TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN VĂN PHÁT- 52100919**

**HUỜNG QUANG HUY- 52100893**

**HUỲNH GIA HUY - 52100919**

**DESIGN AND ANALYSIS OF ALGORITHM FOR DISCOVERING FREQUENT WEIGHTED ITEMSET**

**BÁO CÁO CUỐI KÌ**

**MÔN PHÂN TÍCH VÀ THIẾT KẾ**

**GIẢI THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN VĂN PHÁT- 52100919**

**HUỜNG QUANG HUY- 52100893**

**HUỲNH GIA HUY - 52100919**

**DESIGN AND ANALYSIS OF ALGORITHM FOR DISCOVERING FREQUENT WEIGHTED ITEMSET**

**BÁO CÁO CUỐI KÌ**

**MÔN PHÂN TÍCH VÀ THIẾT KẾ**

**GIẢI THUẬT**

Người hướng dẫn

**TS. Nguyễn Chí Thiện**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn thầy Nguyễn Chí Thiện đã hỗ trợ hết mình trong thời gian chúng em làm báo cáo này, thầy đã chia sẻ các kiến thức thuộc môn học này đầy nhiệt huyết và liên kết các kiến thức trong môn học với thực tiễn.

Em cũng xin cảm ơn Trường Đại học Tôn Đức Thắng đã tạo điều kiện cho chúng em có thể học tập trực tiếp.

Cuối lời, chúng em xin chúc thầy cùng tập thể giáo viên có nhiều sức khỏe và hạnh phúc.

*TP. Hồ Chí Minh, ngày 14 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Trần Văn Phát*

*Hường Quang Huy*

*Huỳnh Gia Huy*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của chúng tôi và được sự hướng dẫn khoa học của TS. Nguyễn Chí Thiện. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 14 tháng 12 năm 2023*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Trần Văn Phát*

*Hường Quang Huy*

*Huỳnh Gia Huy*

**DESIGN AND ANALYSIS OF ALGORITHM FOR DISCOVERING FREQUENT WEIGHTED ITEMSET**

**TÓM TẮT**

Trong bài báo cáo này chúng em đã code lại các thuật toán có trong bài báo “Efficient weighted probabilistic frequent itemset mining in uncertain databases” bằng ngôn ngữ java.

Các phần chính của thuật toán được thể hiện trong file Apriori.java đính kèm. Đồng thời, chúng em cũng thực hiện việc thí nghiệm với số lượng tập uncertain dataset tăng dần.

Với đầu vào là một uncertain dataset, một weight table, một mean of size-1-itemset, một probabilistic threshold, một minimum support. Thuật toán sẽ cho đầu ra sẽ là tập hợp các itemset thỏa điều kiện là weighted probablistic frequent itemset với kích thước của từng itemset là giống nhau và đạt tối đa.

**DESIGN AND ANALYSIS OF ALGORITHM FOR DISCOVERING FREQUENT WEIGHTED ITEMSET**

**ABSTRACT**

In this final report, we coded the algorithms that appeared in the article “Efficient weighted probabilistic frequent itemset mining in uncertain databases” by java language programming.

The main part of algorithm was coded in file Apriori.java. In addition, we also experimented with increasing size of uncertain dataset.

The inputs contained an uncertain dataset, a weight table, a mean of size-1-itemset, a probabilistic threshold, a minimum support. The algorithm will return the ouput that is the set of itemsets that satisfied the weighted probablistic frequent itemset with the size of itemsets are the same and maximum.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vii](#_Toc154051148)

[CHƯƠNG 1. INTRODUCTION 1](#_Toc154051149)

[CHƯƠNG 2. PROBLEM STATEMENT 3](#_Toc154051150)

[CHƯƠNG 3. DEFINITION INPUT AND OUTPUT 5](#_Toc154051151)

[3.1 Định nghĩa input 5](#_Toc154051152)

[3.2 Định nghĩa ouput 6](#_Toc154051153)

[CHƯƠNG 4. METHODS 7](#_Toc154051154)

[4.1 Sơ lược phần phân tích 7](#_Toc154051155)

