tập dữ liệu sẽ mang lại điểm thưởng hoặc hình phạt. Việc đưa ra hình phạt hoặc điểm thưởng sẽ cập nhật cấu trúc CLA. CLA chính thức được định nghĩa như sau.

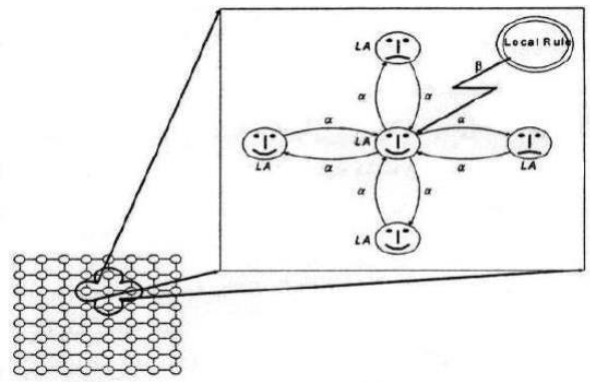
Một CLA đa chiều có cấu trúc: $A = (Z^d, \Phi, A, N, F)$

- Z^d là một mạng lưới d-tuples các số nguyên mà các mạng này có thể là vô hạn, hữu hạn hoặc bán hữu hạn.
- $\Phi = \{1, \dots, m\}$ là một tập hữu hạn các đỉnh
- A là tập hợp các tự động học (LA), mỗi ô trong số đó được gán cho một ô của CLA. Mỗi ô có thể có một hoặc nhiều hơn một LA.

- $N = \{x_1, x_2, \dots, x_m\}$ là tập con hữu hạn Z^d gọi là vector láng giềng ($x_i \epsilon\ Z^d$).

F là quy tắc cục bộ của automata di động.

Mỗi LA trong CLA chọn một hành động giữa các hành động của nó. Việc lựa chọn hành động có thể được lựa chọn dựa trên quan sát ngẫu nhiên hoặc tình cờ. Hành động được chọn gây ra chuyển động của LA từ ô này sang ô khác. Quy tắc của CA xác định tín hiệu tăng cường cho mỗi LA nằm trong ô đó như trong Hình 3.4. Các hành động của LA trong mỗi ô hoạt động sẽ nhận được phần thưởng hoặc hình phạt dựa trên LA hiện tại của các hành động trong LA và các quy tắc láng giềng. Các điểm thưởng hoặc phạt cập nhật cấu trúc nội bộ của LA. Điểm thưởng hoặc hình phạt tiếp tục cho đến khi hệ thống chuyển sang trạng thái duy trì. Cuối cùng, mỗi automata di động học cập nhật hành động của véc tơ xác suất dựa trên tín hiệu tăng cường được cung cấp và hành động được chọn. Quá trình này tiếp tục cho đến khi thu được kết quả mong muốn.



Hình 3.4: Automata di động học

3.2.3.1 Automata di động học có quy tắc

Một Automata di động học được gọi là chính quy nếu tất cả các ô (ô đang hoạt động và các ô lân cận của nó) có thứ tự theo quy tắc. Quy tắc thông thường có nghĩa là, thứ tự một chiều hoặc hai chiều được sử dụng bởi Von Neumann và Moore. Cho đến nay, CLA có quy tắc đã sử dụng các ứng dụng khác nhau.

3.2.3.2 Automata di động học bất quy tắc

Một Automata di động học bất quy tắc ICLA (Irregular learning automata) không có giới hạn về ô mạng. ICLA là một đồ thị vô hướng, trong đó mỗi đỉnh hiển thị một ô được chứa với dữ liệu tự động học. LA xác định hành động của nó dựa trên hành động của véc tơ xác suất. Giống như CLA, có một quy tắc mà ICLA hoạt động

theo quy tắc đó. Quy tắc và hành động được chọn bằng cách học Automata của các láng giềng. LA sẽ gửi một tín hiệu tăng cường cho một LA nằm trong một ô đang hoạt động. ICLA chính thức được định nghĩa dưới đây:

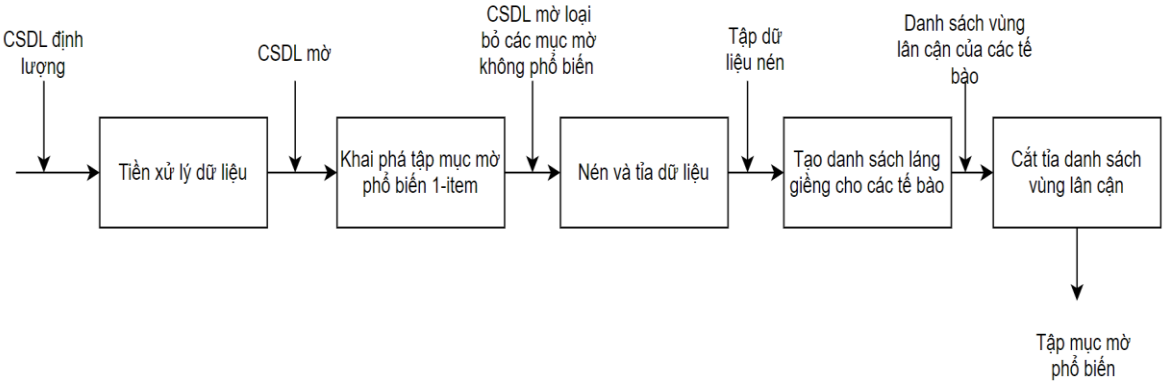
Một automata di động học CLA đa chiều là một cấu trúc $A = (G, E, \langle V \rangle^d, \emptyset, A, F)$

- G là đồ thị vô hướng trong đó V là tập các đỉnh và E là tập các cạnh
- $\emptyset$ là tập hữu hạn các trạng thái
- A là tập hợp các LA nằm trong mỗi ô CLA
- F là tập các quy tắc cục bộ của ICLA trong mỗi đỉnh

3.3 Thuật toán khai phá tập mục phổ biến mờ sử dụng CLA

3.3.1 Ý tưởng thuật toán

Trong thuật toán *CLA-FuzzyMining* [CT3], cơ sở dữ liệu định lượng ban đầu được chuyển thành cơ sở dữ liệu mờ trong bước tiền xử lý. Sau khi trích xuất các tập phổ biến mờ 1-item từ tập dữ liệu, các mục mờ không phổ biến sẽ bị loại bỏ. Môi trường automata di động CA sẽ bắt đầu hoạt động sau giai đoạn tiền xử lý và tạo ra các CA phù hợp với từng mục mờ phổ biến 1-item. Từng dòng dữ liệu trong CSDL nén được đọc và phân phối đồng thời đến các ô, sau đó chúng sẽ hoạt động song song với nhau. Trong phương pháp này, chế độ LRI được sử dụng. Với mô hình này sẽ không sử dụng hình phạt và không có quy định đối với vùng lân cận giữa các ô. ICLA đề cập đến cụm ô được tạo ra bởi các giao dịch tập dữ liệu. Danh sách lân cận cho các láng giềng được duy trì bởi các ô và được cập nhật mỗi khi nhận được giao dịch từ môi trường. Bước cuối cùng, mỗi ô kiểm tra danh sách vùng lân cận của nó và cắt bỏ các vùng lân cận có mức hỗ trợ dưới ngưỡng tối thiểu của người dùng. Sau đó, mỗi ô thu thập các mục mờ phổ biến dựa trên khảo sát được sắp xếp theo danh sách lân cận, gửi chúng vào môi trường.



Hình 3.5: Quy trình thực hiện thuật toán CLA-Fuzzy Mining

Thuật toán CLA-FuzzyMining được thực hiện qua các bước:

Bước 1: Tiền xử lý: Chuyển cơ sở dữ liệu định lượng sang cơ sở dữ liệu mờ.

1.1: Thực thi thuật toán phâm cụm cải tiến EMC

1.2: Thực thi thuật toán phân vùng mờ

Bước 2: Khai phá tập mục phổ biến 1-item

Bước 3: Khai phá tập mục phổ biến k-itemset

3.1: Nén và tỉa dữ liệu

3.2: Duyệt cơ sở dữ liệu nén môi trường sẽ đọc từng bảng ghi của dữ liệu giao dịch.

3.3: Môi trường tạo ra các ô của CLA theo từng mục mờ phổ biến và thực hiện xử lý đồng thời cho các ô đó (chỉ thực hiện đối với bảng ghi 1). Ở bước này nhằm khởi tạo cho tất cả các item có trong giao dịch và các mục này sẽ được tạo tương ứng với mỗi ô, các giá trị ô này có thể là null nếu mục này không tồn tại trong giao dịch đầu tiên.

3.4: Tạo danh sách láng giềng cho các ô. Đọc dữ liệu dòng thứ 2 trong cơ sở dữ liệu nén. Duyệt tất cả các ô đã được xử lý từ bước 3.3 và cập nhật lại độ hỗ trợ mờ các giao dịch.

3.5: Cắt tỉa danh sách vùng lân cận có độ hỗ trợ nhỏ hơn độ hỗ trợ tối thiểu

3.6: Thuật toán kết thúc với kết quả quả là tập phổ biến được lưu trữ tại các ô.

