**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**CHƯƠNG TRÌNH CHẤT LƯỢNG CAO**

**LÊ THÀNH NAM – LÊ HỒNG QUANG**

**KHAI THÁC LUẬT CÓ THỨ TỰ BÁN PHẦN**

**TRONG CƠ SỞ DỮ LIỆU CHUỖI**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**TP.HCM, NĂM 2022**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**CHƯƠNG TRÌNH CHẤT LƯỢNG CAO**

**LÊ THÀNH NAM – 18127158**

**LÊ HỒNG QUANG – 18127190**

**KHAI THÁC LUẬT CÓ THỨ TỰ BÁN PHẦN**

**TRONG CƠ SỞ DỮ LIỆU CHUỖI**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**GIÁO VIÊN HƯỚNG DẪN**

**GS.TS LÊ HOÀI BẮC**

**TP.HCM, NĂM 2022**

# **Lời cảm ơn**

Đầu tiên chúng tôi muốn dành lời cảm ơn sâu sắc đến giảng viên hướng dẫn của chúng tôi – Thầy Lê Hoài Bắc vì đã tận tình hướng dẫn và trợ giúp chúng tôi xuyên suốt quá trình thực hiện khóa luận. Khóa luận này gần như không thể hoàn thành nếu không có sự tham gia của thầy.

Tiếp theo, chúng tôi xin chân thành cảm ơn các giáo viên của Khoa Công nghệ thông tin, trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TP.Hồ Chí Minh vì đã cho chúng tôi một môi trường làm việc và học tập chuyên nghiệp trong suốt bốn năm học.

Chúng tôi cũng xin cảm tạ sự hỗ trợ nhiệt tình của phía gia đình và bạn bè trong những lúc khó khăn cũng như trong suốt quá trình thực hiện khóa luận.

Trong khóa luận này, mặc dù đã cố gắng hoàn thiện những gì trong khả năng nhưng vẫn không tránh được khả năng mắc sai sót. Rất mong quý thầy cô đưa ra những nhận xét và đánh giá để chúng tôi tiến hành chỉnh sửa trong thời gian sớm nhất.

Một lần nữa, xin chân thành cảm ơn và mong nhận được sự giúp đỡ của quý thầy cô để khóa luận tốt nghiệp được hoàn thành trọn vẹn.

Thành phố Hồ Chí Minh, tháng 07 năm 2022

Nhóm sinh viên thực hiện

Lê Hồng Quang – Lê Thành Nam

# **Mục lục**

Nhận xét của GVHD

Nhận xét của GVPB

[Lời cảm ơn i](#_Toc113625340)

Đề cương

[Mục lục ii](#_Toc113625341)

[Danh sách các hình iv](#_Toc113625342)

[Danh sách các bảng vi](#_Toc113625343)

[Chữ viết tắt vii](#_Toc113625344)

[Tóm tắt viii](#_Toc113625345)

[**Chương 1: Giới thiệu đề tài** 1](#_Toc113625346)

[1.1 Tổng quan bài toán khai thác luật: 1](#_Toc113625347)

[1.2 Động lực nghiên cứu: 2](#_Toc113625348)

[1.3 Mục tiêu nghiên cứu: 5](#_Toc113625349)

[**Chương 2: Cơ sở lý thuyết** 6](#_Toc113625350)

[2.1 Vấn đề bài toán: 6](#_Toc113625351)

[2.2 Thuật toán POERM: 10](#_Toc113625352)

[**Chương 3: Kết quả nghiên cứu** 20](#_Toc113625353)

[3.1 Đánh giá thực nghiệm: 20](#_Toc113625354)

[3.2 Nhận xét: 30](#_Toc113625355)

[3.3 Cải thiện thuật toán: 30](#_Toc113625356)

[3.3.1 Ý tưởng: 30](#_Toc113625357)

[3.3.2 Nguyên nhân: 30](#_Toc113625358)

[3.4 Kết quả so sánh giữa hai phiên bản POERM: 31](#_Toc113625359)

[3.4.1 Tập OnlineRetail: 31](#_Toc113625360)

[3.4.2 Tập Fruithut: 33](#_Toc113625361)

[3.4.3 Tập BMS WebView 1: 34](#_Toc113625362)

[3.4.4 Tập Retail: 36](#_Toc113625363)

[3.4.5 Tập Foodmart: 37](#_Toc113625364)

[3.5 Khảo sát ảnh hưởng của các tham số: 39](#_Toc113625365)

[3.5.1 Khảo sát sự ảnh hưởng về thời gian: 40](#_Toc113625366)

[3.5.2 Khảo sát ảnh hưởng của bộ nhớ: 43](#_Toc113625367)

[3.5.3 Các luật được khám phá: 46](#_Toc113625368)

[**Chương 4: Kết luận và hướng phát triển** 48](#_Toc113625369)

[4.1 Kết luận: 48](#_Toc113625370)

[4.2 Hướng phát triển: 48](#_Toc113625371)

[**Tài liệu tham khảo** 49](#_Toc113625372)

# **Danh sách các hình**

[Hình 1: Ví dụ minh họa một chuỗi sự kiện phức tạp 7](#_Toc110420210)

[Hình 2: Ví dụ minh họa cho 3 tham số thời gian do người dùng định nghĩa 8](#_Toc110420211)

[Hình 3: Ví dụ minh họa một chuỗi sự kiện phức tạp 10](#_Toc110420212)

[Hình 4: Ví dụ minh họa cho 3 tham số thời gian do người dùng định nghĩa 11](#_Toc110420213)

[Hình 5: Thuật toán POERM 13](#_Toc110420214)

[Hình 6: Quy trình MiningXEventSet 16](#_Toc110420215)

[Hình 7: Các mốc thời gian được tìm kiếm ứng với mỗi luật X **→** Y 19](#_Toc110420216)

[Hình 8: Input file OnlineRetail 21](#_Toc110420217)

[Hình 9: Output OnlineRetail 21](#_Toc110420218)

[Hình 10: Input của dataset Fruithut 22](#_Toc110420219)

[Hình 11: Output đã được xử lý của Fruithut 22](#_Toc110420220)

[Hình 12: Output đã chuyển đổi về trạng thái chưa xử lý 23](#_Toc110420221)

[Hình 13: Input BMS WebView 1 25](#_Toc110420222)

[Hình 14: Output BMS WebView 1 26](#_Toc110420223)

[Hình 15: Input retail 27](#_Toc110420224)

[Hình 16: Output retail 28](#_Toc110420225)

[Hình 17: Input foodmart 29](#_Toc110420226)

[Hình 18: Output foodmart 29](#_Toc110420227)

[Hình 19: Kết quả so sánh về thời gian trên tập OnlineRetail 31](#_Toc110420228)

[Hình 20: Kết quả so sánh về bộ nhớ trên tập OnlineRetail 31](#_Toc110420229)

[Hình 21: Kết quả so sánh về thời gian trên tập Fruithuit 33](#_Toc110420230)

[Hình 22: Kết quả so sánh về bộ nhớ trên tập Fruithuit 33](#_Toc110420231)

[Hình 23: Kết quả so sánh về thời gian trên tập BMS WebView 1 34](#_Toc110420232)

[Hình 24: Kết quả so sánh về bộ nhớ trên tập BMS WebView 1 35](#_Toc110420233)

[Hình 25: Kết quả so sánh về thời gian trên tập Retail 36](#_Toc110420234)

[Hình 26: Kết quả so sánh về bộ nhớ trên tập Retail 36](#_Toc110420235)

[Hình 27: Kết quả so sánh về thời gian trên tập Foodmart 37](#_Toc110420236)

[Hình 28: Kết quả so sánh về bộ nhớ trên tập Foodmart 38](#_Toc110420237)

[Hình 29: Quy ước về biểu đồ 39](#_Toc110420238)

[Hình 30: Ảnh hưởng của minsup đến thời gian trên tập OnlineRetail 40](#_Toc110420239)

[Hình 31: Ảnh hưởng của minsup đến thời gian trên tập Fruithut 40](#_Toc110420240)

[Hình 32: Ảnh hưởng của minconf đến thời gian trên tập OnlineRetail 41](#_Toc110420241)

[Hình 33: Ảnh hưởng của minconf đến thời gian trên tập Fruithut 41](#_Toc110420242)

[Hình 34: Ảnh hưởng của Span đến thời gian trên tập OnlineRetail 42](#_Toc110420243)

[Hình 35: Ảnh hưởng của Span đến thời gian trên tập Fruithut 42](#_Toc110420244)

[Hình 36: Ảnh hưởng của minsup đến bộ nhớ trên tập OnlineRetail 43](#_Toc110420245)

[Hình 37: Ảnh hưởng của minsup đến bộ nhớ trên tập Fruithut 43](#_Toc110420246)

[Hình 38: Ảnh hưởng của minconf đến thời gian trên tập OnlineRetail 44](#_Toc110420247)

[Hình 39: Ảnh hưởng của minconf đến bộ nhớ trên tập Fruithut 44](#_Toc110420248)

[Hình 40: Ảnh hưởng của Span đến bộ nhớ trên tập OnlineRetail 45](#_Toc110420249)

[Hình 41: Ảnh hưởng của Span đến bộ nhớ trên tập Fruithut 45](#_Toc110420250)

# **Danh sách các bảng**

[Bảng 1: Kết quả thực thi trên hai bộ dữ liệu 20](#_Toc110420251)