[4.2 Phân tích các phương thức bổ trợ cho phương thức sovle() 7](#_Toc154051156)

[4.2.1 Phương thức getWeight(Set<T> X) 7](#_Toc154051157)

[4.2.2 Phương thức isSubset(Set<T> X, Map<T, Double> Ti) 8](#_Toc154051158)

[4.2.3 Phương thức probXInTi(Set<T> X, Map<T, Double> Ti) 9](#_Toc154051159)

[4.2.4 Phương thức probXInUD(Set<T> X, int msup, double t, double wX) 10](#_Toc154051160)

[4.2.5 Phương thức isWPFI(double wX, double probXInUD, double t) 12](#_Toc154051161)

[4.2.6 Phương thức itemsInPrevWPFI(Set<Set<T>> WPFI) 12](#_Toc154051162)

[4.2.7 Phương thức genSize1WPFI() 13](#_Toc154051163)

[4.2.8 Phương thức scanFindKItemset() 14](#_Toc154051164)

[4.2.9 Phương thức genWPFIApriori() 15](#_Toc154051165)

[4.3 Phương thức chính của thuật toán Apriori, phương thức solve() 19](#_Toc154051166)

[CHƯƠNG 5. EXPERIMENT 22](#_Toc154051167)

[5.1 Sinh bộ dữ liệu và kết quả khi thực thi thuật toán 22](#_Toc154051168)

[5.1.1 Sinh bộ dữ liệu 22](#_Toc154051169)

[5.1.2 Kết quả khi thực thi thuật toán 24](#_Toc154051170)

[5.2 Vẽ biểu đồ theo phân tích lý thuyết của phương thức solve() thực thi 24](#_Toc154051171)

[5.3 Vẽ biểu đồ theo thực tế phương thức solve() thực thi 26](#_Toc154051172)

[CHƯƠNG 6. KẾT LUẬN 27](#_Toc154051173)

[TÀI LIỆU THAM KHẢO 28](#_Toc154051174)

# DANH MỤC HÌNH VẼ

[Ảnh 3.1: Ví dụ về uncertain dataset với T là Integer. 5](#_Toc154051175)

[Ảnh 3.2: Ví dụ về weight table với T là Integer 5](#_Toc154051176)

[Ảnh 3.3: Ví dụ về mean của các size-1-itemset. 6](#_Toc154051177)

[Ảnh 3.4: Ví dụ về output với T là Integer. 6](#_Toc154051178)

[Ảnh 4.1: Phần code phương thức getWeight() 7](#_Toc154051179)

[Ảnh 4.2: Phần code phương thứcn isSubset 8](#_Toc154051180)

[Ảnh 4.3: Ví dụ về phương thức probXInTi() 9](#_Toc154051181)

[Ảnh 4.4: Phần code phương thức probXInUD() 10](#_Toc154051182)

[Ảnh 4.5: Bảng quy hoạch động cho mô tả lại thuậtt toán trên 11](#_Toc154051183)

[Ảnh 4.6: Bảng quy hoạch động thể hiện việc tỉa 11](#_Toc154051184)

[Ảnh 4.7: Phần code phương thức isWPFI() 12](#_Toc154051185)

[Ảnh 4.8: Phần code phương thức itemsInPrevWPFI() 13](#_Toc154051186)

[Ảnh 4.9: Phần code phương thức genSize1WPFI() 13](#_Toc154051187)

[Ảnh 4.10: Phần code phương thức scanFindKItemset() 14](#_Toc154051188)

[Ảnh 4.11: Mã giả cho phương thức genWPFIApriori() 16](#_Toc154051189)

[Ảnh 4.12: Phần code đầu của phương thúc genWPFIAPriori() 17](#_Toc154051190)

[Ảnh 4.13: Phần code tiếp theo của phương thức genWPFIApriori() 18](#_Toc154051191)

[Ảnh 4.14: Mã giả cho phương thức solve() 19](#_Toc154051192)

[Ảnh 4.15: Phần code phương thức solve() 20](#_Toc154051193)

[Ảnh 5.1: Lệnh sinh ra Uncertain datatset, Weight table, Mean size-1-itemsets 22](#_Toc154051194)