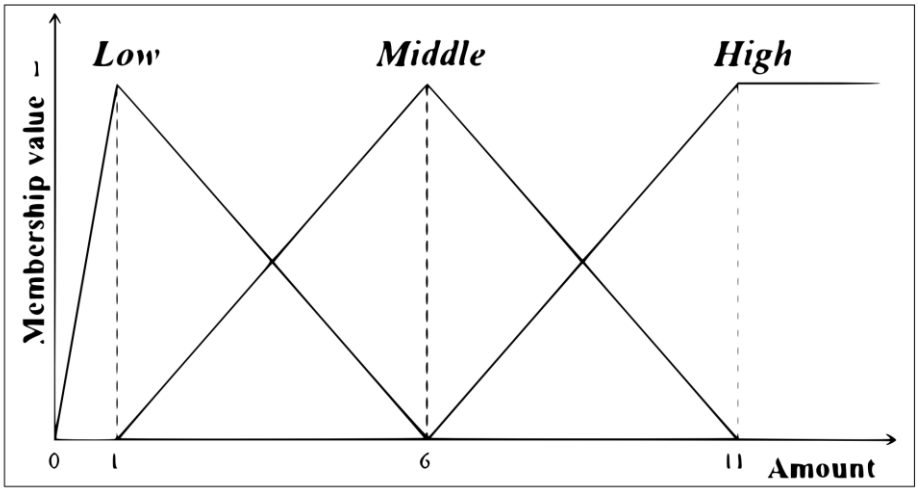
3.3.2 Tiền xử lý dữ liệu

Trong bước này, CSDL được chuyển đổi từ CSDL định lượng sang CSDL mờ. Giả sử có một CSDL định lượng có tám giao dịch và chứa các mục A, B, C, D, E như trong bảng , độ hỗ trợ tối thiểu được thiết lập là 30%.

Bảng 3.1: Bảng CSDL định lượng mẫu

Tid	Items
1	(A: 3) (C:9) (D:3) (E:6)
2	(A: 7) (B:2) (C: 8)
3	(B: 2) (C:9)
4	(A: 8) (C:10) (D:2)
5	(A: 5) (B:2) (C: 7)
6	(A: 5) (C:10) (D:3) (E:2)
7	(A: 3) (B: 3) (C: 9) (E: 6)
8	(C: 9) (D: 3)

Giả sử rằng hàm thành viên là như nhau cho tất cả các mục thể hiện như trong hình 3.5. Giả sử trong ví dụ này các mục có số lượng được thể hiện bởi 3 vùng mờ: {Low}, {Middle} và {High}. Các giá trị thành viên được tính cho mỗi item được xác định dựa trên hàm thành viên trong hình 3.6.



Hình 3.6: Hàm thành viên được sử dụng trong ví dụ

Cơ sở dữ liệu được chuyển đổi từ CSDL định lượng sang CSDL mờ được thể hiện trong bảng 3.2.

Bảng 3.2: Cơ sở dữ liệu mờ được chuyển đổi từ bảng 3.1

Tid	Tập mục mờ
1	$\left(\frac{0.6}{A.Low}, \frac{0.4}{A.Middle}\right)\ \left(\frac{0.4}{C.Middle}, \frac{0.6}{C.High}\right) \left(\frac{0.6}{D.Low}, \frac{0.4}{D.Middle}\right) \left(\frac{0.2}{E.Low}, \frac{0.8}{E.Middle}\right)$
2	$\left(\frac{0.8}{A.Middle}, \frac{0.2}{A.High}\right)\ \left(\frac{0.8}{B.Low}, \frac{0.2}{B.Middle}\right) \left(\frac{0.6}{C.Middle}, \frac{0.4}{C.High}\right)$
3	$\left(\frac{0.8}{B.Low}, \frac{0.2}{B.Middle}\right) \left(\frac{0.4}{C.Middle}, \frac{0.6}{C.High}\right)$
4	$\left(\frac{0.6}{A.Middle}, \frac{0.4}{A.High}\right)\ \left(\frac{0.2}{C.Middle}, \frac{0.8}{C.High}\right) \left(\frac{0.8}{D.Low}, \frac{0.2}{D.Middle}\right)$
5	$\left(\frac{0.2}{A.Low}, \frac{0.8}{A.Middle}\right)\ \left(\frac{0.8}{B.Low}, \frac{0.2}{B.Middle}\right) \left(\frac{0.8}{C.Middle}, \frac{0.2}{C.High}\right)$
6	$\left(\frac{0.2}{A.Low}, \frac{0.8}{A.Middle}\right)\ \left(\frac{0.2}{C.Middle}, \frac{0.8}{C.High}\right) \left(\frac{0.6}{D.Low}, \frac{0.4}{D.Middle}\right) \left(\frac{0.8}{E.Low}, \frac{0.2}{E.Middle}\right)$
7	$\left(\frac{0.6}{A.Low}, \frac{0.4}{A.Middle}\right)\ \left(\frac{0.6}{B.Low}, \frac{0.4}{B.Middle}\right) \left(\frac{0.4}{C.Middle}, \frac{0.6}{C.High}\right) \left(\frac{0.8}{E.Middle}, \frac{0.2}{E.High}\right)$
8	$\left(\frac{0.4}{C.Middle}, \frac{0.6}{C.High}\right) \left(\frac{0.4}{D.Low}, \frac{0.6}{D.Middle}\right)$

3.3.3 Khai phá tập mục phổ biến mờ 1-item

Việc khai phá tập mục phổ biến mờ 1-item được thực hiện như các thuật toán ở chương 2. Độ hỗ trợ mờ của mỗi mục trong giao dịch được tính theo công thức và được kiểm tra với độ hỗ trợ tối thiểu. Độ hỗ trợ của tập mờ được thể hiện trong bảng 3.3. Những mục mờ không thỏa mãn độ hỗ trợ tối thiểu 30% sẽ bị loại khỏi tập dữ liệu. Những mục mờ còn lại thỏa mãn độ hỗ trợ tối thiểu 30% được thể hiện trong bảng 3.4.

Bảng 3.3: Độ hỗ trợ các mục mờ

Các mục mờ	Độ hỗ trợ
A.Low	1,6
A.Middle	3,8
A.High	0,6
B.Low	3
B.Middle	1
B.High	0
C.Low	0
C.Middle	3,4
C.High	4,6
D.Low	2,6
D.Middle	1,4
D.High	0
E.Low	1
E.Middle	1,8
E.High	0,2

Bảng 3.4: Các mục mờ còn lại và độ hỗ trợ của chúng

Các mục mờ	Độ hỗ trợ
C.High	4,6
A.Middle	3,8
C.Middle	3,4
B.Low	3
D.Low	2,6

Trong ví dụ này, NCS đã sử dụng 15 mục mờ và độ hỗ trợ tối thiểu được thiết lập là 30%. Vì vậy, chỉ những mục mờ nào có độ hỗ trợ lớn hơn 2.4 sẽ được giữ lại và loại bỏ các mục mờ có độ hỗ trợ nhỏ hơn 2.4. Kết quả thể hiện trong bảng 3.4, các mục mờ còn lại trong mỗi giao tác được sắp xếp theo thứ tự ở bảng 3.5.

Bảng 3.5: CSDL mờ sau khi loại bỏ các mục mờ không thỏa mãn minsup =30%

Tid	Tập mục mờ	Độ hỗ trợ
1	$\left(\frac{0.6}{C.High}\right)\left(\frac{0.4}{A.Middle}\right)\left(\frac{0.4}{C.Middle}\right)\left(\frac{0.6}{D.Low}\right)$	0.4
2	$\left(\frac{0.4}{C.High}\right)\left(\frac{0.8}{A.Middle}\right)\left(\frac{0.6}{C.Middle}\right)\left(\frac{0.8}{B.Low}\right)$	0.4
3	$\left(\frac{0.6}{C.High}\right)\left(\frac{0.4}{C.Middle}\right)\left(\frac{0.8}{B.Low}\right)$	0.4
4	$\left(\frac{0.8}{C.High}\right)\left(\frac{0.6}{A.Middle}\right)\left(\frac{0.2}{C.Middle}\right)\left(\frac{0.8}{D.Low}\right)$	0.2
5	$\left(\frac{0.2}{C.High}\right)\left(\frac{0.8}{A.Middle}\right)\left(\frac{0.8}{C.Middle}\right)\left(\frac{0.8}{B.Low}\right)$	0.2
6	$\left(\frac{0.8}{C.High}\right)\left(\frac{0.8}{A.Middle}\right)\left(\frac{0.2}{C.Middle}\right)\left(\frac{0.6}{D.Low}\right)$	0.2
7	$\left(\frac{0.6}{C.High}\right)\left(\frac{0.4}{A.Middle}\right)\left(\frac{0.4}{C.Middle}\right)\left(\frac{0.6}{B.Low}\right)$	0.4
8	$\left(\frac{0.6}{C.High}\right)\left(\frac{0.4}{C.Middle}\right)\left(\frac{0.4}{D.Low}\right)$	0.4

3.3.4 Khai phá tập mục phổ biến n-itemset

- **Thực hiện nén dữ liệu**

Khi các mục mờ 1-item phổ biến được trích xuất, để tăng tốc độ hoạt động, tập dữ liệu cần được nén và theo đó các mục dư thừa cần được cắt bỏ. Mục đích của việc cắt tỉa là để tránh công việc vô ích không ảnh hưởng đến kết quả đầu ra và chỉ làm tăng thời gian xử lý. NCS thực hiện nhóm các giao dịch có liên quan lại với nhau để giảm nhu cầu về các thủ tục lặp đi lặp lại. Trước tiên, đọc hàng hiện tại và trích xuất các mục mờ phổ biến 1-item và loại bỏ các mục mờ không thường xuyên khỏi tập dữ liệu, tập dữ liệu nén trước tiên được đặt thành Null.

Khi các mục giao dịch đã được cắt bớt và vẫn còn nhiều hơn 0 mục. Khi đó, độ hỗ trợ của các mục còn lại trong mỗi giao dịch là mức hỗ trợ tối thiểu của các mục; sự hiện diện của nó trong tập dữ liệu được xem xét; nếu đã có, độ hỗ trợ nhỏ nhất trong các mục sẽ được thêm vào, nếu không, nó được tạo như một giao dịch mới với giá trị ban đầu là độ hỗ trợ tối thiểu trong tập dữ liệu nén. Kết quả nén tập dữ liệu được hiển thị trong bảng 3.6.