[Bảng 2: Thông số của ba bộ dữ liệu khác 24](#_Toc110420252)

[Bảng 3: Kết quả thực thi của ba bộ dữ liệu 24](#_Toc110420253)

[Bảng 4: So sánh 2 phiên bản POERM và POERM-Op trên tập OnlineRetail 32](#_Toc110420254)

[Bảng 5: So sánh 2 phiên bản POERM và POERM-Op trên tập Fruithut 34](#_Toc110420255)

[Bảng 6: So sánh 2 phiên bản POERM và POERM-Op trên tập BMS Web View 1 35](#_Toc110420256)

[Bảng 7: So sánh 2 phiên bản POERM và POERM-Op trên tập Retail 37](#_Toc110420257)

[Bảng 8: So sánh 2 phiên bản POERM và POERM-Op trên tập Foodmart 38](#_Toc110420258)

[Bảng 9: Ví dụ một số luật được khám phá 46](#_Toc110420259)

# **Chữ viết tắt**

Partially-Ordered Episode Rule Miner – POERM

Frequent Episode Mining – FEM

Partially-Ordered Episode Rule – POER

# **Tóm tắt**

Khai thác luật có thứ tự là một công đoạn quan trọng trong lĩnh vực khai thác dữ liệu phục vụ cho mục đích phân tích chuỗi các sự kiện hoặc ký tự. Công đoạn trên bao gồm xác định các chuỗi con xuất hiện thường xuyên trong một chuỗi lớn và kết hợp chúng lại với nhau tạo nên các tập luật có mối liên kết chặt chẽ giữa các sự kiện. Từ các luật được tạo nên, dữ liệu sẽ được thấu hiểu và khám phá một cách dễ dàng.

Vấn đề cốt lõi của khai thác luật có thứ tự nằm ở sự rập khuôn – sự lặp đi lặp lại của các luật theo các thứ tự khác nhau, dù về mặt ý nghĩa các thứ tự khác nhau này cùng biểu diễn cho một luật. Chính vì vậy, sẽ có các luật được xem là khác nhau nhưng trên thực tế lại miêu tả cùng một trường hợp.

Để có thể tìm ra các luật nhỏ mang tính tổng quát hơn và có khả năng thay thế phần lớn luật, khóa luận này nghiên cứu một cách tiếp cận mới để khai thác các luật, thuật toán Partially-Ordered Episode Rules Miner.

Kết quả đánh giá thực nghiệm trên một vài bộ dữ liệu chuẩn cho thấy sử dụng phương pháp này sẽ giảm nhiều gánh nặng về bộ nhớ cũng như tốc độ thực thi. Việc này sẽ giúp ích rất nhiều khi áp dụng việc khai thác luật cho các bộ dữ liệu có kích thước lớn. Kết quả thu được của phương pháp là các bộ luật hợp lệ tổng quát có thể biểu diễn cho hầu hết các thứ tự xuất hiện của bộ luật đó.

# **Chương 1: Giới thiệu đề tài**

## **1.1 Tổng quan bài toán khai thác luật:**

Hiện nay, sự bùng nổ thông tin cùng với sự phát triển của công nghệ lưu trữ và mạng Internet đã làm cho lượng dữ liệu phục vụ cho nhu cầu hằng ngày của các tổ chức trở nên phức tạp, đa dạng và phong phú hơn. Do đó, nhu cầu khám phá thông tin, tri thức cần thiết để định hướng chiến lược cho tổ chức đang là một thách thức lớn. Chính vì thế, lĩnh vực khai thác dữ liệu đã ra đời để giải mã các thông tin và tri thức tiềm ẩn sau dữ liệu.

Khai thác dữ liệu, hay còn gọi là khám phá tri thức, là quá trình khám phá những mẫu hay tri thức có ích từ nguồn dữ liệu, ví dụ như CSDL, văn bản, ảnh, dữ liệu Web, v.v…

Các mẫu khai thác được phải mang lại giá trị và lợi ích cho tổ chức và đặc biệt dễ hiểu.

Quá trình khám phá tri thức trong cơ sở dữ liệu được chia thành ba giải đoạn chính: Tiền xử lý dữ liệu, khai thác dữ liệu, và hậu xử lý. Quá trình này lặp đi lặp lại đến khi đạt được kết quả khả quan cuối cùng.

Hiện tại, khai thác dữ liệu thường tập trung vào các hướng chính và được đề cập nhiều nhất là phân lớp, phân cụm dữ liệu, khai thác luật kết hợp và khai thác luật tuần tự. Các mô hình liên quan được đề xuất gồm khai thác văn bản, khai thác dữ liệu Web, khai thác dữ liệu không gian và thời gian, v.v…

Một trong những mô hình mang tầm ảnh hưởng lớn và được tập trung nghiên cứu đó là mô hình cơ sở dữ liệu chuỗi. Trong đó, khai thác luật có thứ tự là một trong những chủ đề nghiên cứu mang tính thiết thực và quan trọng bậc nhất trong lĩnh vực khai thác dữ liệu. Mục đích của việc này là tìm ra các luật biểu diễn các quy luật tiềm ẩn trong cơ sở dữ liệu chuỗi. Mỗi luật thể hiện mỗi quan hệ giữa các mẫu dữ liệu theo thứ tự thời gian.

Quá trình khai thác luật có thứ tự trong cơ sở dữ liệu chuỗi thường được chia thành hai giai đoạn chính: Khai thác mẫu và tiếp theo là giai đoạn sinh luật có thứ tự.

Do vậy, số lượng luật tuần tự khai thác được là nhân tố quyết định đến độ hiệu quả của việc sinh luật.

Hạn chế chính của các thuật toán khai thác mẫu tuần tự tổng quát là cố gắng tìm tất cả những luật tuần tự có thể có. Điều này làm gia tăng nhanh chóng mẫu khai thác được, dẫn đến sự dư thừa, trùng lặp của các mẫu. Đồng thời, một số thuật toán phải lưu giữ luật khai thác được ở bước trước nhằm mở rộng và kết hợp tạo thành các luật mới dẫn đến sự tiêu tốn bộ nhớ và thời gian lưu trữ lớn.

## **1.2 Động lực nghiên cứu:**

Trong những năm gần đây, đã có rất nhiều thuật toán được thiết kế nhằm mục đích nhận dạng mẫu trong chuỗi dữ liệu rời rạc (một chuỗi các sự kiện hoặc ký hiệu) bởi tính phổ biến của loại dữ liệu này là cực kỳ cao. Ví dụ, một văn bản có thể được biểu diễn dưới dạng nhiều chuỗi các ký tự, một danh sách lịch sử mua hàng của một khách hàng có thể biểu diễn thành một chuỗi các giao dịch và việc xác định vị trí bằng vệ tinh có thể được biểu diễn bằng một chuỗi các địa điểm.

Trong khi một số phương pháp nhận dạng mẫu tìm thấy các mẫu tương đồng của các chuỗi dữ liệu [6, 7, 15, 16], có một số phương pháp khác lại tìm ra mẫu tương đồng chỉ trong một chuỗi dữ liệu có độ dài lớn. Một trong những công đoạn phổ biến của các phương pháp khác ở trên chính là khai thác luật phổ biến (Frequent Episode Mining - FEM) [9, 11, 12, 14]. Nó bao gồm việc tìm tất cả các luật phổ biến trong một chuỗi các sự kiện mà trong đó các chuỗi con có tần suất xuất hiện lớn hơn ngưỡng minsup được định nghĩa bởi người dùng.

Có hai kiểu chuỗi dữ liệu trong FEM: chuỗi đơn giản – chuỗi chỉ có một sự kiện xảy ra tại một mốc thời gian, tất cả các sự kiện đều có các mốc thời gian và hoàn toàn theo thứ tự, chuỗi phức tạp – chuỗi cho phép có nhiều sự kiện xảy ra trên một mốc thời gian.

Đã có nhiều thuật toán được ra đời phục vụ cho việc khám phá luật phổ biến như MINEPI hay WINEPI [12], EMMA và MINEPI+ [9] và TKE [8]. Trong khi một số thuật toán tìm ra các tập nối tiếp (tức là danh sách các sự kiện theo thứ tự), các thuật toán còn lại tìm các tập song song (tập các sự kiện đồng thời) hoặc tập kết hợp (sự kết hợp của các tập nối tiếp/song song). Mặc dù việc tìm các tập phổ biến là hữu ích thì các tập chỉ được khám phá dựa trên cơ sở hỗ trợ của chúng (tần suất xuất hiện) [1]. Do đó, một vài sự kiện chỉ có thể xuất hiện cùng nhau trong một tập một cách tình cờ. Hơn nữa, những tập phổ biến không cung cấp thông tin về khả năng một số sự kiện sẽ xảy ra sau một số sự kiện khác.

Để giải quyết những vấn đề này, một bước xử lý hậu kỳ có thể được áp dụng sau FEM, đó là kết hợp các cặp tập thường xuyên để tạo tập luật. Tập luật là một luật có dạng , chỉ ra rằng nếu một tập xuất hiện, nó sẽ được theo sau bởi một tập với một độ tin cậy hoặc xác suất được cho trước [4, 12].