[Ảnh 5.2: Uncertain dataset khi được sinh ra 24](#_Toc154051195)

[Ảnh 5.3: Weight table khi được sinh ra 24](#_Toc154051196)

[Ảnh 5.4: Mean size-1-itemsets khi được sinh ra 24](#_Toc154051197)

[Ảnh 5.5: Lưạ chọn các tham số cho thuật toán Apriori và thực thi bằng cách gọi phương thức solve() 24](#_Toc154051198)

[Ảnh 5.6: Kết qủa sau khi thực thi 24](#_Toc154051199)

[Ảnh 5.7: Kết quả thực thi khi phân tích theo lý thuyết 25](#_Toc154051200)

[Ảnh 5.8: Biểu đồ thời gian chạy khi phân tích theo lý thuyết 25](#_Toc154051201)

[Ảnh 5.9: Kết quả thời gian chạy theo thực tế 26](#_Toc154051202)

[Ảnh 5.10: Biểu đồ thời gian chạy theo thực tế 26](#_Toc154051203)

# INTRODUCTION

Khám phá các tập phổ biến có trọng số là một nhiệm vụ cơ bản trong khai thác dữ liệu và có ứng dụng trong nhiều lĩnh vực khác nhau như phân tích giỏ hàng thị trường, hệ thống khuyến nghị và phân tích lưu lượng mạng. Không giống như khai thác tập mục thường xuyên truyền thống, trong đó mỗi mục có cùng trọng số, khai thác tập mục thường xuyên có trọng số xem xét tầm quan trọng hoặc tầm quan trọng của từng mục trong tập dữ liệu. Việc gán trọng số này cho phép chúng tôi khám phá các mẫu có ý nghĩa phản ánh sở thích, mức độ ưu tiên hoặc điểm mạnh cơ bản liên quan đến từng mục.

Trong bài nghiên cứu này, chúng tôi đi sâu vào chủ đề khám phá các tập phổ biến có trọng số và khám phá các thuật toán và kỹ thuật khác nhau đã được phát triển để giải quyết vấn đề này. Chúng tôi thảo luận về những thách thức và cơ hội liên quan đến việc khai thác các tập mục có trọng số và nêu bật những lợi ích của việc kết hợp trọng số vào quá trình khai thác. Bằng cách xem xét trọng lượng của các mặt hàng, chúng tôi có thể khám phá những hiểu biết có giá trị và tạo ra kết quả chính xác và có ý nghĩa hơn.

Hơn nữa, chúng tôi kiểm tra các ứng dụng và trường hợp sử dụng khác nhau trong đó việc khai thác tập mục có trọng số thường xuyên đóng một vai trò quan trọng. Chúng tôi giới thiệu cách áp dụng kỹ thuật này trong các lĩnh vực khác nhau để khám phá các mẫu, mối liên kết và sự phụ thuộc có thể không rõ ràng khi chỉ xem xét tần suất của các mục. Bằng cách khám phá những mối quan hệ tiềm ẩn này, các tổ chức có thể đưa ra quyết định sáng suốt, tối ưu hóa quy trình và cải thiện hiệu suất tổng thể của mình.

Thông qua bài nghiên cứu này, chúng tôi mong muốn cung cấp một cái nhìn tổng quan toàn diện về lĩnh vực khám phá các tập phổ biến có trọng số, bao gồm cả nền tảng lý thuyết và cách triển khai thực tế. Chúng tôi khám phá các thuật toán, phương pháp và số liệu đánh giá tiên tiến nhất được sử dụng trong lĩnh vực này và thảo luận về những thách thức hiện tại cũng như hướng nghiên cứu trong tương lai. Bằng cách hiểu được sự phức tạp của việc khai thác các tập mục có trọng số, các nhà nghiên cứu và người thực hành có thể tận dụng kiến thức này để rút ra những hiểu biết có giá trị từ bộ dữ liệu của họ và đạt được lợi thế cạnh tranh trong các lĩnh vực tương ứng của họ.