Bảng 3.6: Tập dữ liệu nén

Tid	**Mục mờ**	**Độ hỗ trợ**
1	(C.High) (A.Middle) (C.Middle) (D.Low)	0.8
2	(C.High) (A.Middle) (C.Middle) (B.Low)	1.0
3	(C.High) (C.Middle) (B.Low)	0.4
4	(C.High) (C.Middle) (D.Low)	0.4

Thuật toán nén dữ liệu được thể hiện trong Thuật toán 3.1

Thuật toán 3.1: Data_Compression()

Input: $minsup$: độ hỗ trợ tối thiểu
F_1: tập mục phổ biến 1-item
D_f: CSDL mờ sau khi loại bỏ các tập mục không phổ biến

Output: $CDS:$ CSDL nén

Begin

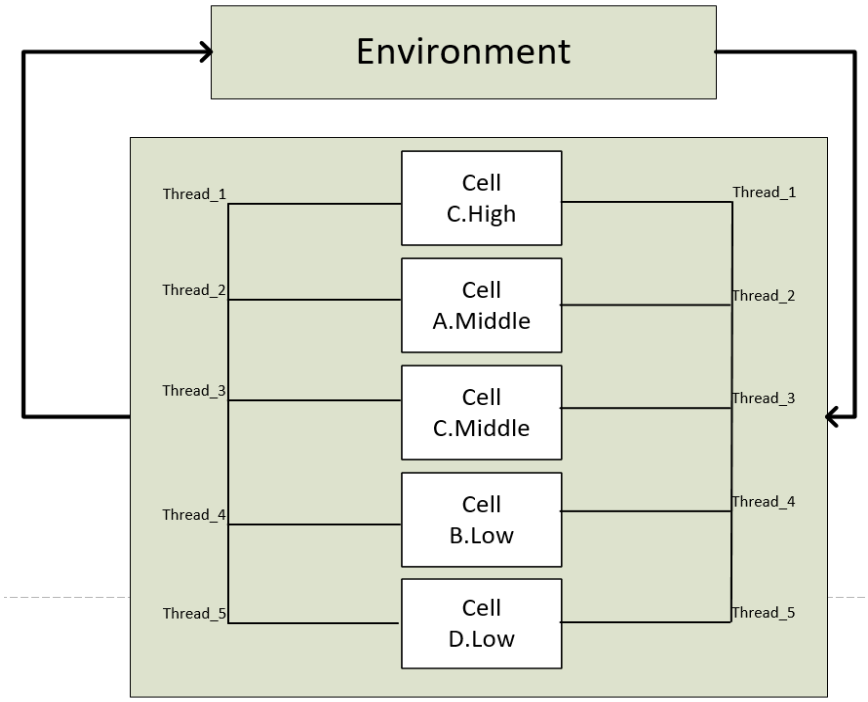
1: **for** $i = 1$ to D_f **do**
2: For $j = 1$ to $items$ **do**
3: **If** $items(i,j) == items(i+1,j)$ **then**
4: Remove (rows (i+1));
5: Update support (rows(i)+ rows(i+1));
6: End if
7: End for
8: End for
9: Return CDS

End.

- **Xác định danh sách láng giềng**

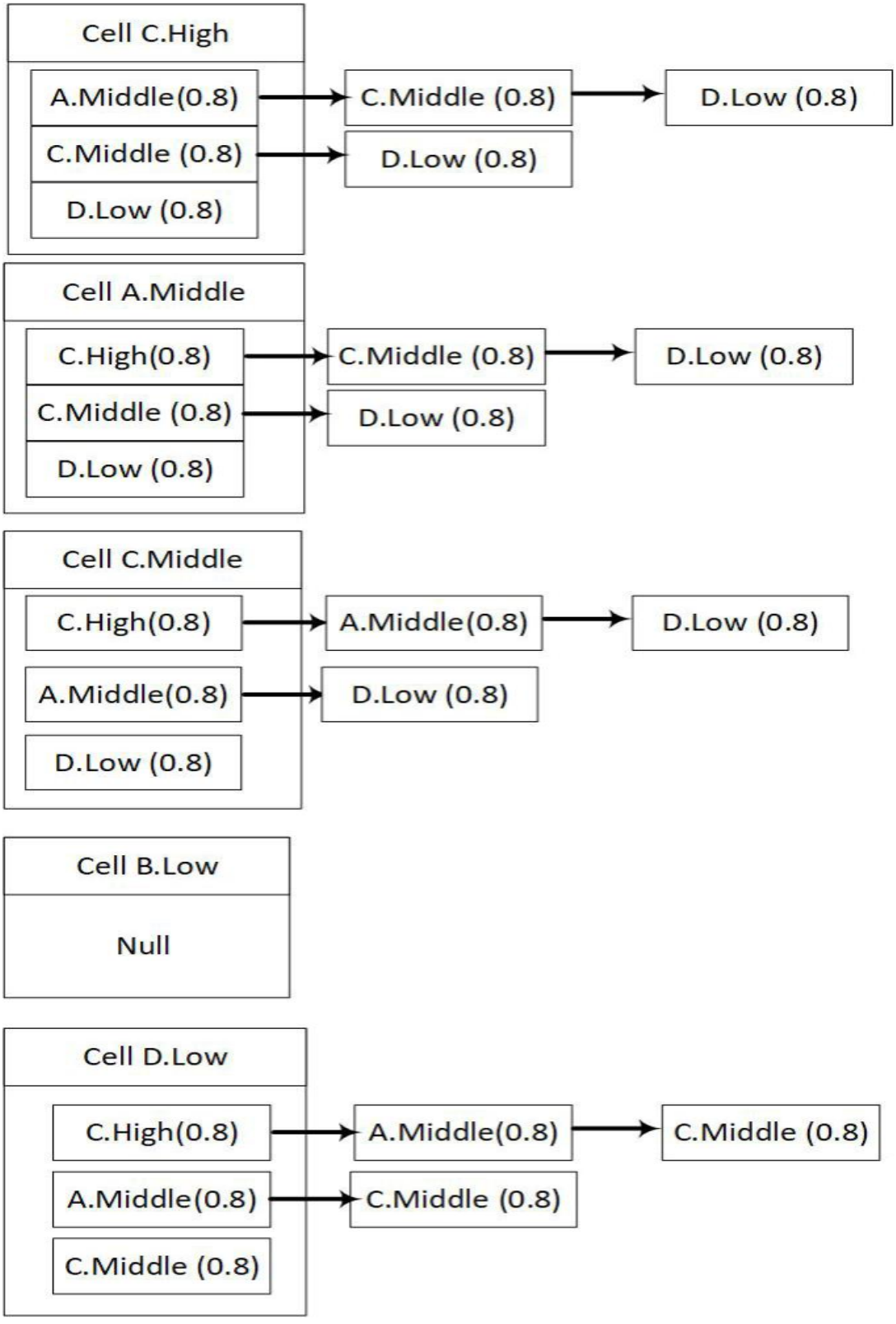
Các ô của automata di động được tạo từ tập mục mờ phổ biến 1-item. Do đó, ở đây có 5 ô, mỗi ô hiển thị một mục mờ phổ biến như trong Hình 3.7. Vì không có quy tắc cụ thể trong vùng lân cận của ô, cấu trúc vùng lân cận thông thường như Von Neumann hoặc Moore không thể được áp dụng. Đây được gọi là automata di động học bất quy tắc (ICLA). Môi trường CA đọc lần lượt tất cả các hàng của tập dữ liệu và sau đó chuyển chúng đến các ô trong mỗi bước. Sau khi nhận được một hàng tập dữ liệu nén, các ô sẽ bắt đầu hoạt động của chúng đồng thời với các ô khác. Các ô

này sẽ cập nhật danh sách vùng lân cận của chúng tùy thuộc vào các mục mờ trong giao dịch đã nhận.



Hình 3.7: Các automata di động học theo tập mục mờ phổ biến 1-item

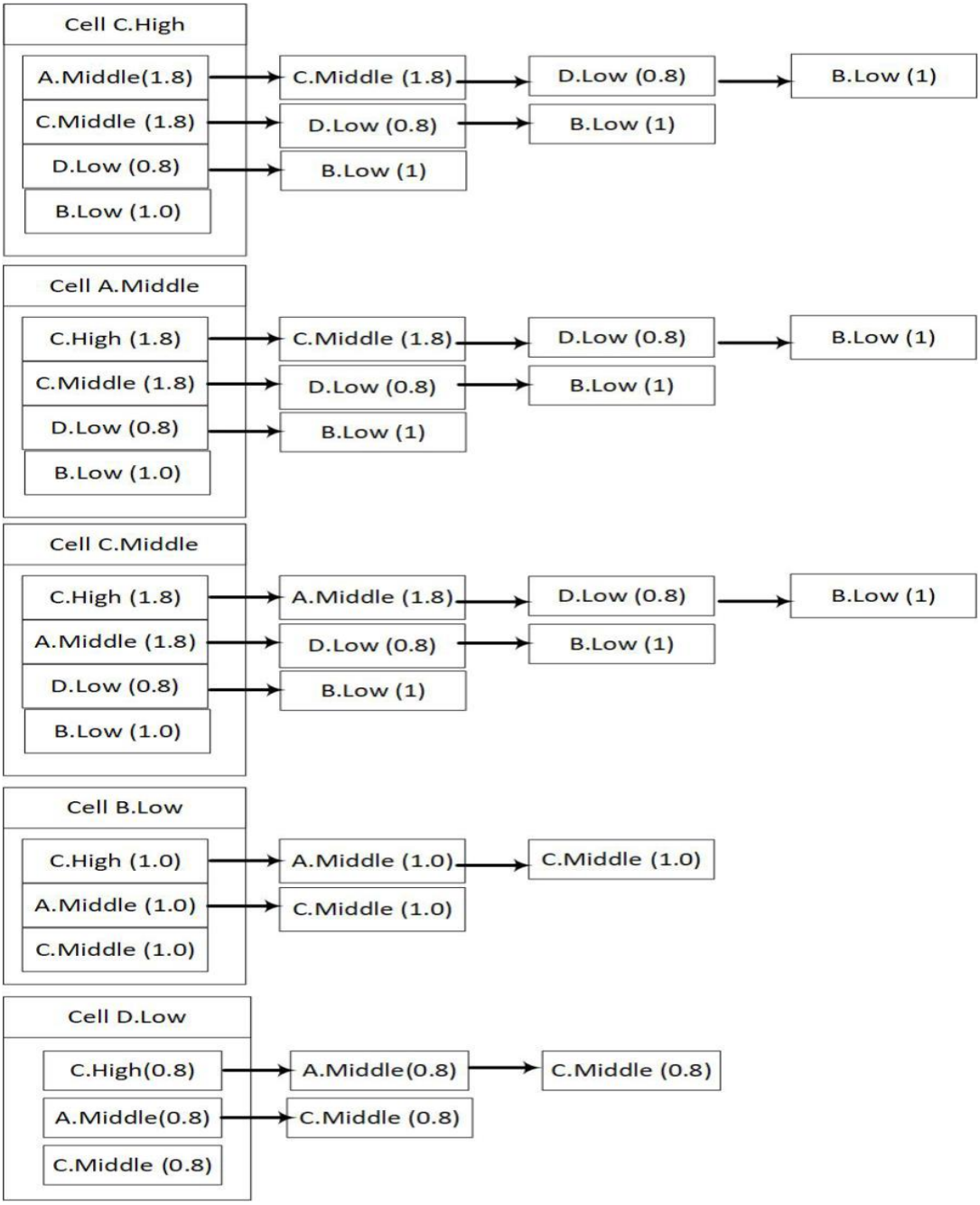
Ví dụ, môi trường đọc hàng đầu tiên chứa {C.High, A.Middle, C.Middle, D.Low} và chuyển chúng đến tất cả các ô, chúng xử lý trên tập phổ biến mờ này. C.High sẽ lần lượt nhận ra A.Middle, C.Middle và D.Low là láng giềng của nó. Nó được tạo với giá trị hỗ trợ mờ là 0,8. Tương tự với các mục A.Middle, C.Middle, D.Low. Bởi vì B.Low không tồn tại trong giao dịch đầu tiên, vì vậy nó không có hành động. Danh sách láng giềng và vùng lân cận của mỗi ô được hiển thị trong hình 3.8.