Khai thác tập luật là hữu ích bởi vì việc này có thể thể hiện mối quan hệ thời gian mạnh mẽ giữa các sự kiện trong dữ liệu từ nhiều miền [2-4, 12]. Ví dụ, một luậtcó thể được tìm thấy trong dữ liệu xem phim của khán giả, qua đó cho biết rằng nếu một người xem các bộ phim a, b và c theo thứ tự thì người đó sau đấy sẽ xem bộ phim d. Dựa trên những luật như vậy, các quyết định tiếp thị hoặc các khuyến nghị đến khách hàng có thể được đưa ra một cách hiệu quả.

Tuy nhiên, một nhược điểm lớn của những thuật toán khai thác luật truyền thống chính là các sự kiện trong từng luật phải theo thứ thự một cách chặt chẽ. Dẫn đến, các luật tương tự nhau lại được xử lý khác nhau. Ví dụ, luật được cho là khác với những luật:

.

Tất cả những luật trên đều chứa cùng các sự kiện giống nhau. Đây là một vấn đề vì tất cả những luật trên là tương tự nhau và trong thực tế có thể đại diện cho cùng một tình huống, chẳng hạn một người đã xem ba bộ phim (ví dụ: Frozen, Sleeping Beauty và Lion King) thì sau đó sẽ xem bộ phim khác (ví dụ: Harry Potter). Bởi vì những luật này được xem là khác biệt, tần suất xuất hiện và độ tin cậy của chúng được tính toán tách biệt và có thể rất khác nhau. Hơn nữa, việc phân tích số lượng lớn các luật đại diện cho cùng một tình huống với những thay đổi nhỏ về thứ tự sẽ không thuận tiện cho người dùng. Do đó, sẽ có mong muốn trích xuất các loại luật mang tính tổng quát và linh hoạt hơn, cụ thể sự thay đổi về thứ tự giữa các sự kiện là chấp nhận được.

Khóa luận này giải quyết vấn đề trên bằng cách nghiên cứu một loại luật mới được gọi là các tập luật có thứ tự bán phần – Partially-Ordered Episode Rules, với những sự kiện trong phần tiền tố của một luật và trong phần hậu tố của một luật là không có thứ tự. Một POER có dạng với và là tập các sự kiện. Một luật được hiểu như sau: nếu tất cả sự kiện trong xuất hiện theo thứ tự bất kỳ thì chúng sẽ được theo sau bởi tất cả sự kiện trong theo thứ tự bất kỳ.

Ví dụ, một POER chỉ ra rằng nếu một người xem các bộ phim a,b và c theo thứ tự bất kỳ thì người đó sẽ xem bộ phim d. Lợi ích của việc tìm ra các POER là một luật đơn lẻ có thể thay thế được nhiều tập luật. Ví dụ, có thể thay thế

Tuy nhiên việc khám phá các POER là thử thách bởi vì chúng không bắt nguồn từ các tập. Do đó, một thuật toán mới phải được thiết kế để tìm ra các POER trong một dữ liệu chuỗi một cách hiệu quả.

Trong khóa luận này, vấn đề của việc khám phá các POER sẽ được định nghĩa và các thuộc tính của nó sẽ được nghiên cứu. Sau đấy, một thuật toán hiệu quả tên là Partially-Ordered Episode Rule Miner – Khai thác luật có thứ tự bán phần, sẽ được trình bày. Cuối cùng, một đánh giá thực nghiệm được thực hiện trên một số bộ dữ liệu chuẩn để đánh giá POERM. Các kết quả cho thấy rằng thuật toán này có hiệu suất tuyệt vời.

## **1.3 Mục tiêu nghiên cứu:**

Mục tiêu nghiên cứu mà khóa luận muốn hướng đến là áp dụng thuật toán POERM được đề xuất bởi Philippe Fournier-Viger và cộng sự để khai thác hiệu quả các luật có thứ tự bán phần, thông qua các luật đó giúp người dùng hiểu rõ thêm về bộ dữ liệu và từ đó làm tiền đề để thực hiện các bước khai thác dữ liệu tiếp theo một cách hiệu quả.

# **Chương 2: Cơ sở lý thuyết**

**2.1 Vấn đề bài toán:**

Kiểu dữ liệu sẽ xem xét và xử lý trong khai thác luật phổ biến là một kiểu chuỗi chứa các các dữ liệu đi cùng với các mốc thời gian cụ thể [9, 12].

Gọi chuỗi các sự kiện là tập hợp biểu diễn cho các vật phẩm cũng như kí tự. Thêm vào đó, gọi các mốc thời gian là tập hợp với bất kỳ số nguyên và luôn có mối quan hệ .

Khoảng thời gian sẽ có thời lượng là đơn vị thời gian. Hơn nữa, hai khoảng thời gian và được gọi là không trùng lặp nếu hay . Nói cách khác, hai khoảng thời gian này sẽ không chồng chéo lên nhau nếu chúng không trùng lặp.

Một tập hợp X E được gọi là chuỗi sự kiện. Theo đó, X được gọi là chuỗi *k*-sự kiệnnếu X có chứa *k* sự kiện.

Một chuỗi sự kiện phức tạp là một danh sách các chuỗi sự kiện theo thứ tự đi cùng với các mốc thời gian, được biểu diễn dưới dạng với ,

Một chuỗi sự kiện liên tục trong một chuỗi sự kiện phức tạp là một chuỗi sự kiện mà các sự kiện bên trong nó xảy ra cùng lúc hay cùng mốc thời gian. Nếu một chuỗi sự kiện phức tạp chứa không quá một sự kiện trong mỗi mốc thời gian, nó là chuỗi sự kiện đơn giản.

Dữ liệu các loại khác nhau đều có thể biểu diễn dưới dạng chuỗi như chuỗi báo động, chuỗi dữ liệu về vị trí di chuyển, ... [8].

Ví dụ, một chuỗi phức tạp được biểu diễn trong Hình 1. Chuỗi này bao gồm mười một mốc thời gian và tập sự kiện *.* Chuỗi trên chỉ ra rằng sự kiện c xuất hiện tại mốc thời gian *t1*, theo sau đó là sự kiện *{a, b}* là sự kiện liên tục tại mốc thời gian *t2*, theo tiếp là sự kiện đơn giản *d* tại mốc thời gian *t3*, sự kiện *a* tại mốc thời gian *t5* và sự kiện *c* tại mốc thời gian *t6*, v.v…

Events:

Timestamps :

ts:

Timestamps

mps :

c

a,b

d

a

c

b

d

a,b,c

a

Hình 1: Ví dụ minh họa một chuỗi sự kiện phức tạp

Khóa luận này nghiên cứu một loại luật mới được gọi là tập luật có thứ tự bán phần. Một POER có dạng với X E và Y E và X, Y . Ý nghĩa của luật như vậy là nếu tất cả các sự kiện từ X xuất hiện theo bất kỳ thứ tự nào trong chuỗi, chúng sẽ được theo sau bởi tất cả sự kiện từ Y. Để tránh việc tìm thấy các luật chứa các sự kiện quá xa rời, ba ràng buộc được chỉ định: (1) các sự kiện từ X phải xuất hiện trong khoảng thời gian tối đa , (2) các sự kiện từ Y phải xuất hiện trong khoảng thời gian tối đa và (3) thời gian giữa X và Y không được lớn hơn hằng số . Ba ràng buộc XSpan, YSpan và XYSpan phải được xác định bởi người dùng và được minh họa trong hình 2 cho một luật .

Hình 2: Ví dụ minh họa cho 3 tham số thời gian do người dùng định nghĩa

Hơn nữa, để lựa chọn các luật thú vị, hai độ đo được sử dụng gọi là độ hỗ trợ và độ tin cậy, được lấy cảm hứng từ các công việc trước đây trong khai thác luật. Chúng được định nghĩa dựa trên khái niệm về sự xuất hiện.

Nguyên nhân sử dụng độ hỗ trợ và độ tin cậy: độ hỗ trợ là một thước đo quan trọng bởi vì một luật có độ hỗ trợ rất thấp có thể xảy ra một cách tình cờ. Ngoài ra, từ góc độ kinh doanh, một luật với độ hỗ trợ thấp có thể không thú vị vì nó có thể không mang lại lợi nhuận khi quảng cáo các mặt hàng mà các khách hàng hiếm khi mua cùng nhau. Độ hỗ trợ được khai thác để phát hiện hiệu quả các quy tắc kết hợp. Mặt khác, độ tin cậy đo lường sự đáng tin của suy luận được đưa ra bởi một luật. Đối với luật cho trước, độ tin cậy càng cao thì càng có nhiều khả năng Y có mặt trong các giao dịch chứa X. Độ tin cậy cũng cung cấp ước tính về xác suất có điều kiện của Y với X được cho trước.

Một thời điểm diễn ra của một tập sự kiện F E trong một chuỗi sự kiện phức tạp S là một khoảng thời gian nơi mà tất cả sự kiện từ F xuất hiện, đó là F

Một thời điểm diễn ra của một luật trong một chuỗi sự kiện phức tạp S là một khoảng thời gian sao cho tồn tại một số mốc thời gian với , X có sự xuất hiện trong , Y có sự kiện trong ,

Việc phân tích sự xuất hiện của các tập sự kiện hoặc các luật, qua đó có thể tiết lộ các mối quan hệ thú vị giữa các sự kiện. Tuy nhiên, có một vấn đề là một vài sự xuất hiện có thể chồng lên nhau, do đó một sự kiện có thể được tính là một phần của nhiều lần xuất hiện. Để giải quyết vấn đề này, khóa luận này đề xuất chỉ xem xét một tập con của tất cả các sự xuất hiện được định nghĩa như sau. Một thời điểm diễn ra được cho là dư thừa trong một tập các sự xuất hiện nếu tồn tại một khoảng thời điểm diễn ra trùng lặp , sao cho hoặc Cho biểu thị tập hợp các lần xuất hiện không dư thừa của một tập sự kiện F trong một chuỗi S. Hơn nữa, cho biểu thị tập các lần xuất hiện không trùng lặp của một luật trong một chuỗi S.