# PROBLEM STATEMENT

Vấn đề phát hiện các tập phổ biến có trọng số phát sinh khi các kỹ thuật khai thác tập phổ biến truyền thống không còn phù hợp do thiếu các trọng số bằng nhau được gán cho mỗi mục. Trong nhiều tình huống thực tế, tầm quan trọng hoặc tầm quan trọng của các mục trong tập dữ liệu có thể khác nhau và việc xem xét các trọng số này trở nên quan trọng để phân tích và ra quyết định chính xác.

Thách thức chính nằm ở việc xác định một cách hiệu quả các tập mục xuất hiện đủ thường xuyên để được coi là quan trọng, có tính đến trọng số liên quan của chúng. Không giống như khai thác tập mục truyền thống, nơi chỉ tập trung vào tần suất, việc khai thác tập mục có trọng số yêu cầu một cách tiếp cận khác kết hợp thông tin trọng số. Vấn đề này càng trở nên phức tạp hơn khi kích thước tập dữ liệu tăng lên và số lượng mục cũng như trọng số liên quan của chúng tăng lên.

Ngoài ra, cần phát triển các thuật toán và kỹ thuật có thể xử lý nhiều loại phép gán trọng số khác nhau, chẳng hạn như giá trị liên tục hoặc rời rạc, trọng số dương hoặc âm và mức độ quan trọng khác nhau. Quá trình khai thác phải có khả năng xử lý các phân bổ trọng lượng đa dạng này và trích xuất các mẫu và liên kết có ý nghĩa phản ánh mối quan hệ cơ bản giữa các mục.

Một thách thức khác là xác định ngưỡng hoặc giá trị hỗ trợ thích hợp để xác định tầm quan trọng của tập mục có trọng số. Các biện pháp hỗ trợ truyền thống chỉ dựa trên tần suất có thể không phù hợp trong bối cảnh này vì chúng không xem xét đến trọng lượng khác nhau của các hạng mục. Do đó, việc xác định một biện pháp hỗ trợ hiệu quả kết hợp các trọng số mục là điều cần thiết để khai thác tập mục có trọng số thường xuyên chính xác và có ý nghĩa.

Hơn nữa, khả năng mở rộng là mối quan tâm đáng kể khi xử lý các tập dữ liệu lớn và phân bổ trọng số phức tạp. Cần có các thuật toán và kỹ thuật tối ưu hóa hiệu quả để xử lý độ phức tạp tính toán liên quan đến việc khai thác các tập mục có trọng số thường xuyên. Quá trình khai thác phải có khả năng xử lý kích thước ngày càng tăng và độ phức tạp của bộ dữ liệu trong khi vẫn duy trì thời gian thực hiện hợp lý.

Việc giải quyết những thách thức này và phát triển các giải pháp hiệu quả để khám phá các tập mục có trọng số thường xuyên là rất quan trọng để cho phép các tổ chức thu được những hiểu biết có giá trị từ dữ liệu của họ. Bằng cách khám phá các mô hình và mối liên hệ có ý nghĩa liên quan đến trọng lượng của các mặt hàng, doanh nghiệp có thể đưa ra quyết định sáng suốt, cải thiện quy trình và nâng cao hiệu suất tổng thể của mình.

# DEFINITION INPUT AND OUTPUT

## Định nghĩa input

UD: uncertain dataset (bộ dữ liệu không chắc chắn) có kiểu dữ liệu List<Map<T, Double>>.

|  |
| --- |
| Uncertain Dataset UD:  {5=0.6536502707047002, 10=0.1}  {18=0.38427311120601937, 4=0.5729134224539177}  {1=1.0, 3=0.7993203318668238, 4=0.8896676089671387, 8=0.13979915595230485}  {2=0.1811806960117825, 18=0.47162335370355196, 3=0.7541895392522893, 14=0.1}  {7=0.17167431695360585, 13=0.47426824263626466, 14=0.9195431726812944} |

Ảnh .: Ví dụ về uncertain dataset với T là Integer.