Hình 3.8: Các ô trong danh sách láng giềng và vùng lân cận của hàng đầu tiên

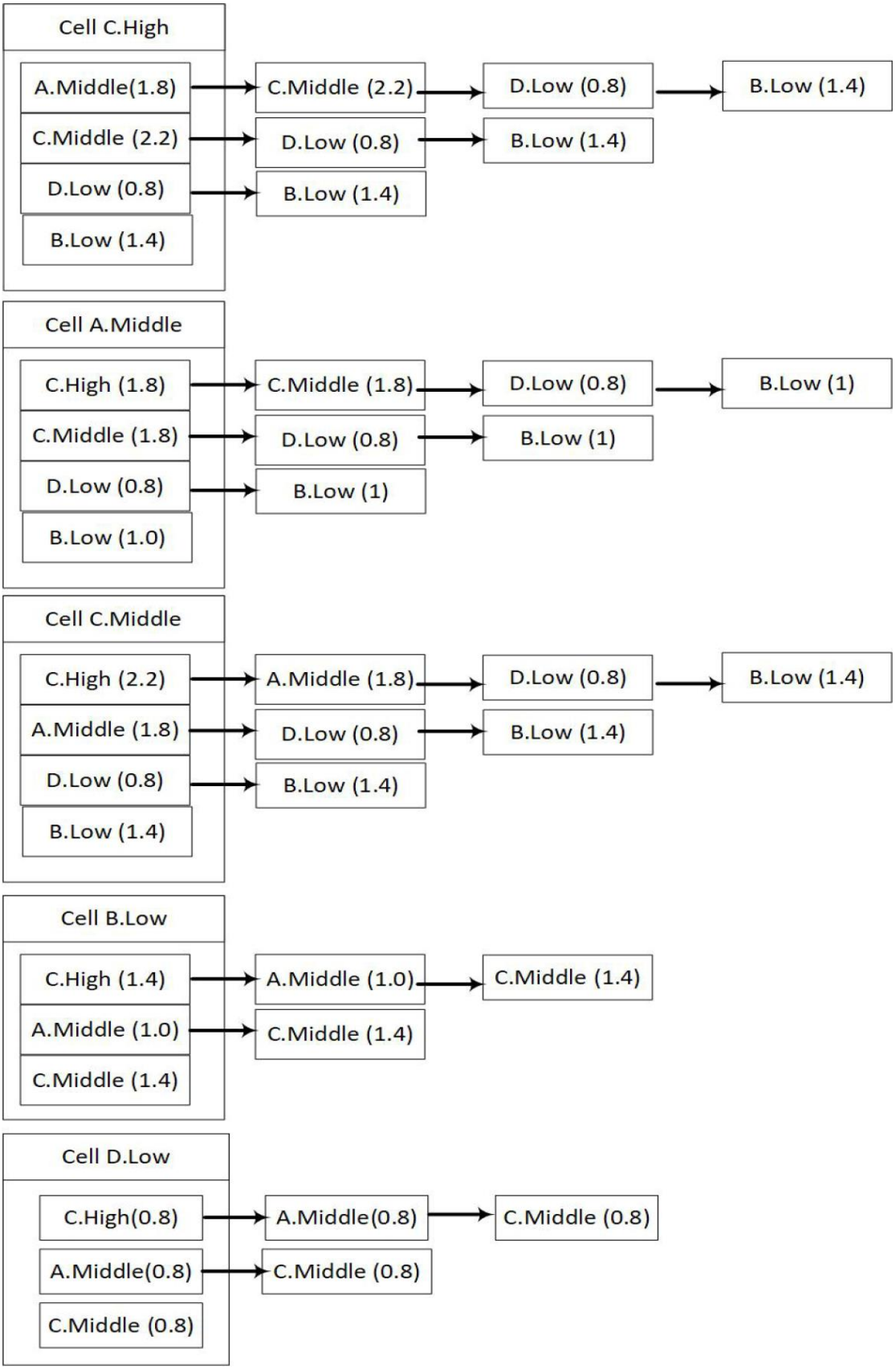
Các ô được đọc với giá trị hỗ trợ là 1,0 và bản ghi thứ hai của tập dữ liệu nén, chứa các giá trị {C.High, A.Middle, C.Middle, D.Low, B.Low} được gửi đến chúng. Đối với ô {B.Low}, các đánh giá của láng giềng về {A.Middle, C.Middle, B.Low} được tính đến. Mục {B.Low} được tạo với giá trị 1,0 vì nó chưa có mặt ở các vùng

lân cận. Các mục hiện có tăng giá trị bằng cách được thêm vào, tăng giá trị độ hỗ trợ của chúng nếu chúng còn tồn tại. Trong trường hợp này, {A.Middle, C.Middle} đã tồn tại trong vùng lân cận của {C.High}, giá trị của {A.Middle} và {C.Middle} là 1,8. Ngoài ra, mục {D.Low} đã có sẵn trong vùng lân cận {C.High} nhưng không có trong hàng thứ 2 của tập dữ liệu, nó vẫn nằm trong vùng lân cận {C.High} với cùng giá trị là 0,8. Các ô khác thực hiện các hành động này đồng thời. Sau khi xử lý bản ghi thứ hai của tập dữ liệu, các vùng lân cận đã tạo và danh sách các ô lân cận được hiển thị trong Hình 3.9.