Độ hỗ trợ của một luật được định nghĩa là . Độ hỗ trợ của một tập sự kiện F được định nghĩa là . Độ tin cậy của một luật được định nghĩa là . Nó đại diện cho xác suất có điều kiện mà các sự kiện từ X được theo sau bởi các sự kiện từ Y.

**Định nghĩa 1:**

Cho X, Y là các tập sự kiện, S là một chuỗi sự kiện phức tạp và năm tham số được định nghĩa bởi người dùng: XSpan, YSpan, XYSpan, minsup và minconf. Vấn đề của việc khai thác các POER là tìm tất cả các POER hợp lệ. Một POER r được gọi là phổ biến nếu và một POER r được gọi là hợp lệ nếu POER đó thường xuyên và .

Ví dụ:

Events:

Timestamps :

ts:

Timestamps

mps :

c

a,b

d

a

c

b

d

a,b,c

a

Hình 3: Ví dụ minh họa một chuỗi sự kiện phức tạp

Xét *hình 3*, minsup = 3, minconf = 0.6, XSpan = 3, XYSpan = 1 và YSpan = 1. Sự xuất hiện của . Sự xuất hiện của luật là . Do đó, sup(R) = 3, conf (R) = 2/3 và R là một luật hợp lệ.

**2.2 Thuật toán POERM:**

Khó khăn của việc khai thác POER nằm ở số lượng luật được sinh ra tương ứng với số lượng sự kiện. Một tập dữ liệu gồm m sự kiện riêng biệt có thể sản sinh ra số lượng luật lên đến con số . Hơn nữa, mỗi luật có thể có một số lượng lớn các thời điểm diễn ra trong một chuỗi. Nói cách khác, gánh nặng về bộ nhớ lưu trữ là quá lớn khi sử dụng bộ luật POER.

Đầu tiên, thuật toán POERM sẽ tìm tất cả các tập sự kiện có khả năng là tiền tố của một luật hợp lệ cũng như các thời điểm diễn ra của chúng trong chuỗi đầu vào và lưu lại chúng. Tiếp đến, thuật toán tìm kiếm các hậu tố có thể kết hợp với các tiền tố kể trên để xây dựng các POER. Các POER hợp lệ này sẽ được lưu giữ lại và biểu diễn cho người sử dụng thấy được. Để tránh việc suy xét tất cả các luật có thể, thuật toán POERM tối ưu hóa bằng việc cắt giảm không gian tìm kiếm qua Quy tắc 1.

**Quy tắc 1**:

Một tập sự kiện X không thể là tiền tố của một luật hợp lệ nếu . Một tập sự kiện Y không thể là hậu tố của một luật hợp lệ nếu .

Thuật toán POERM được biểu diễn qua hình 5, nhận dữ liệu đầu vào là một chuỗi sự kiện phức tạp S cùng với các tham số được định nghĩa bởi người dùng là XSpan, YSpan, XYSpan, minsup và minconf.

Hình 4: Ví dụ minh họa cho 3 tham số thời gian do người dùng định nghĩa

Ban đầu, thuật toán đọc chuỗi sự kiện S và tạo ra XFres – một bản copy của chuỗi S nhưng chỉ chứa các sự kiện có support không bé hơn minsup. Các sự kiện khác sẽ được loại bỏ bởi vì chúng không thể xuất hiện trong các luật hợp lệ dựa theo Quy tắc 1. Sau đó, thuật toán tìm kiếm các tiền tố của luật bằng cách gọi quy trình MiningXEventSet với các tham số đầu vào XFres, XSpan và minsup. Quy trình này trả về giá trị là một danh sách xSet chứa các chuỗi sự kiện có khả năng làm tiền tố của các POER hợp lệ, trong đó từng tập sự kiện có tối thiểu minsup số lần diễn ra không trùng lặp của một khoảng thời gian không lớn hơn XSpan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm 1:** *POERM* | | | |  |
| **Input:** an event sequence S, the *XSpan, YSpan, XYSpan, minsup* and *minconf* parameters; | | | |  |
| **Output:** the set POERs of valid partially-ordered episode rules | | | |  |
| **1** | *XFres* = *loadFrequentSequence*(*S*, *minsup*); | | |  |
| **2** | *xSet* = *MiningXEventSet*(*XFres*, *XSpan*, *minsup*); | | |  |
| **3** | *YFres* = *loadFrequentSequence*(*XFres*, *minsup* × *minconf*); | | |  |
| **4** | POERs ← ∅; | | |  |
| **5** | **foreach** event set *x* in *xSet* do | | |  |
| **6** |  | **for** *i* = 1 to (*XY Span +Y Span*) **do** | |  |
| **7** |  |  | *xOccurrenceList*←{*t*|*t* = *occur*.*end* + *i, occur ∈ x.OccurrenceList*} |  |
| **8** |  | **end** | |  |
| **9** |  | Scan each timestamp of *YFres* in *xOccurrenceList* to obtain a map *conseMap* that records each event e and its occurrence list; | |  |
| **10** |  | Scan *conseMap* and put the pair (*x→e*, *OccurrenceList*) in a queue *candidateRuleQueue* (note: infrequent rules are kept because event e may be extended to obtain some frequent rules); | |  |
| **11** |  | Add each rule *x → e* such that *|occ(x → e)| ≥ minconf* × *|occur(x)|* into the set POERs; | |  |
| **12** |  | **while** *candidateRuleQueue* ≠ ∅ **do** | |  |
| **13** |  |  | Pop a rule X → Y from *candidateRuleQueue*; |  |
| **14** |  |  | For each occurrence occur of X → Y , let; |  |
| **15** |  |  | *start* ←*max*(*occur.X.end* + 1, *occur.Y.end* − *YSpan* + 1); |  |
| **16** |  |  | *end* ←*min*(*occur.X.end* + *XYSpan* + *YSpan*, *occur.X.start* +  *YSpan*); |  |
| **17** |  |  | Scan each timestamp in *[start,end),* add each candidate rule in *candidateRuleQueue*, and add each valid POER in POERs; |  |
| **18** |  | **end** | |  |
| **19** | **end** | | |  |
| **20** | **return** POERs; | | |  |

Hình 5: Thuật toán POERM

Quy trình MiningXEventSet được biểu diễn qua hình 6. Trước hết, quy trình quét chuỗi XFres để tìm danh sách các thời điểm diễn ra của từng sự kiện. Dữ kiện này được lưu lại trong một map-fresMap bao gồm các cặp giá trị (key, value) với key biểu thị cho sự kiện và value là danh sách thời điểm diễn ra của sự kiện đó. Sau đó, quy trình sẽ quét tất cả các cặp giá trị trong fresMap và đưa cặp giá trị (key, value) mà có tối thiểu minsup số lần diễn ra không trùng lặp hay có thể hiểu |value| (số lượng phần tử trong value) – số lượng thời điểm diễn ra không trùng lặp lớn hơn hoặc bằng minsup – vào xSet. Tại thời điểm này, xSet sẽ chứa tất cả các tập 1-sự kiện có khả năng làm tiền tố của luật hợp lệ theo Quy tắc 1. Sau đó, quy trình quét toàn bộ chuỗi XFres thêm một lần nữa để mở rộng các tập 1-sự kiện thành các tập 2-sự kiện và tiếp tục mở rộng các tập 2-sự kiện thành các tập 3-sự kiện và chỉ kết thúc cho tới khi không tăng số lượng sự kiện trong chuỗi sự kiện lên được nữa. Trong quá trình mang tính lặp lại trên, mỗi tập sự kiện được sản sinh ra mà có số lượng các thời điểm diễn ra không trùng lặp lớn hơn minsup sẽ được thêm vào danh sách xSet và được xem xét để mở rộng thành các tập chứa số lượng sự kiện lớn hơn. Cuối cùng, quy trình MiningXEventSet sẽ trả về xSet, danh sách chứa tất cả các tiền tố hợp lệ của các POER hợp lệ.

Một thử thách trong việc cài đặt quy trình MiningXEventSet hiệu quả là các chuỗi sự kiện – theo định nghĩa – không có quy luật về mặt thứ tự. Chính vì vậy, các thứ tự khác nhau của cùng một tập sự kiện sẽ đại diễn cho cùng một tập sự kiện. Ví dụ, ba chuỗi sự kiện {a, b, c}, {b, a, c}, {a, c, b} là cùng một chuỗi sự kiện. Nhằm loại bỏ việc sản sinh ra các tập sự kiện trùng lặp nhiều lần, các sự kiện trong một tập sự kiện sẽ được sắp xếp theo thứ tự của bảng chữ cái (ví dụ {a,b,c}) hoặc thứ tự số học. Cụ thể ở ví dụ trên, cả ba chuỗi sự kiện sẽ chỉ được biểu diễn bằng chuỗi sự kiện {a, b, c} và quy trình sẽ chỉ mở rộng một chuỗi sự kiện F bằng cách thêm một sự kiện e vào F với điều kiện sự kiện e phải lớn hơn sự kiện cuối cùng trong F (sự kiện mang giá trị lớn nhất về mặt chữ cái hoặc số học).