W: weight table, bảng tra cứu weight của từng item, có kiểu dữ liệu Map<T, Double>.

|  |
| --- |
| Weight table W:  {19=0.7130295873797434, 4=0.6195098055180615, 20=0.13890321239813663, 5=0.7850104461075407, 8=0.8458882225403455} |

Ảnh .: Ví dụ về weight table với T là Integer

UI: mean của các size-1-itemset, hỗ trợ cho việc tính toán trong hàm sinh các candidates, kiểu dữ liệu Map<T, Double>. Ví dụ về weight table với T là Integer.

|  |
| --- |
| Mean size-1-itemsets UI:  {1=3.6168538989072925, 2=1.3000000000000003, 3=2.0835408156233894, 4=2.814490031790164, 5=3.650675384619278, 7=2.8675217048648265, 8=4.231356127843847, 9=3.3604552818464395, 10=3.1742957991907823, 11=2.3052713053095477, 12=0.8662619551719937, 13=2.6842255786783817, 14=0.18859654113953112, 15=3.1880434874021977, 16=1.3884137203509468, 17=4.695694982546859, 18=4.296524030996622, 19=4.072722266730358, 20=1.2921556650920079} |

Ảnh .: Ví dụ về mean của các size-1-itemset.

msup: minimum support, kiểu dữ liệu int.

n: số lượng transaction có trong Uncertain dataset, kiểu dữ liệu int

alpha: scale factor, tham số giúp cho việc tỉa nhánh, kiểu dữ liệu double

t: probabilistic threshold (ngưỡng xác suất), dùng để xác định xem một itemset có phải là weighted probabilistic frequent itemset (wPFI) hay không, kiểu dữ liệu double.

Các input này sẽ được lưu thành các thuộc tính của instance Apriori. Ngoài ra, trong class Apriori còn có các thuộc tính như uhat, WPFIK, UWPFIK, UI và I, được sinh ra khi thêm các input vừa đề cập cho instance Apriori, các thuộc tính này hỗ trợ cho việc tính toán của các phương thức trong class Apriori.

## Định nghĩa ouput

Output có kiểu dữ liệu Set<Set<T>>, là tập các wPFI có cùng kích thước với nhau.

|  |
| --- |
| [[10, 11, 13, 14], [18, 19, 12, 15], [16, 18, 4, 12], [16, 3, 4, 12], [18, 3, 4, 12], [11, 13, 14, 15], [1, 19, 4, 15], [4, 12, 14, 15], [7, 11, 13, 15]] |

Ảnh .: Ví dụ về output với T là Integer.

# METHODS

## Sơ lược phần phân tích

Class Apriori<T> phục vụ chính là tìm ra các itemset là wPFI dựa vào các input được đề cập trong định nghĩa input. Các input này sẽ là argument dành cho việc tạo instance của Apriori. Đối với việc tìm ra các wPFI, chỉ cần gọi phương thức solve() và phương thức solve() này cần nhiều phương thức khác có trong class Apriori để tìm ra wPFI. Vì thế độ phức tạp của hàm solve() cần phải dựa vào các phương thức bổ trợ cho nó.

Với sự bổ trợ các phương thức khác, phương thức solve() có độ phức tạp là , chi tiết phần phân tích sẽ được đề cập ở phía dưới.

## Phân tích các phương thức bổ trợ cho phương thức sovle()

### Phương thức getWeight(Set<T> X)

A screenshot of a computer code

Description automatically generated

Ảnh .: Phần code phương thức getWeight()

* Mô tả đầu vào và chức năng của phương thức:

Phương thức này nhận đầu vào là một itemset. Trong xuyên suốt phần cài đặt, itemset luôn được định nghĩa theo kiểu Set<T>.

Phương thức này dùng để tính weight của một itemset theo công thức:

* Input size:

với là kích thước của itemset .

* Basic operation:

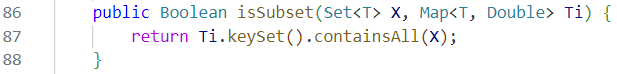
Dòng số 68

* Worst case:

Không có worst case.

Do vòng for duyệt hết các item có trong itemset X, nên độ phức tạp của thuật toán này luôn là .

### Phương thức isSubset(Set<T> X, Map<T, Double> Ti)



Ảnh .: Phần code phương thứcn isSubset

* Mô tả đầu vào và chức năng của phương thức:

Phương thức này nhận đầu vào là một itemset Set<T> X và một transaction Map<T, Double> Ti. Trong xuyên suốt phần cài đặt, transaction được định nghĩa kiểu Map<T, Double>.