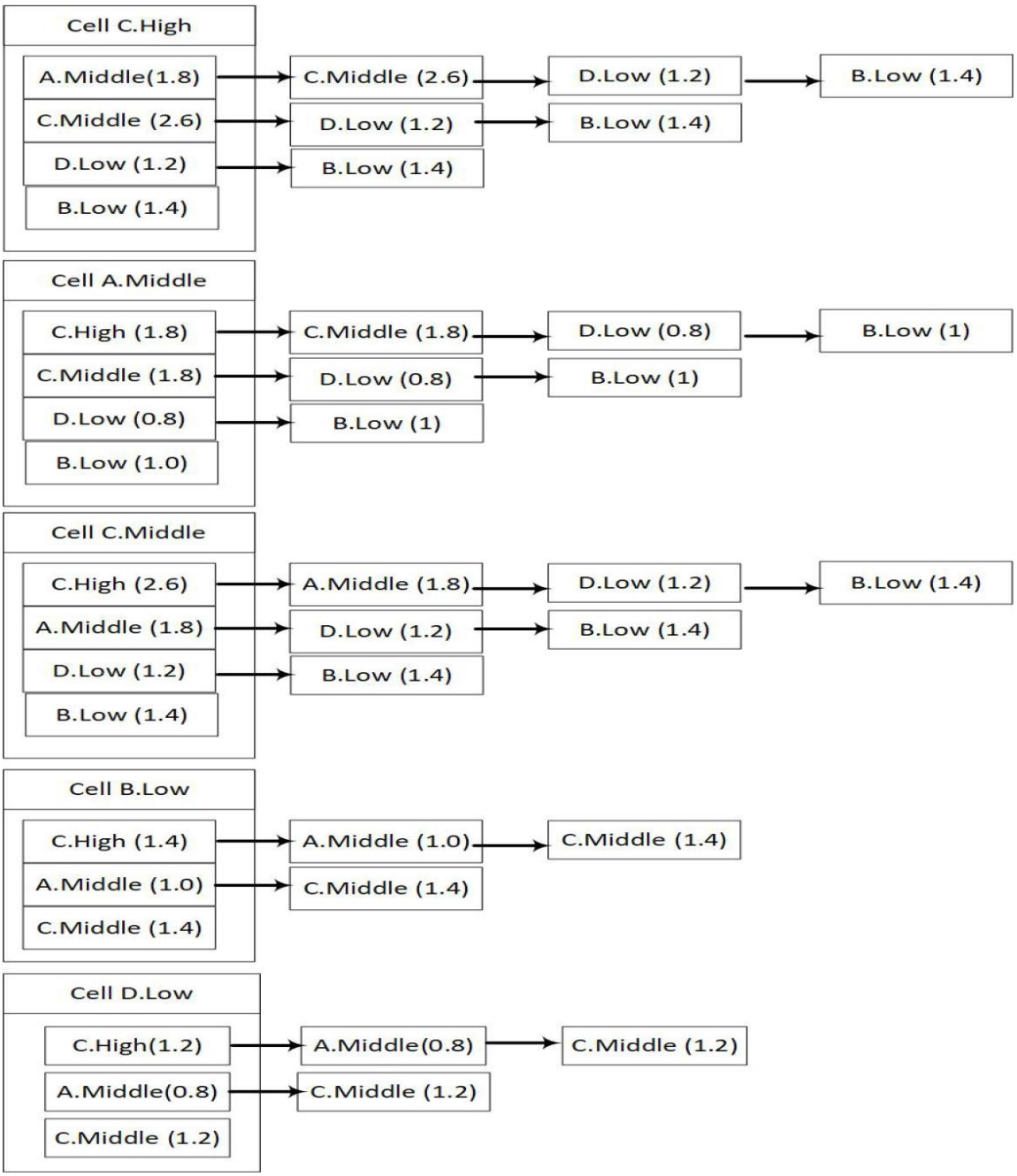


Hình 3.9: Các ô trong danh sách láng giềng và vùng lân cận của hàng thứ 2

Tương tự, hình 3.9, 3.10 hiển thị các vùng lân cận đã tạo và danh sách các ô lân cận sau khi chạy hàng thứ ba và hàng thứ tư của tập dữ liệu.



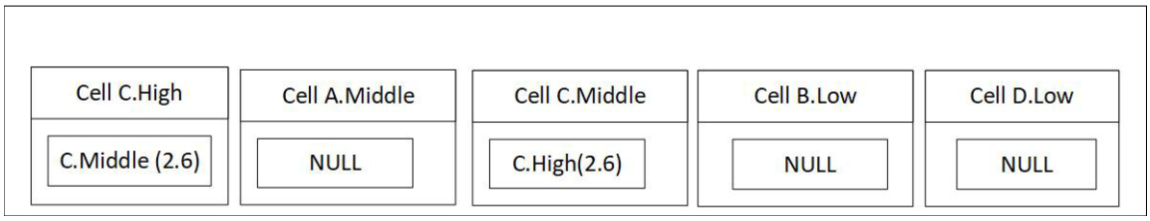
Hình 3.10: Các ô trong danh sách láng giềng và vùng lân cận của hàng thứ 3



Hình 3.11: Các ô trong danh sách láng giềng và vùng lân cận của hàng thứ 4

- **Cắt tỉa danh sách vùng lân cận**

Khi tất cả các giao dịch được môi trường gửi đến các ô, mỗi ô sẽ xóa các láng giềng và vùng lân cận có độ hỗ trợ nhỏ hơn ngưỡng tối thiểu đã được người dùng xác định khỏi danh sách vùng lân cận của nó. Trong ví dụ này, các mục mờ có độ hỗ trợ lớn hơn hoặc bằng 2.4 sẽ được giữ lại, các mục mờ khác bị tỉa khỏi vùng lân cận. Kết quả cắt tỉa được thể hiện trong hình 3.10. Danh sách vùng lân cận lại được sử dụng để quét và cuối cùng thu được tập phổ biến mờ k-Itemset.



Hình 3.12: Vùng lân cận sau khi được tỉa với minsup =30%

Khi môi trường gửi tất cả các mục có sẵn trong tập dữ liệu nén vào các ô quản lý để hoàn thành các tác vụ liên quan, môi trường có xu hướng thu thập và trích xuất các mục mờ phổ biến. Đầu tiên {C.High, A.Middle, C.Middle, B.Low, D.Low} là các mục mờ phổ biến 1-item, và được đặt trong danh sách mục mờ phổ biến. Sau đó, mỗi ô sẽ hoạt động cùng lúc. Ô {C.High} đặt các láng giềng của nó{C.Middle} trong danh sách mục mờ phổ biến, với tổ hợp của chính nó là {C.High, C.Middle} là tập mục mờ phổ biến 2-itemset.

Như đã trình bày ở trên, mỗi ô nhận được một tập phổ biến. Bằng cách thu thập các kết quả của mỗi ô, tất cả các mục mờ phổ biến có thể nhận được. Tuy nhiên, mỗi ô có thể tạo ra các mục giống nhau, thuật toán cần kiểm tra sự hiện diện của các mục mờ trước khi chúng được gửi đến danh sách tổng hợp các mục mờ phổ biến. Nếu các mục này đã có trong danh sách này, chúng sẽ bị loại bỏ; nếu không, các mục này sẽ được đưa vào trong danh sách các mục mờ phổ biến.

3.3.5 Thuật toán CLA-FuzzyMining

Thuật toán CLA-FuzzyMining được mô tả như trong Thuật toán 3.2

Thuật toán 3.2: CLA_Fuzzy_Mining

Input: $minsup$: độ hỗ trợ tối thiểu
F_1: tập mục phổ biến 1-item
D_f: CSDL mờ sau khi loại bỏ các tập mục không phổ biến
CDS: CSDL nén
Output: $FFIL$: Danh sách các tập mục mờ phổ biến
Begin
1: **for** $i = 1$ to CDS **do**
3: ***CLA_Thread();***
4: End for
5: Khởi tạo FFIL;

6: for i=1 to automata cells do

7: Thực hiện PruneNeighbors() for cell[i];

8: Thực hiện DFS() function for cells[i];

9: for each anItemset on cell[i].FrequentItemset do

10: if anItemset does not exist in FFIL then

11: FFIL.add (anItemset);

12: else

13: Nothing;

14: End if

15: End for

16: End for

17: Return FFIL;

End.

Hàm CLA_Thread() được mô tả trong Thuật toán 3.3.

Thuật toán 3.3: CLA_Thread()
Input: Recodset (bản ghi dữ liệu nén), NodeParent[Cell] (đại diện của các cell)
Output: automata cells
Begin

1: Thread theard=new Thread();

2: thread.Start();

3: Initialize nodeChil=new Node();

4: **for** $i = 1$ to Recodset **do**

5: nodeChil.data= Recodset[value];

6: **If**(nodeChil in (Recodset)) then

7: nodeChil.data= Recodset[value]+ nodeChil.data;

8: else

9: NodeParent[Cell].next= nodeChil;

10: End if

11: End for

12: Return AutomataCells;

End

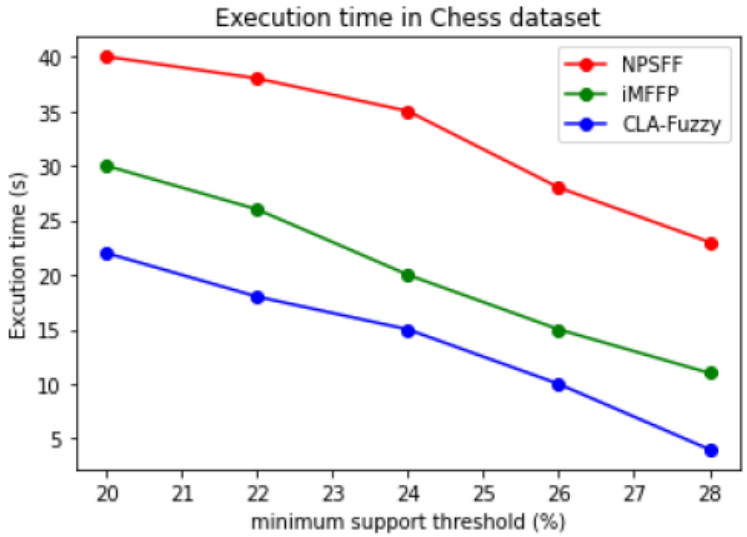
3.4 Thực nghiệm

Trong phần thực nghiệm NCS sử dụng bộ dữ liệu cửa hàng Foodmart, Chess và ChainStore từ bộ dữ liệu khai phá tập phổ biến [84] cho thử nghiệm này. Mô tả của tập dữ liệu được hiển thị trong bảng 3.7. Thực nghiệm này giới thiệu các kết quả thử nghiệm từ các thuật toán và so sánh chúng với kết quả của thuật toán NPSFF [CT2] và thuật toán iMFFP [99]. Thuật toán CLA- Fuzzy Mining có hiệu quả hơn hai thuật toán trước về thời gian xử lý và bộ nhớ lưu trữ tạm thời, theo kết quả thử nghiệm dựa trên tập dữ liệu được trình bày trong bảng 3.7. Thuật toán được đề xuất và tất cả các thuật toán được so sánh trong nghiên cứu này đã được chạy và thử nghiệm trong môi trường lập trình tích hợp IDE (integrated development environment), JDK8 (Java Development Kit) và ngôn ngữ lập trình hướng đối tượng JAVA trên máy tính chạy Windows 10 x64 được trang bị bộ vi xử lý Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz Intel 2,8 GHz và RAM 16 GB.