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm 2:** *MiningXEventSet* | | | |
| **Input:** *XFres*: the sequence with only events having support ≥ *minsup*; XSpan: maximum window size; *minsup* threshold; | | | |
| **Output:** a list of event sets that may be antecedents of valid POERs | | | |
| **1** | Scan the sequence *XFres* to record the occurrence list of each event in a map *fresMap* (*key* = event set, *value* = occurrence list); | | |
| **2** | *xSet* ← all the pairs of *fresMap* such that |value| ≥ *minsup*; | | |
| **3** | *start* ← 0; | | |
| **4** | **while** *start* < |*xSet*| **do** | | |
| **5** |  | F ← *xSet*[*start*].*getKey*; | |
| **6** |  | *OccurrenceList* ← *xSet*[*start*].*getValue*; | |
| **7** |  | Clear *fresMap*; | |
| **8** |  | *start* = *start* +1; | |
| **9** |  | **foreach** occurrence pos in *OccurrenceList* **do** | |
| **10** |  |  | *pStart* ← pos.end− *XSpan* +1; *pEnd* ←− pos.*start*+ *XSpan*; |
| **11** |  |  | Search the time intervals [*pStart*, *pos*.*start*), [*pos*.*end* + 1, *pEnd*), [*pos*.*start*, *pos*.*end*] to add each event set *F* ∪ {*e*} such that *e* > *F.lastItem* and its occurrences of the forms [*i, pos.end], [pos.start, i]* or *[pos.start, pos.end*], in the map *fresMap*; |
| **12** |  | **end** | |
| **13** |  | Add each pair of *fresMap* such that |value| ≥ *minsup* into *xSet;* | |
| **14** | **end** | | |
| **15** | return *xSet* | | |

Hình 6: Quy trình MiningXEventSet

Một điều kiện cốt lõi được xem như là thử thách thứ hai trong việc cài đặt quy trình trên là tìm ra phương pháp để mở rộng một tập *l*-sự kiện F thành các tập (*l*+1)-sự kiện và tính toán các thời điểm diễn ra của nó bằng cách sử dụng chuỗi XFres. Để tránh việc quét hết toàn bộ chuỗi XFres trên, công việc tính toán sẽ được thực hiện bằng cách tìm kiếm xung quanh mọi thời điểm diễn ra pos (occurrence pos) của F trong danh sách OccurrenceList của F. Cho pos.start, pos.end là hai mốc thời gian bắt đầu và kết thúc của thời diểm diễn ra pos. Thuật toán tìm kiếm các sự kiện diễn ra trong ba khoảng thời gian [pos.end – XSpan + 1, pos.start), [pos.end +1, pos.start + Xspan) và [pos.start, pos.end]. Với mỗi sự kiện e lớn hơn sự kiện cuối cùng trong F, các thời điểm diễn ra của chuỗi sự kiện sẽ có dạng [i, pos.end], [pos.start, i] hay [pos.start, pos.end] với i là mốc thời gian nằm trong ba khoảng không gian tìm kiếm đã được đề cập trước đó. Và các thời điểm diễn ra của chuỗi sự kiện trên sẽ được thêm vào fresMap. Tiếp đến, quy trình quét toàn bộ fresMap để đếm số lượng các thời điểm diễn ra của từng tập (*l*+1)-node sự kiện và tất cả các tập sự kiện có các thời điểm diễn ra không trùng lặp không bé hơn minsup được thêm vào xSet.

Việc đếm được tối đa số lần các thời điểm diễn ra không trùng lặp của một tập *l*-sự kiện F được thực hiện bằng cách sử dụng OccurenceList của F (không được nêu trong hình 6). Quy trình đầu tiên áp dụng thuật toán sắp xếp quick sort để sắp xếp lại OccurenceList theo thứ tự mốc thời gian kết thúc (pos.end) tăng dần. Sau đó, một tập CoverSet được tạo ra để lưu trữ số lượng tối đa các thời điểm diễn ra không trùng lặp của F. Thuật toán thực hiện vòng lặp đi hết các phần tử trong OccurrenceList của F để kiểm tra từng thời điểm diễn ra từ lần đầu tiên đến lần cuối cùng. Nếu thời điểm diễn ra đang được xét ở hiện tại không trùng lặp với thời điểm cuối cùng vừa được thêm vào CoverSet thì nó sẽ được thêm vào CoverSet. Nếu không, nó sẽ bị bỏ qua. Khi vòng lặp kết thúc, tập CoverSet sẽ chứa số lượng tối đa các thời điểm diễn ra không trùng lặp của F.

Sau khi áp dụng quy trình MiningXEventSet để tìm kiếm tiền tố, thuật toán POERM tìm kiếm các tập sự kiện có thể là hậu tố kết hợp được với các tiền tố đã được thu thập trước đó nhằm mục đích hình thành các POER hợp lệ. Ở bước này, việc đầu tiên thuật toán thực hiện là loại bỏ tất cả các sự kiện có số lượng thời điểm diễn ra nhỏ hơn minsup × minconf từ chuỗi XFres để tạo nên YFres dựa vào Quy tắc 1. Tiếp đến, một vòng lặp đi qua mỗi tiền tố x trong xSet để tìm kiếm hậu tố. Trong vòng lặp trên, các khoảng thời gian mà hậu tố có khả năng xuất hiện sẽ được xác định trước và lưu vào biến xOccurrenceList. Thêm vào đó, một cấu trúc dữ liệu dạng map conseMap sẽ được tạo ra để lưu mỗi sự kiện e cùng với danh sách các thời điểm diễn ra của nó cho những mốc thời gian này trong YFres. Sau đó, map sẽ được quét toàn bộ để tạo ra hàng chờ candidateRuleQueue chứa các luật có dạng x **→** e cùng với danh sách các thời điểm diễn ra của luật trên. Nếu như luật x **→** e có |occ (x **→** e) | ≥ minconf × |occ(x)| thì sẽ được thêm vào tập chứa các POER hợp lệ. Lúc này, candidateRuleQueue sẽ chứa tất cả các luật ứng cử viên với hậu tố là *1*-sự kiện.

Ở giai đoạn này, thuật toán thực hiện thêm một vòng lặp để loại các luật khỏi hàng chờ để tiến hành mở rộng hậu tố bằng cách kết hợp với các sự kiện đơn. Giai đoạn này được tiến hành thông qua việc quét toàn bộ conseMap. Luật mở rộng thu được sẽ có dạng và danh sách các thời điểm diễn ra của luật trên được thêm vào candidateRuleQueue để tiếp tục phục vụ cho quá trình mở rộng tiếp theo. Hơn nữa, mỗi luật có |occ (x **→** e) | ≥ minconf × |occ(x)| sẽ được thêm vào tập chứa các POER hợp lệ. Vòng lặp sẽ tiếp tục cho đến khi candidateRuleQueue trở thành hàng chờ rỗng. Cuối cùng, thuật toán trả về tất cả các POER hợp lệ.

Có thể nhận thấy rằng một thời điểm diễn ra của luật có hậu tố là tập l-sự kiện có thể không được đếm vào tần suất xuất hiện của nó trong khi lại có thể được đếm vào tần suất của một luật có hậu tố (l+1)-sự kiện – luật được mở rộng dựa vào luật trước đó. Ví dụ, xem xét chuỗi trong hình 1, ta có XSpan = YSpan = 2 và XYSpan = 1. Thời điểm diễn ra của luật {a} **→** {b} trong khoàng thời gian [t5, t7] không được đếm vào tần suất xuất hiện của luật {a} **→** {b} bởi vì khoảng cách thời gian giữa a và b lớn hơn XYSpan. Nhưng luật {a} **→** {b} có thể được mở rộng hậu tố trở thành {a} **→** {b, c} và lúc này thời điểm diễn ra trong khoảng thời gian [t5, t7] được tính vào tần suất xuất hiện của luật. Để tìm ra các thời điểm diễn ra của luật dạng được mở rộng từ luật , một cách tiếp cận sau được sử dụng. Với từng thời điểm diễn ra của luật một biến start được khởi tạo giá trị max(occur.X.end +1, occur.Y.end – Yspan +1), một biến end được khởi tạo giá trị (occur.X.end + XYSpan + YSpan, occur.Y.start +YSpan). Tiếp đến, ba khoảng thời gian được khoanh vùng là [start, occur.Y.end), (occur.Y.end, end) và [occur.Y.start, occur.Y.end] để thêm luật sao cho e > Y.lastItem cùng các thời điểm diễn ra của nó sẽ xuất hiện dưới dạng [i, occur.Y.end], [occur.Y.start, i] hay [occur.Y.start, occur.Y.end] vào conseMap. Ba khoảng thời gian trên đều được lưu trữ trong hình 7 và cho phép thuật toán tìm ra chính xác các thời điểm diễn ra của các luật có hậu tố là (*l+1*)-node.

Trong hình 7 dưới, tY biểu diễn cho occur.Y và tX biểu diễn cho occur.X:

Hình 7: Các mốc thời gian được tìm kiếm ứng với mỗi luật X **→** Y

Thuật toán POERM được đề xuất có thể tìm thấy tất cả bộ POER hợp lệ vì nó cắt giảm các sự kiện không thể là thành phần của luật hợp lệ theo Quy tắc 1. Chương dưới đây đưa ra các đánh giá tổng quan về POERM, nơi hiệu suất được so sánh với phiên bản đã cải tiến POERM-Optimize.