Phương thức này dùng để kiểm tra xem itemset X có phải là tập con của transaction Ti hay không.

* Input size:

với là kích thước của itemset X.

là kích thước của transaction Ti.

* Basic operation:

Dòng số 87

* Worst case:

Không có worst case.

Khi cài đặt bằng brute-force, vòng lặp bên ngoài dành cho X, vòng lặp bên trong dành cho Ti, hai vòng lặp này để kiểm tra itemset X có phải là tập con của transaction Ti hay không. Nên độ phức tạp của thuật toán sẽ là

Tuy nhiên, do phương thức này được cài đặt bằng cách tận dụng các phương thức của Map nên thực tế độ phức tạp hay thời gian thực thi có thể nhỏ hơn kết quả vừa phân tích, với phương thức keySet() có độ phức tạp là và containsAll() là vì thế phương thức này có độ phức tạp là

### Phương thức probXInTi(Set<T> X, Map<T, Double> Ti)

A screen shot of a computer code

Description automatically generated

Ảnh .: Ví dụ về phương thức probXInTi()

* Mô tả đầu vào và chức năng của phương thức:

Một itemset Set<T> X, một transaction Map<T, Double> Ti.

Phương thức này dùng để tính xác suất của itemset X là tập con của transaction Ti hay .

* Input size:

với là kích thước itemset X.

là kích thước của transaction Ti.

* Basic operation:

Dòng số 104 và 106.

* Worst case:

Không có worst case.

Phương thức này dùng lại phương thức isSubset(), có độ phức tạp là để kiểm tra điều kiện và vòng lặp để tính toán xác suất của itemset có độ phức tạp là . Tóm lại, độ phức tạp của cả phương thức này là .

### Phương thức probXInUD(Set<T> X, int msup, double t, double wX)

A screenshot of a computer code

Description automatically generated

Ảnh .: Phần code phương thức probXInUD()

* Mô tả đầu vào và chức năng của phương thức:

Nhận một itemset Set<T> X, một minimum support int , probablistic threshold double . và cũng chính là thuộc tính của class Apriori.

Phương thức này giúp cho việc tính xác suất của itemset lớn hơn hoặc bằng trong uncertain dataset bằng dynamic programming có kèm theo cách tỉa, nhằm kết thúc thuật toán sớm, đồng thời tính cả tổng xác suất của itemset trong uncertain dataset.

Phương thức này được cài đặt theo công thức truy hồi dưới đây, với kết quả là :

Trong đó:

|  |  |
| --- | --- |
|  | Là chỉ số cho transaction có trong uncertain dataset. |
|  | Là chỉ số cho msup, đi từ 0 đến msup    Ảnh .: Bảng quy hoạch động cho mô tả lại thuậtt toán trên  (Nguồn: [1]) |

Để cho việc tính toán trở nên nhanh hơn trong bài báo [1] có giới thiệu một số cách tỉa nhằm kết thúc thuật toán sớm như sau:

* Với mỗi lần lặp , chỉ cần chạy từ 0 đến n – msup +
* Có thể dừng thuật toán khi , với điều kiện

A graph with lines and numbers

Description automatically generated

Ảnh .: Bảng quy hoạch động thể hiện việc tỉa

(Nguồn: [1])

* Input size:

, với là kích thước của itemset .

Phương thức còn sử dụng một thuộc tính của instance Apriori là n, với n là

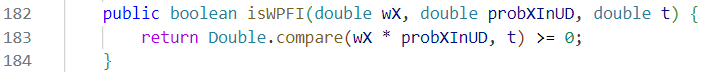
* Basic operation:

Dòng số 146 và 148.

* Worst case:

Trường hợp tệ nhất vòng lặp ở dòng 146 phải lặp đến msup lần và vòng lặp ở dòng 148 lặp n lần. Nên độ phức tạp lúc này là .

### Phương thức isWPFI(double wX, double probXInUD, double t)



Ảnh .: Phần code phương thức isWPFI()

* Mô tả đầu vào và chức năng của phương thức:

wX là weight của itemset X có kiểu là double, probXInUD chính là có kiểu là double, t là probablistic threshold có kiểu số thực.