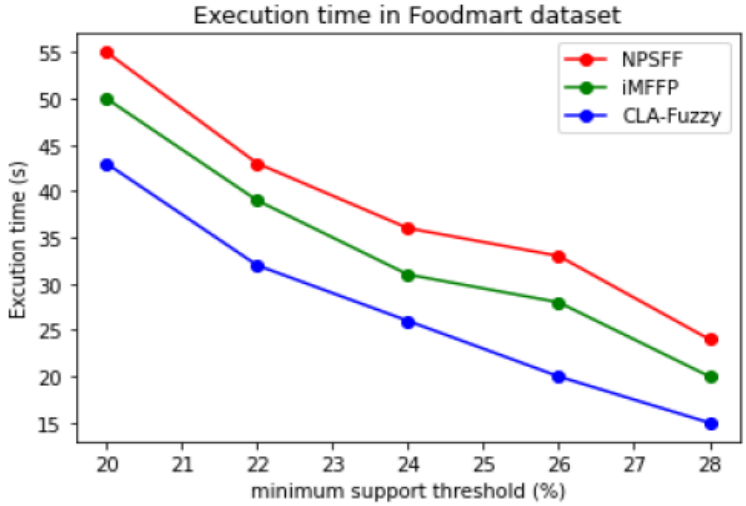
Bảng 3.7: Bảng dữ liệu thực nghiệm

Dataset name	Transaction#	Items#	Size (MB)
Chess	3,196	175	0.78 MB
Foodmart	4,141	1,559	12.4 MB
ChainStore	111,294	46,086	28.17 MB

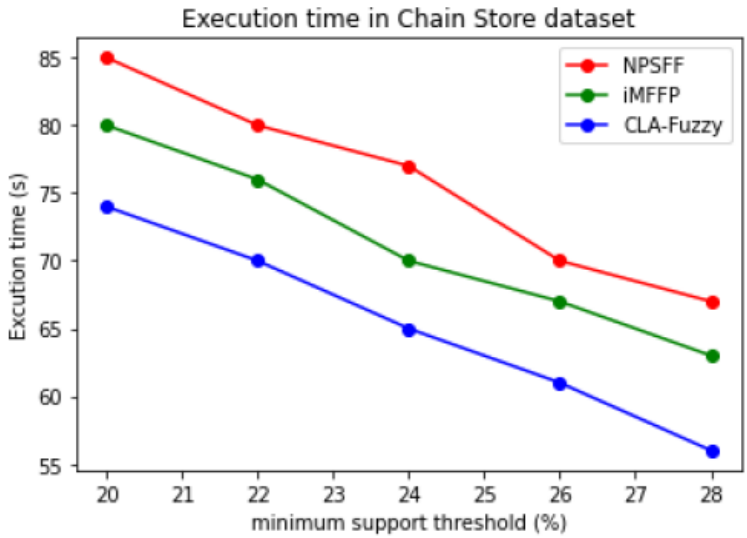
Hình 3.13 – 3.15 hiển thị kết quả của việc chạy hai thuật toán NPSFF, iMFFP và thuật toán mới được đề xuất trên bộ dữ liệu tiêu chuẩn thực.



Hình 3.13: Thời gian thực thi các thuật toán trên tập dữ liệu Chess Dataset

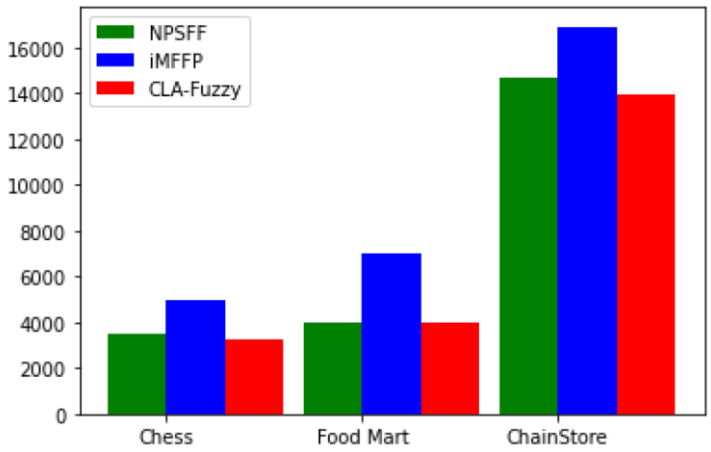


Hình 3.14:Thời gian thực thi các thuật toán trên tập dữ liệu Chess Dataset



Hình 3.15: Thời gian thực thi các thuật toán trên tập dữ liệu Chess Dataset

Hình 3.13, 3.14, 3.15 cho thấy hiệu suất của thuật toán được đề xuất CLA-Fuzzy Mining so với hai thuật toán iMFFP và NPSFF được thực hiện trên tất cả các tập dữ liệu đều giảm. Việc ứng dụng phương pháp loại bỏ các giao dịch dư thừa để nén tập dữ liệu và kết hợp với phương pháp xử lý song song đối với các ô chứa tập mục mờ phổ biến. So với các phương pháp xử lý song song khác, kỹ thuật CLA hoạt động tốt hơn đáng kể nhờ vào việc tự động cập nhật thông tin của môi trường cho các ô xung quanh.



Hình 3.16: Đánh giá bộ nhớ sử dụng của các thuật toán trên các tập dữ liệu

Theo hình 3.16, mức sử dụng bộ nhớ của iMFFP và NPSFF cao hơn so với CLA-Fuzzy. Quy trình khai phá các tập phổ biến mờ dựa trên danh sách láng giềng giúp hạn chế việc sử dụng bộ nhớ được yêu cầu.

3.5 Kết luận chương 3

Nhằm tăng tính hiệu quả tính toán trong các mô hình dữ liệu lớn, chương 3 luận án đề xuất phương pháp khai phá tập mục phổ biến mờ theo kỹ thuật xử lý song song CLA [CT3]. Theo CLA, không gian được biểu diễn như một mạng, với mỗi phần tử là một ô, từng dòng một, dữ liệu giao dịch sẽ được đọc và đồng thời được chuyển đến các ô, chúng sẽ cộng tác với nhau song song. Với việc không sử dụng quy tắc vùng lân cận, một loại tự động dữ liệu được gọi là tự động học di động bất thường (ICLA) được sử dụng để tạo danh sách vùng lân cận cho mỗi ô. Thông qua việc sử dụng các ô dữ liệu tự động này, việc khai phá tập phổ biến mờ được thực hiện. Quá trình này rút ngắn thời gian thực thi của thuật toán.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Mục đích chính của luận án là nghiên cứu một số phương pháp khai phá luật kết hợp mờ. Luận án nghiên cứu các phương pháp khai phá luật kết hợp trên cơ sở dữ liệu mờ dựa trên sự kết hợp của toán học mờ và cơ sở dữ liệu định lượng đã được đề xuất. Tuy nhiên, các phương pháp này đang trong quá trình phát triển, việc đề xuất các giải pháp mới nhằm hoàn thiện cho các là rất cần thiết. Vì vậy, luận án đề xuất hướng tiếp cận hiệu quả cho vấn đề khai phá các luật kết hợp mờ.

Các kết quả chính của luận án đạt được như sau:

(1) Đề xuất phương pháp xác định các tập mờ cho mỗi thuộc tính định lượng trong cơ sở dữ liệu thông qua kỹ thuật phân cụm EMC. Sau đó, các cụm này được sử dụng để phân loại mỗi thuộc tính định lượng như một tập mờ và xác định các hàm thuộc của chúng. Kết quả của bước này để chuyển đổi cơ sở dữ liệu định lượng sang cơ sở dữ liệu mờ. [CT2], [CT4].

(2) Đề xuất phương pháp khai phá tập mục mờ phổ biến dựa trên cấu trúc Nodelist, bước quan trọng trong khai phá luật kết hợp mờ. Quy trình khai phá tập mục mờ phổ biến dựa trên PP_code hoặc POS_code giúp hạn chế mức tiêu thụ bộ nhớ được yêu cầu. [CT1], [CT2], [CT5].

(3) Đề xuất phương pháp xử lý song song cho quá trình khai phá tập mục mờ phổ biến bằng cách sử dụng lý thuyết tự động học di động CLA. Với đề xuất này nhằm giải quyết giảm thời gian xử lý cho các cơ sở dữ liệu lớn. [CT3].

Những vấn đề đặt ra từ kết quả nghiên cứu của luận án:

- Các đánh giá hiệu suất cho kết quả nghiên cứu này dựa trên dữ liệu tĩnh, nhưng để đối phó với các vấn đề trong thế giới thực, dữ liệu có thể được phát triển theo thời gian với dữ liệu động. Trong những nghiên cứu tiếp theo, NCS tập trung nghiên cứu các phương pháp khai phá tập mục phổ biến mờ trên cơ sở dữ liệu luồng (data stream), dữ liệu chuỗi (sequence).
- Nghiên cứu các thuật toán xử lý khai phá dữ liệu cho cơ sở dữ liệu mờ có trọng số hoặc chứa yếu tố thời gian.

DANH MỤC CÁC CÔNG TRÌNH CỦA TÁC GIẢ

STT	TÊN BÀI BÁO
[CT1]	Tran, T. T., Nguyen, G. L., Truong, C. N., & Nguyen, T. T. "Mining Frequent Fuzzy Itemsets Using Node-List". Information Systems Design and Intelligent Applications Springer, Singapore, 37-48, 2018
[CT2]	Tran, T. T., Nguyen, T. N., Nguyen, T. T., Nguyen, G. L., & Truong, C. N., "A Fuzzy Association Rules Mining Algorithm with Fuzzy Partitioning Optimization for Intelligent Decision Systems". International Journal of Fuzzy Systems, 1-14, 2022 (SCIE – Q2)
[CT3]	Tran, T. T., Nguyen, T. T., Nguyen, G. L., & Truong, C. N. "Parallel Fuzzy Frequen Itemset Mining Using Cellular Automata". Journal of Computer Science and Cybernetics, 38(4), 293-310, 2022.
[CT4]	Trần Thị Thúy Trinh, Nguyễn Long Giang, Trương Ngọc Châu, Nguyễn Tấn Thuận. "Phân vùng dữ liệu mờ bằng phương pháp thống kê trong khai phá luật kết hợp mờ". Kỷ yếu hội thảo quốc gia về Các vấn đề chọn lọc Công nghệ thông tin và truyền thông – Quy Nhơn), 2017
[CT5]	Trần Thị Thúy Trinh, Nguyễn Tấn Thuận, Nguyễn Long Giang, Trương Ngọc Châu, Nguyễn Quang Huy. "Mô hình tư vấn học tập thông minh ứng dụng luật kết hợp mờ". Hội thảo quốc gia lần thứ XXIII: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông – Quảng Ninh, 2020

TÀI LIỆU THAM KHẢO

[1] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[2] J. Han, J. Pei, M. Kamber, D. J. Hand, and N. M. Adams, "Data Mining," *Wiley StatsRef Stat. Ref. Online*, pp. 1–7, 2014.