# **sChương 3: Kết quả nghiên cứu**

## **3.1 Đánh giá thực nghiệm:**

Thực hiện chạy thuật toán trên hai bộ dữ liệu chuẩn từ thư viện SPMF [5], có tên OnlineRetail và Fruithut.

OnlineRetail là một chuỗi gồm 541,909 giao dịch từ một cửa hàng bán lẻ trực tuyến có trụ sở tại Vương quốc Anh với 2,603 sự kiện riêng biệt cho biết việc mua các mặt hàng.

Fruithut là một chuỗi gồm 181,970 giao dịch của khách hàng từ một cửa hàng bán lẻ ở Mỹ, chủ yếu là bán trái cây với 1,265 loại sự kiện riêng biệt.

Thuật toán có năm tham số (minSupport, minConfidence, XSpan, YSpan, XYSpan), các tham số sẽ được tùy chỉnh sao cho phù hợp với từng bộ dữ liệu trên, lần lượt là:

OnlineRetail : (7000, 0.5, 5, 5, 5)

Fruithut: (5000, 0.5, 5, 5, 5)

Bảng 1: Kết quả thực thi trên hai bộ dữ liệu

|  |  |  |
| --- | --- | --- |
| Tên bộ dữ liệu | Thời gian (s) | Bộ nhớ (MB) |
| OnlineRetail | 1848 | 2012 |
| Fruithut | 733 | 595 |

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 8: Input file OnlineRetail

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 9:Output OnlineRetail

Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 10:Input của dataset Fruithut

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 11: Output đã được xử lý của Fruithut

Ảnh có chứa văn bản, tờ báo

Mô tả được tạo tự động

Hình 12: Output đã chuyển đổi về trạng thái chưa xử lý

Ở tập dữ liệu Fruithut, do dữ liệu đầu vào bao gồm tên các sự kiện và các con số là ID của từng loại sự kiện khác nhau nên khi thực thi thuật toán, chúng ta sẽ chỉ làm việc với các con số (ID). Sau khi có kết quả như hình 11 chúng ta sẽ thực thi hàm chuyển đổi để biến đổi các ID trong dữ liệu đầu ra thành các tên gọi sự kiện ứng với từng ID trong dữ liệu đầu vào như hình 12.

Thử nghiệm trên ba bộ dữ liệu khác:

Bảng 2: Thông số của ba bộ dữ liệu khác

|  |  |  |  |
| --- | --- | --- | --- |
| Tên bộ dữ liệu | Số lượng giao dịch | Loại sự kiện | Tham số |
| BMS WebView 1 | 59602 | 497 | (200,0.5,5,5,5) |
| retail | 9284 | 2626 | (4000,0.5,3,3,3) |
| foodmart | 4141 | 1559 | (5,0.5,5,5,5) |

Bảng 3: Kết quả thực thi của ba bộ dữ liệu

|  |  |  |
| --- | --- | --- |
| Tên bộ dữ liệu | Thời gian (s) (xấp xỉ) | Bộ nhớ (MB) (xấp xỉ) |
| BMS WebView 1 | 471 | 410 |
| retail | 383 | 461 |
| foodmart | 14 | 33 |

Ảnh có chứa văn bản

Mô tả được tạo tự độngHình 13: Input BMS WebView 1

Ảnh có chứa văn bản, tờ báo

Mô tả được tạo tự động

Hình 14: Output BMS WebView 1

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 15: Input retail

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 16: Output retail

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 17: Input foodmart

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 18: Output foodmart

## **3.2 Nhận xét:**

Với tập dữ liệu có kích thước lớn thì thuật toán POERM sẽ tiêu tốn nhiều thời gian thực thi cũng như bộ nhớ hơn so với tập dữ liệu có kích thước nhỏ. Do đó, cần cải thiện thời gian thực thi hoặc lượng bộ nhớ tiêu tốn của thuật toán.

## **3.3 Cải thiện thuật toán:**

### **3.3.1 Ý tưởng:**

Thay đổi cấu trúc dữ liệu được sử dụng trong thuật toán. Cụ thể, thay vì kết hợp cả hai kiểu dữ liệu Lists và Numpy Arrays để lưu trữ và xử lý, chúng ta sẽ chỉ dùng kiểu dữ liệu Lists.

### **3.3.2 Nguyên nhân:**

Về mặt lý thuyết, Numpy Arrays có ba điểm ưu việt hơn Lists. Numpy Arrays sẽ chiếm ít không gian bộ nhớ hơn, có thời gian thực thi nhanh hơn khi xử lý trên tập dữ liệu có kích thước lớn và thư viện có sẵn các hàm tối ưu hơn Lists.

Nhưng trong một vài trường hợp Numpy Arrays sẽ tỏ ra kém hiệu quả hơn Lists. Chẳng hạn các thao tác thêm mới phần tử vào Lists/Numpy Arrays. Cụ thể, trong thuật toán POERM, các thao tác tính toán – điều giúp Numpy Arrays hiệu quả hơn Lists chiếm rất ít, đa số các thao tác so sánh, thêm mới phần tử và thực hiện các vòng lặp.

## **3.4 Kết quả so sánh giữa hai phiên bản POERM:**

### **3.4.1 Tập OnlineRetail:**

Các tham số: minSupport = 7000, minConfidence = 0.5, XSpan = YSpan = XYSpan = 5.

Chart, bar chart

Description automatically generated

Hình 19: Kết quả so sánh về thời gian trên tập OnlineRetail

Chart, bar chart

Description automatically generated

Hình 20: Kết quả so sánh về bộ nhớ trên tập OnlineRetail

Bảng 4: So sánh 2 phiên bản POERM và POERM-Op trên tập OnlineRetail

|  |  |  |
| --- | --- | --- |
| Thuật toán | Thời gian (s) | Bộ nhớ (MB) |
| POERM | 1848 | 2012 |
| POERM-Op | 836 | 2127 |

Nhận xét:

Thời gian thực thi giảm đáng kể, cụ thể giảm đi hơn 2 lần, tuy nhiên bộ nhớ tăng không đáng kể, cụ thể tăng khoảng 1.05 lần.

**3.4.2 Tập Fruithut:**

Các tham số: minSupport = 5000, minConfidence = 0.5, XSpan = YSpan = XYSpan = 5.

Chart, bar chart

Description automatically generated

Hình 21: Kết quả so sánh về thời gian trên tập Fruithuit

Chart, bar chart

Description automatically generated

Hình 22: Kết quả so sánh về bộ nhớ trên tập Fruithuit

Bảng 5: So sánh 2 phiên bản POERM và POERM-Op trên tập Fruithut

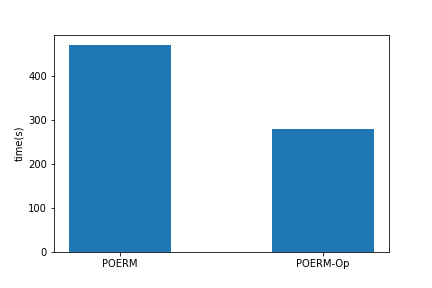
|  |  |  |
| --- | --- | --- |
| Thuật toán | Thời gian (s) | Bộ nhớ (MB) |
| POERM | 733 | 595 |
| POERM-Op | 342 | 617 |

Nhận xét:

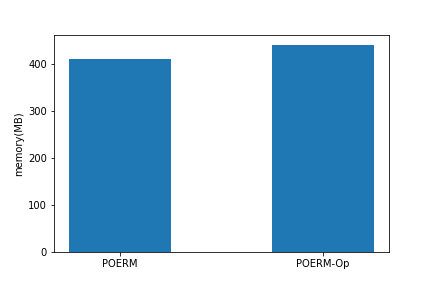
Thời gian thực thi giảm đi khoảng 1.68 lần, tuy nhiên bộ nhớ tăng không đáng kể, cụ thể tăng khoảng 22MB.

**3.4.3 Tập BMS WebView 1:**

Các tham số: minSupport = 200, minConfidence = 0.5, XSpan = YSpan = XYSpan = 5.



Hình 23: Kết quả so sánh về thời gian trên tập BMS WebView 1



Hình 24: Kết quả so sánh về bộ nhớ trên tập BMS WebView 1

Bảng 6: So sánh 2 phiên bản POERM và POERM-Op trên tập BMS Web View 1

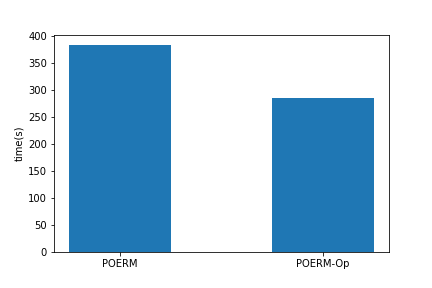
|  |  |  |
| --- | --- | --- |
| Thuật toán | Thời gian (s) | Bộ nhớ (MB) |
| POERM | 471 | 410 |
| POERM-Op | 280 | 440 |

Nhận xét:

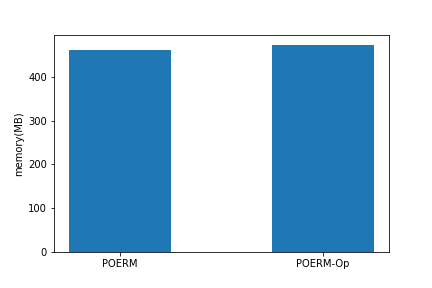
Thời gian thực thi giảm đáng kể, cụ thể giảm đi khoảng 2.14 lần, tuy nhiên bộ nhớ tăng không đáng kể, cụ thể tăng khoảng 30MB.

### **3.4.4 Tập Retail:**

Các tham số: minSupport = 200, minConfidence = 0.5, XSpan = YSpan = XYSpan = 3.



Hình 25: Kết quả so sánh về thời gian trên tập Retail



Hình 26: Kết quả so sánh về bộ nhớ trên tập Retail

Bảng 7: So sánh 2 phiên bản POERM và POERM-Op trên tập Retail

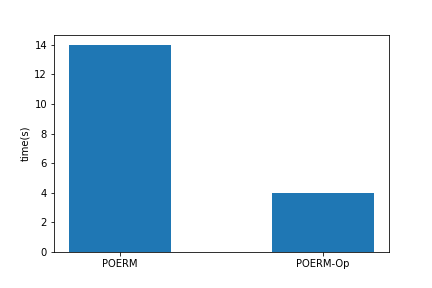
|  |  |  |
| --- | --- | --- |
| Thuật toán | Thời gian (s) | Bộ nhớ (MB) |
| POERM | 383 | 461 |
| POERM-Op | 285 | 473 |

Nhận xét:

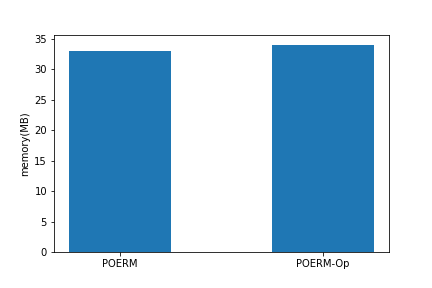
Thời gian thực thi giảm đi khoảng 1.34 lần, tuy nhiên bộ nhớ tăng không đáng kể, cụ thể tăng khoảng 12MB.

### **3.4.5 Tập Foodmart:**

Các tham số: minSupport = 5, minConfidence = 0.5, XSpan = YSpan = XYSpan = 5.



Hình 27: Kết quả so sánh về thời gian trên tập Foodmart



Hình 28: Kết quả so sánh về bộ nhớ trên tập Foodmart

Bảng 8: So sánh 2 phiên bản POERM và POERM-Op trên tập Foodmart

|  |  |  |
| --- | --- | --- |
| Thuật toán | Thời gian (s) | Bộ nhớ (MB) |
| POERM | 14 | 33 |
| POERM-Op | 4 | 34 |

Nhận xét:

Thời gian thực thi giảm đáng kể, cụ thể giảm đi khoảng 3.5 lần, tuy nhiên bộ nhớ tăng không đáng kể, cụ thể tăng khoảng 1MB.

## **3.5 Khảo sát ảnh hưởng của các tham số:**

Trong mỗi thử nghiệm, một tham số sẽ thay đổi trong khi các tham số khác là cố định. Bởi vì thuật toán có năm tham số: minSupport, minConfidence , XSpan, YSpan và XYSpan và không gian không cho phép đánh giá từng tham số riêng biệt nên ba tham số ràng buộc thời gian XSpan, YSpan và XYSpan được đặt thành cùng một giá trị được gọi là Span và sẽ chỉ khảo sát trên hai bộ dữ liệu chuẩn là Online Retail và Fruithut. Các thử nghiệm được khảo sát trên hai tập dữ liệu chuẩn OnlineRetail và Fruithut.

Quy ước:

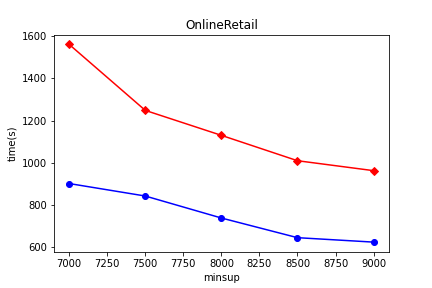
Ảnh có chứa văn bản

Mô tả được tạo tự động

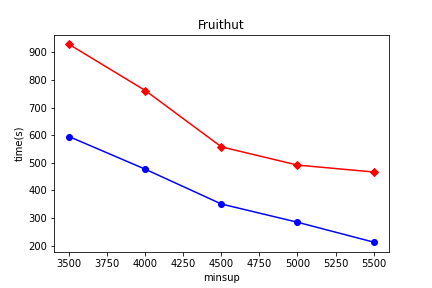
Hình 29: Quy ước về biểu đồ

### **3.5.1 Khảo sát sự ảnh hưởng về thời gian:**

**3.5.1.1 Ảnh hưởng của minSupport:**



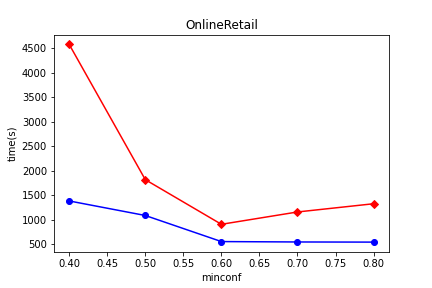
Hình 30: Ảnh hưởng của minsup đến thời gian trên tập OnlineRetail



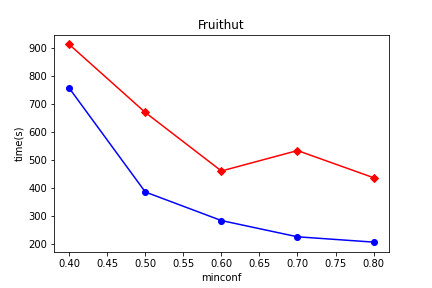
Hình 31: Ảnh hưởng của minsup đến thời gian trên tập Fruithut

Hai hình trên cho thấy khi minsup càng tăng thì thời gian thực thi càng giảm. Lý do là khi minsup càng nhỏ thì thuật toán sẽ phải xem xét nhiều tập tiền tố () và hậu tố () hơn.

**3.5.1.2** **Ảnh hưởng của minConfidence:**



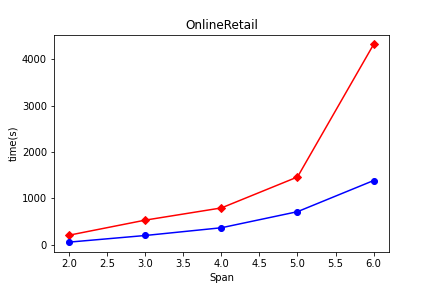
Hình 32: Ảnh hưởng của minconf đến thời gian trên tập OnlineRetail



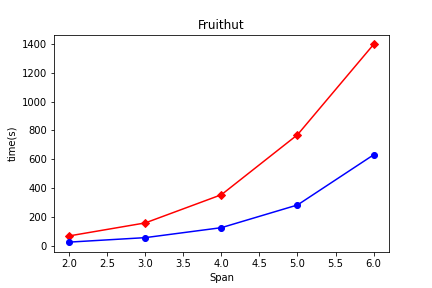
Hình 33: Ảnh hưởng của minconf đến thời gian trên tập Fruithut

Hai hình trên cho thấy khi minconf càng giảm thì thời gian thực thi càng tăng.

**3.5.1.3 Ảnh hưởng của Span:**



Hình 34: Ảnh hưởng của Span đến thời gian trên tập OnlineRetail

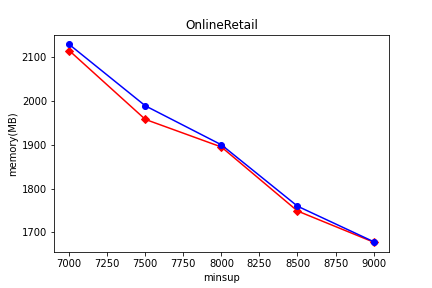


Hình 35: Ảnh hưởng của Span đến thời gian trên tập Fruithut

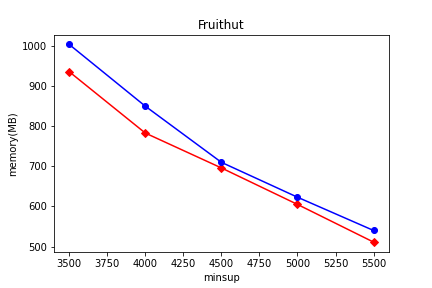
Hai hình trên cho thấy khi tham số Span sẽ tỉ lện thuận với thời gian thực thi do khi Span tăng thì thuật toán phải tìm nhiều tiền tố hơn so với trước.

### **3.5.2 Khảo sát ảnh hưởng của bộ nhớ:**

**3.5.2.1 minSupport:**



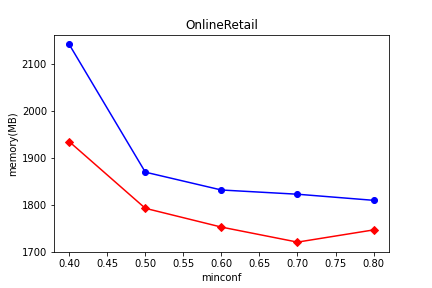
Hình 36: Ảnh hưởng của minsup đến bộ nhớ trên tập OnlineRetail



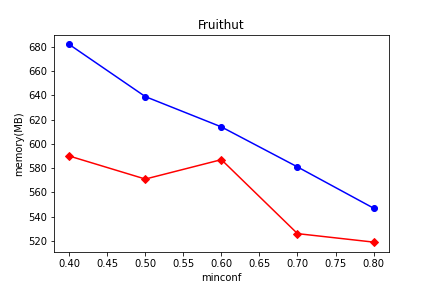
Hình 37: Ảnh hưởng của minsup đến bộ nhớ trên tập Fruithut

Khi minsup tăng lượng bộ nhớ sử dụng sẽ giảm do các sự kiện thuật toán phải xem xét sẽ ít hơn.Minsup ảnh hưởng ít đến bộ nhớ do chênh lệch về bộ nhớ giữa hai thuật toán không quá đáng kể.