[3] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[4] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 1993, pp. 207–216.

[5] R. Agrawal, R. Srikant, and others, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, 1994, vol. 1215, pp. 487–499.

[6] T.-P. Hong, C.-W. Lin, and Y.-L. Wu, "Incrementally fast updated frequent pattern trees," *Expert Syst. Appl.*, vol. 34, no. 4, pp. 2424–2435, 2008.

[7] K. Hu, Y. Lu, L. Zhou, and C. Shi, "Integrating classification and association rule mining: A concept lattice framework," in *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, 1999, pp. 443–447.

[8] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "The Pre-FUFP algorithm for incremental mining," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9498–9505, 2009.

[9] Y. G. Sucahyo and R. P. Gopalan, "Building a more accurate classifier based on strong frequent patterns," in *Australasian Joint Conference on Artificial Intelligence*, 2004, pp. 1036–1042.

[10] B. Woźniak Michałand Krawczyk, "Combined classifier based on feature space partitioning," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 4, pp. 855–866, 2012.

[11] M. F. Zaman and H. Hirose, "Classification performance of bagging and boosting type ensemble methods with small training sets," *New Gener. Comput.*, vol. 29, no. 3, pp. 277–292, 2011.

[12] B. Lent, A. Swami, and J. Widom, "Clustering association rules," in

Proceedings 13th International Conference on Data Engineering, 1997, pp. 220–231.

[13] T.-P. Hong, C.-H. Wu, and others, “An improved weighted clustering algorithm for determination of application nodes in heterogeneous sensor networks,” 2011.

[14] F. Liu, Z. Lu, and S. Lu, “Mining association rules using clustering,” *Intell. Data Anal.*, vol. 5, no. 4, pp. 309–326, 2001.

[15] R. Agrawal and R. Srikant, “Mining sequential patterns,” in *Proceedings of the eleventh international conference on data engineering*, 1995, pp. 3–14.

[16] R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements,” in *International conference on extending database technology*, 1996, pp. 1–17.

[17] C. J. C. M. N. & S. J. M. Berzal F., “TBAR: An efficient method for association rule mining in relational databases,” *Data \& Knowl. Eng.*, vol. 37, no. 1, pp. 47–64, 2001.

[18] M.-S. Chen, J. Han, and P. S. Yu, “Data mining: an overview from a database perspective,” *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 866–883, 1996.

[19] J. S. Park, M.-S. Chen, and P. S. Yu, “Using a hash-based method with transaction trimming for mining association rules,” *IEEE Trans. Knowl. Data Eng.*, vol. 9, no. 5, pp. 813–825, 1997.

[20] Z. Deng, Z. Wang, and J. Jiang, “A new algorithm for fast mining frequent itemsets using N-lists,” *Sci. China Inf. Sci.*, vol. 55, no. 9, pp. 2008–2030, 2012.

[21] F. H. AL-Zawaidah, Y. H. Jbara, and A. L. Marwan, “An improved algorithm for mining association rules in large databases,” *World Comput. Sci. Inf. Technol. J.*, vol. 1, no. 7, pp. 311–316, 2011.

[22] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[23] R. Jain and W. Stallings, “Comments on" Fuzzy Set Theory versus Bayesian Statistics",” *IEEE Trans. Syst. Man. Cybern.*, vol. 8, no. 4, pp. 332–333, 1978.

[24] P. Pulkkinen and H. Koivisto, “A dynamically constrained multiobjective genetic fuzzy system for regression problems,” *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 1, pp. 161–177, 2009.

[25] R. Senge and E. Hüllermeier, “Top-down induction of fuzzy pattern trees,”

IEEE Trans. Fuzzy Syst., vol. 19, no. 2, pp. 241–252, 2010.

[26] X.-Z. Wang, L.-C. Dong, and J.-H. Yan, "Maximum ambiguity-based sample selection in fuzzy decision tree induction," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1491–1505, 2011.

[27] C. H. Nguyen, T. L. Pham, T. N. Nguyen, C. H. Ho, and T. A. Nguyen, "The linguistic summarization and the interpretability, scalability of fuzzy representations of multilevel semantic structures of word-domains," *Microprocess. Microsyst.*, vol. 81, p. 103641, 2021, doi: https://doi.org/10.1016/j.micpro.2020.103641.

[28] T.-P. Hong, K.-Y. Lin, and B.-C. Chien, "Mining fuzzy multiple-level association rules from quantitative data," *Appl. Intell.*, vol. 18, no. 1, pp. 79–90, 2003.

[29] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 1996, pp. 1–12.

[30] K. C. C. Chan, "Mining Fuzzy Association Rules 2 Related Work 3 F-APACS for Mining Fuzzy Association Rules," *Cikm97*, pp. 209–215, 1997, [Online]. Available: http://portal.acm.org/citation.cfm?doid=266714.266898

[31] C. M. Kuok, A. Fu, and M. H. Wong, "Mining Fuzzy Association Rules in Databases," *SIGMOD Rec. (ACM Spec. Interes. Gr. Manag. Data)*, vol. 27, no. 1, pp. 41–46, 1998, doi: 10.1145/273244.273257.

[32] T.-P. Hong, C.-S. Kuo, and S.-C. Chi, "Mining association rules from quantitative data," *Intell. data Anal.*, vol. 3, no. 5, pp. 363–376, 1999.

[33] A. Mangalampalli and V. Pudi, "Fuzzy association rule mining algorithm for fast and efficient performance on very large datasets," in *2009 IEEE International Conference on Fuzzy Systems*, 2009, pp. 1163–1168.

[34] C. Z. Janikow, "Fuzzy decision trees: issues and methods," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 28, no. 1, pp. 1–14, 1998.

[35] T. Watanabe and R. Fujioka, "Fuzzy association rules mining algorithm based on equivalence redundancy of items," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 1960–1965.

[36] T. Watanabe, "Fuzzy association rules mining algorithm based on output

specification and redundancy of rules," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, 2011, pp. 283–289.

[37] H. Jafarzadeh and M. Sadeghzadeh, "Improved Apriori Algorithm Using Fuzzy Logic," 2014.

[38] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "Linguistic data mining with fuzzy FP-trees," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4560–4567, 2010.

[39] S. H. A.H.M, R. Mustafa, S. K. Mondal, and M. A.-A. Bhuiyan, "A Fuzzy Frequent Pattern-Growth Algorithm for Association Rule Mining," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 5, pp. 21–33, 2015, doi: 10.5121/ijdkp.2015.5502.

[40] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An efficient tree-based fuzzy data mining approach," *Int. J. Fuzzy Syst.*, vol. 12, no. 2, pp. 150–157, 2010.

[41] C. W. Lin and T. P. Hong, "Mining fuzzy frequent itemsets based on UBFFP trees," *J. Intell. Fuzzy Syst.*, vol. 27, no. 1, pp. 535–548, 2014, doi: 10.3233/IFS-131022.

[42] T.-P. Hong, C.-W. Lin, and T.-C. Lin, "THE MFFP-TREE FUZZY MINING ALGORITHM TO DISCOVER COMPLETE LINGUISTIC FREQUENT ITEMSETS," *Comput. Intell.*, vol. 30, no. 1, pp. 145–166, 2014.

[43] J. C. W. Lin, T. P. Hong, and T. C. Lin, "A CMFFP-tree algorithm to mine complete multiple fuzzy frequent itemsets," *Appl. Soft Comput. J.*, vol. 28, pp. 431–439, 2015, doi: 10.1016/j.asoc.2014.11.049.

[44] J. C. W. Lin, T. P. Hong, T. C. Lin, and S. T. Pan, "An UBMFFP tree for mining multiple fuzzy frequent itemsets," *Int. J. Uncertainty, Fuzziness Knowlege-Based Syst.*, vol. 23, no. 6, pp. 861–879, 2015, doi: 10.1142/S0218488515500385.

[45] P. Arora, R. K. Chauhan, and A. Kush, "Frequent Itemsets from Multiple Datasets with Fuzzy data," *Int. J. Comput. Theory Eng.*, vol. 3, no. 2, p. 255, 2011.

[46] A. Zhang and W. Shi, "Mining significant fuzzy association rules with differential evolution algorithm," *Appl. Soft Comput.*, vol. 97, p. 105518, 2020.

[47] Z. Zhang, W. Pedrycz, and J. Huang, "Efficient mining product-based fuzzy association rules through central limit theorem," *Appl. Soft Comput.*, vol. 63,

pp. 235–248, 2018.

[48] J. C. W. Lin, T. Li, P. Fournier-Viger, and T. P. Hong, “A fast Algorithm for mining fuzzy frequent itemsets,” *J. Intell. Fuzzy Syst.*, vol. 29, no. 6, pp. 2373–2379, 2015, doi: 10.3233/IFS-151936.

[49] N. H. Đức, “Khai phá tập mục cổ phần cao và lợi ích cao trong cơ sở dữ liệu,” *Luận án tiến sĩ toán học*, 2010.

[50] N. L. Giang, “Nghiên cứu một số phương pháp khai phá dữ liệu theo tiếp cận lý thuyết tập thô,” *Luận án tiến sĩ toán học*, 2012.

[51] N. C. Đ. Nguyễn Công Hào, “Luật kết hợp mờ dựa trên ngữ nghĩa đại số gia tử,” *Tạp chí khoa học, Đại học Huế*, vol. 74A, no. 5, pp. 39–52, 2012.

[52] P. Fournier-Viger, J. C.-W. Lin, R. Nkambou, B. Vo, and V. S. Tseng, “High-utility pattern mining,” *Cham Springer*, 2019.

[53] Q. Huynh-Thi-Le, T. Le, B. Vo, and B. Le, “An efficient and effective algorithm for mining top-rank-k frequent patterns,” *Expert Syst. Appl.*, vol. 42, no. 1, pp. 156–164, 2015.