**3.5.2.2 minConfidence:**



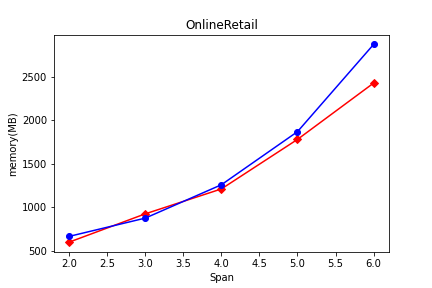
Hình 38: Ảnh hưởng của minconf đến thời gian trên tập OnlineRetail



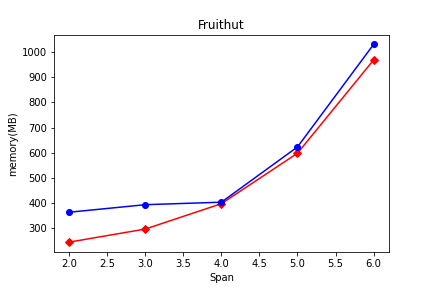
Hình 39: Ảnh hưởng của minconf đến bộ nhớ trên tập Fruithut

Nhìn chung khi minconf càng tăng thì lượng bộ nhớ sử dụng càng giảm.

**3.5.2.3 Span:**



Hình 40: Ảnh hưởng của Span đến bộ nhớ trên tập OnlineRetail



Hình 41: Ảnh hưởng của Span đến bộ nhớ trên tập Fruithut

Tương tự thời gian thực thi, khi Span tăng thì mức sử dụng bộ nhớ tăng do không gian tìm kiếm tăng lên.

### **3.5.3 Các luật được khám phá:**

Bảng 9: Ví dụ một số luật được khám phá

|  |  |  |
| --- | --- | --- |
| LUẬT | SUPPORT | CONFIDENCE  (xấp xỉ) |
| Zucchini green Field Tomatoes ==> Banana Cavendish | 5127 | 0.7138 |
| Lettuce Iceberg Field Tomatoes ==> Banana Cavendish | 5133 | 0.7162 |
| Cucumber Lebanese Field Tomatoes ==> Banana Cavendish | 6152 | 0.7064 |
| Cucumber continental Field Tomatoes ==> Banana Cavendish | 5023 | 0.7203 |
| Broccoli Field Tomatoes ==> Banana Cavendish | 5859 | 0.715 |
| Lettuce Iceberg Field Tomatoes ==> Banana Cavendish | 5133 | 0.7161 |
| Field Tomatoes Watermelon seedless ==> Banana Cavendish | 6131 | 0.7017 |
| Capsicum red Field Tomatoes Banana Cavendish ==> Banana Cavendish | 6352 | 0.7067 |
| Cucumber Lebanese Field Tomatoes Banana Cavendish ==> Banana Cavendish | 5023 | 0.7203 |

Bảng 9 là ví dụ cho thấy một số quy tắc được khám phá từ tập dữ liệu Fruithut có độ tin cậy khá cao khoảng 70% nên từ đó có thể cung cấp cái nhìn tổng quát về tập dữ liệu, giúp các cửa hàng bán lẻ hoặc công ty am hiểu về các lựa chọn của khách hàng và có thể đưa ra các phương án kinh doanh phù hợp.

**Chương 4: Kết luận và hướng phát triển**

**4.1 Kết luận:**

Trong khóa luận này, từ kết quả nghiên cứu của Philippe Fournier-Viger và cộng sự, nhóm đã nghiên cứu các loại luật mới được đề xuất gọi là các luật có thứ tự bán phần, trong đó các sự kiện trong luật được sắp xếp bán phần, dùng để tìm các tập luật tổng quát hơn.

Để tìm một cách hiệu quả tất cả các luật này trong một chuỗi, một thuật toán hiệu quả có tên là POERM – Khai thác tập luật có thứ tự bán phần đã được nghiên cứu.

Nhóm đã đề xuất cải thiện về mặt cấu trúc dữ liệu trong thuật toán để cho hiệu suất tốt hơn.

Một số đánh giá thực nghiệm trên một số tập dữ liệu cho thấy thuật toán POERM có hiệu suất tuyệt vời, đồng thời khảo sát ảnh hưởng của các tham số đến thời gian thực thi và lượng bộ nhớ đã sử dụng khi thực thi thuật toán.

**4.2 Hướng phát triển:**

Một số khả năng phát triển trong tương lai: Mở rộng POERM để xử lý dữ liệu streaming và chạy thuật toán trên dữ liệu lớn hoặc môi trường đa luồng để hưởng lợi từ tính song song; Xem xét các dữ liệu phức tạp hơn, chẳng hạn như các sự kiện được tổ chức theo một phân loại [3] hoặc một luồng [17]; Phát triển mô hình dự đoán chuỗi dựa trên các POER; Các chức năng lựa chọn mẫu khác cũng được xem xét như tiện ích [13,18] và độ hiếm [10].

# **Tài liệu tham khảo**

1. Ao, X., Luo, P., Li, C., Zhuang, F., He, Q.: Online frequent episode mining. In: *2015 IEEE 31st International Conference on Data Engineering*, pp. 891–902. IEEE,2015.
2. Ao, X., Luo, P., Wang, J., Zhuang, F., He, Q.: Mining precise-positioning episode rules from event sequences. IEEE *Transactions on Knowledge and Data Engineering* *30(3),* pp.530–543, 2017.
3. Ao, X., Shi, H., Wang, J., Zuo, L., Li, H., He, Q.: Large-scale frequent episode mining from complex event sequences with hierarchies*. ACM Transactions on Intelligent Systems and Technology (TIST) 10(4)*, pp.1–26, 2019.
4. Fahed, L., Brun, A., Boyer, A.: Deer: Distant and essential episode rules for early prediction. *Expert Systems with Applications 93*, pp.283–298, 2018.
5. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z.,Lam, H.T.: The spmf open-source data mining library version 2. In: *Joint European conference on machine learning and knowledge discovery in databases*, pp. 36–40, Springer,2016.
6. Fournier-Viger, P., Lin, J.C.W., Kiran, U.R., Koh, Y.S.: A survey of sequential pattern mining. *Data Science and Pattern Recognition 1(1)*,pp. 54–77, 2017.
7. Fournier-Viger, P., Wu, C.W., Tseng, V.S., Cao, L., Nkambou, R.: Mining partially ordered sequential rules common to multiple sequences. *IEEE Transactions on Knowledge and Data Engineering 27(8)*, pp. 2203–2216,2015.
8. Fournier-Viger, P., Yang, Y., Yang, P., Lin, J.C.W., Yun, U.: Tke: Mining top k-frequent episodes. In: *Proc. 33rd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems.* Springer, 2020.
9. Huang, K., Chang, C.: Efficient mining of frequent episodes from complex sequences. *Inf. Syst. 33(1),* pp.96–114, 2008.
10. Koh, Y.S., Ravana, S.D.: Unsupervised rare pattern mining: a survey. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10(4),* pp.1–29,2016, Mannila, H., Toivonen, H., Verkamo, A.I.: Discovering frequent episodes in sequences. In: Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining, 1995.
11. Song, W., Huang, C.: Mining high average-utility itemsets based on particle swarm optimization. *Data Science and Pattern Recognition 4(2),* pp. 19–32, 2020.
12. Su, M.Y.: Applying episode mining and pruning to identify malicious online attacks. *Computers & Electrical Engineering 59,* pp.180–188, 2017.
13. Truong, T., Duong, H., Le, B., Fournier-Viger, P.: Fmaxclohusm: An efficient algorithm for mining frequent closed and maximal high utility sequences. *Engineering Applications of Artificial Intelligence 85*, pp.1–20, 2019.
14. Wenzhe, L., Qian, W., Luqun, Y., Jiadong, R., Davis, D.N., Changzhen, H.: Mining frequent intra-sequence and inter-sequence patterns using bitmap with a maximal span. In: *Proc. 14th Web Inf. Syst. and Applications Conf*. pp. 56–61. IEEE, 2017.
15. You, T., Li, Y., Sun, B., Du, C.: Multi-source data stream online frequent episode mining. *IEEE Access 8*, pp.107465–107478, 2020.
16. Yun, U., Nam, H., Lee, G., Yoon, E.: Efficient approach for incremental high utility pattern mining with indexed list structure. *Future Generation Computer Systems 95*, 221–239, 2019.
17. Ginni:[https://www.tutorialspoint.com/why-use-support-and-confidence-in-data-mining#](https://www.tutorialspoint.com/why-use-support-and-confidence-in-data-mining), 2022.
18. Văn Thị Thiên Trang, Khai thác luật tuần từ trên cơ sở dữ liệu chuỗi, Luận văn thạc sĩ Ngành Hệ thống thông tin, Thành phố Hồ Chí Minh, 2010.