[54] B. Vo, S. Pham, T. Le, and Z.-H. Deng, “A novel approach for mining maximal frequent patterns,” *Expert Syst. Appl.*, vol. 73, pp. 178–186, 2017.

[55] T. Le and B. Vo, “An N-list-based algorithm for mining frequent closed patterns,” *Expert Syst. Appl.*, vol. 42, no. 19, pp. 6648–6657, 2015.

[56] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association in Large Databases,” *Proc. 1993 ACM SIGMOD Int. Conf. Manag. data - SIGMOD ’93*, pp. 207–216, 1993.

[57] J. Han, J. Pei, Y. Yin, and R. Mao, “Mining frequent patterns without candidate generation: A frequent-pattern tree approach,” *Data Min. Knowl. Discov.*, vol. 8, no. 1, pp. 53–87, 2004.

[58] Y. Ke, J. Cheng, and W. Ng, “An information-theoretic approach to quantitative association rule mining,” *Knowl. Inf. Syst.*, vol. 16, no. 2, pp. 213–244, 2008.

[59] V. Beiranvand, M. Mobasher-Kashani, and A. A. Bakar, “Multi-objective PSO algorithm for mining numerical association rules without a priori discretization,” *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4259–4273, 2014.

[60] D. Mart\’\in, A. Rosete, J. Alcalá-Fdez, and F. Herrera, “QAR-CIP-NSGA-II:

A new multi-objective evolutionary algorithm to mine quantitative association rules," *Inf. Sci. (Ny).*, vol. 258, pp. 1–28, 2014.

[61] Y. Djenouri, A. Bendjoudi, D. Djenouri, and M. Comuzzi, "GPU-based bio-inspired model for solving association rules mining problem," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2017, pp. 262–269.

[62] B. Minaei-Bidgoli, R. Barmaki, and M. Nasiri, "Mining numerical association rules via multi-objective genetic algorithms," *Inf. Sci. (Ny).*, vol. 233, pp. 15–24, 2013.

[63] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—II," *Inf. Sci. (Ny).*, vol. 8, no. 4, pp. 301–357, 1975.

[64] G. Chen and T. T. Pham, *Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems*. CRC press, 2000.

[65] J. Yen, *Fuzzy logic: intelligence, control, and information*. Pearson Education India, 1999.

[66] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—I," *Inf. Sci. (Ny).*, vol. 8, no. 3, pp. 199–249, 1975.

[67] Z. Komo, "Mining Fuzzy Association Rules on Large ln numerical Data - A Data Mining system for," 2003.

[68] M. Delgado, N. Manín, M. J. Martín-Bautista, D. Sánchez, and M.-A. Vila, "Mining Fuzzy Association Rules: An Overview," *Soft Comput. Inf. Process. Anal.*, pp. 351–373, 2006, doi: 10.1007/3-540-32365-1_15.

[69] K. C. C. Chan and W.-H. Au, "Mining fuzzy association rules," in *Proceedings of the sixth international conference on information and knowledge management*, 1997, pp. 209–215.

[70] A. Gyenesei, "A fuzzy approach for mining quantitative association rules," *Acta Cybern.*, vol. 15, no. 2, pp. 305–320, 2001.

[71] T.-P. Hong, C.-S. Kuo, and S.-L. Wang, "A fuzzy AprioriTid mining algorithm with reduced computational time," *Appl. Soft Comput.*, vol. 5, no. 1, pp. 1–10, 2004.

[72] J. S. Yue, E. Tsang, D. Yeung, and D. Shi, "Mining fuzzy association rules with weighted items," in *Smc 2000 conference proceedings. 2000 ieee*

international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0, 2000, vol. 3, pp. 1906–1911.

[73] G. Chen and Q. Wei, "Fuzzy association rules and the extended mining algorithms," *Inf. Sci. (Ny).*, vol. 147, no. 1–4, pp. 201–228, 2002.

[74] T.-P. Hong, M.-J. Chiang, and S.-L. Wang, "Mining from quantitative data with linguistic minimum supports and confidences," in *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291)*, 2002, vol. 1, pp. 494–499.

[75] S. Papadimitriou and S. Mavroudi, "The fuzzy frequent pattern tree," in *The WSEAS International Conference on Computers*, 2005, pp. 1–7.

[76] K. S. Prabha and R. Lawrance, "Mining fuzzy frequent itemset using compact frequent pattern (CFP) tree algorithm," in *International Conference on Computing and Control Engineering (ICCCE 2012)*, 2012, vol. 12, pp. 512–517.

[77] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4658–4672, 2013.

[78] M. Hao, W. Shi, H. Zhang, and C. Li, "Unsupervised change detection with expectation-maximization-based level set," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 1, pp. 210–214, 2013.

[79] T. Long, W. Jiao, G. He, and W. Wang, "Automatic line segment registration using Gaussian mixture model and expectation-maximization algorithm," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 5, pp. 1688–1699, 2013.

[80] D. McNeill and P. Freiberger, *Fuzzy logic: The revolutionary computer technology that is changing our world*. Simon and Schuster, 1994.

[81] Z. Ma, *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications: Modeling and Applications*. IGI Global, 2004.

[82] M. M. Gupta and T. Yamakawa, *Fuzzy logic in knowledge-based systems, decision and control*. Elsevier Science Inc., 1988.

[83] Q. Li, B. Moon, and others, “Indexing and querying XML data for regular path expressions,” in *VLDB*, 2001, vol. 1, pp. 361–370.

[84] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, “SPMF: a Java open-source pattern mining library,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3389–3393, 2014.

[85] R. Agrawal and J. C. Shafer, “Parallel mining of association rules,” *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 962–969, 1996.

[86] A. Cano, J. M. Luna, and S. Ventura, “High performance evaluation of evolutionary-mined association rules on GPUs,” *J. Supercomput.*, vol. 66, no. 3, pp. 1438–1461, 2013, doi: 10.1007/s11227-013-0937-4.

[87] L. Li and M. Zhang, “The strategy of mining association rule based on cloud computing,” *Proc. 2011 Int. Conf. Bus. Comput. Glob. Informatiz. BCGIn 2011*, pp. 475–478, 2011, doi: 10.1109/BCGIn.2011.125.

[88] S. S. Jain, B. B. Meshram, and M. Singh, “Voice of customer analysis using parallel association rule mining,” *2012 IEEE Students’ Conf. Electr. Electron. Comput. Sci. Innov. Humanit. SCEECS 2012*, no. c, pp. 0–4, 2012, doi: 10.1109/SCEECS.2012.6184770.

[89] W. Yong, Z. Zhe, and W. Fang, “A parallel algorithm of association rules based on cloud computing,” *2013 8th Int. ICST Conf. Commun. Netw. China, CHINACOM 2013 - Proc.*, pp. 415–419, 2013, doi: 10.1109/ChinaCom.2013.6694632.

[90] X. L. Shen and L. Tao, “Association rules parallel algorithm based on FP-tree,” *ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc.*, vol. 4, no. 1, pp. 687–689, 2010, doi: 10.1109/ICCET.2010.5485312.

[91] X. Leng and X. Li, “Alarm fuzzy association rules parallel mining in multi-domain distributed communication network,” *Int. Conf. Commun. Technol. Proceedings, ICCT*, no. 61171090, pp. 501–506, 2012, doi: 10.1109/ICCT.2012.6511270.

[92] E. Alba and J. M. Troya, “A survey of parallel distributed genetic algorithms,” *Complexity*, vol. 4, no. 4, pp. 31–52, 1999, doi: 10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.0.CO;2-4.

[93] T. P. Hong, C. H. Chen, Y. L. Wu, and Y. C. Lee, “A GA-based fuzzy mining

approach to achieve a trade-off between number of rules and suitability of membership functions," *Soft Comput.*, vol. 10, no. 11, pp. 1091–1101, 2006, doi: 10.1007/s00500-006-0046-x.

[94] T.-P. Hong, Y.-C. Lee, and M.-T. Wu, "An effective parallel approach for genetic-fuzzy data mining," *Expert Syst. Appl.*, vol. 41, no. 2, pp. 655–662, 2014.

[95] M. Burda, V. Pavliska, and R. Valasek, "Parallel mining of fuzzy association rules on dense data sets," *IEEE Int. Conf. Fuzzy Syst.*, pp. 2156–2162, 2014, doi: 10.1109/FUZZ-IEEE.2014.6891780.

[96] A. Zhang and W. Shi, "Mining significant fuzzy association rules with differential evolution algorithm," *Appl. Soft Comput.*, vol. 97, no. xxxx, p. 105518, 2020, doi: 10.1016/j.asoc.2019.105518.

[97] S. Jin, D. Dechev, and Z. Qu, "Parallel Particle Swarm Optimization (PPSO) on the coverage problem in pursuit-evasion games," *Simul. Ser.*, vol. 44, no. 6 BOOK, pp. 1–8, 2012.

[98] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel pso using mapreduce," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 7–14.

[99] C.-W. Lin, T.-P. Hong, Y.-F. Chen, T.-C. Lin, and S.-T. Pan, "An Integrated MFFP-tree Algorithm for Mining Global Fuzzy Rules from Distributed Databases.," *J. Univers. Comput. Sci.*, vol. 19, no. 4, pp. 521–538, 2013.

[100] R. Viswanathan and K. S. Narendra, "Stochastic automata models with applications to learning systems," *IEEE Trans. Syst. Man. Cybern.*, no. 1, pp. 107–111, 1973.

[101] M. Esmaeilpour, V. Naderifar, and Z. Shukur, "Cellular learning automata for mining customer behaviour in shopping activity," *Int. J. Innov. Comput. Inf. Control*, vol. 8, no. 4, pp. 2491–2511, 2012.

[102] H. Beigy and M. R. Meybodi, "A mathematical framework for cellular learning automata," *Adv. Complex Syst.*, vol. 7, no. 03n04, pp. 295–319, 2004.