Phương thức này đơn giản cho biết itemset có phải là wPFI hay không bằng cách trả về true hoặc false.

* Input size:

Không có

* Basic operation:

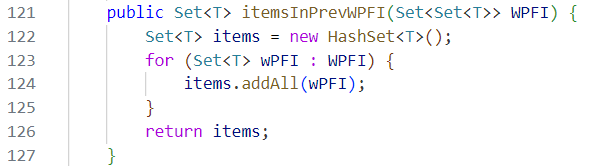
Không có

* Worst case:

Không có.

Độ phức tạp của phương thức này là .

### Phương thức itemsInPrevWPFI(Set<Set<T>> WPFI)



Ảnh .: Phần code phương thức itemsInPrevWPFI()

* Mô tả đầu vào và chức năng của phương thức:

Một ArrayList<Set<T>> có thể hiểu là tập hợp các itemset thật sự là wPFI.

Hàm này đơn giản là tìm tập các item có trong wPFI hiện tại.

* Input size:

, với là kích thước của tham số WPFI, là kích thức của tất cả size-k-itemset có trong WPFI.

* Basic operation:

Dòng số 123 và 124.

* Worst case:

Không có worst case.

Do tận dụng phương thức addAll() của Set, có độ phức tạp là và duyệt qua tất cả các size-1-itemset, có độ phức tạp là . Nên độ phức tạp của thuật toán .

### Phương thức genSize1WPFI()

A screen shot of a computer code

Description automatically generated

Ảnh .: Phần code phương thức genSize1WPFI()

* Mô tả đầu vào và chức năng phương thức:

Phương thức này sử dụng thuộc tính của class Apriori, là tập hợp các item có trong . Phương thức này lấy biến WPFI1 gán cho thuộc tính hay tập hợp các size-k-itemset hiện tại.

* Input size:

với là kích thước của

* Basic operation:

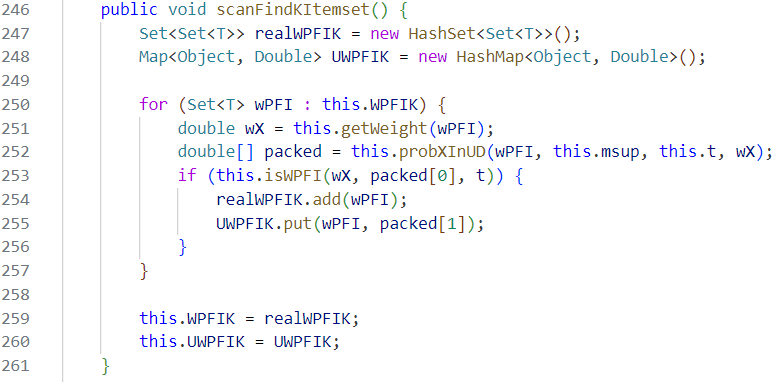
Dòng số 218.

* Worst case:

Không có worst case.

Do phương thức duyệt qua tất cả các item có trong thuộc tính , nên độ phức tạp là

### Phương thức scanFindKItemset()



Ảnh .: Phần code phương thức scanFindKItemset()

* Mô tả đầu vào và chức năng của phương thức:

Đầu vào của phương thức này là tận dụng thuộc tính msup, t, và WPFIK hay tập hợp các size-k-itemset hiện tại có trong instance Apriori.

Phương thức này sẽ duyệt hết các wPFI trong WPFIK, nếu wPFI thỏa điều kiện thì sẽ thêm vào realWPFI. Đồng thời thêm các mean của wPFI thỏa điều kiện vào UWPFI. Và cuối cùng là đặt lại WPFIK là realWPFI, đồng thời gán giá trị mới cho thuộc tính UWPFIK.

* Input size:

Có thể xem input của phương thức này là các thuộc tính msup, t,và WPFIK.

Trong đó là kích thước của WPFIK

* Basic operation:

Dòng 243, 244 và 245.

* Worst case:

Phụ thuộc vào phương thức probXInUD().

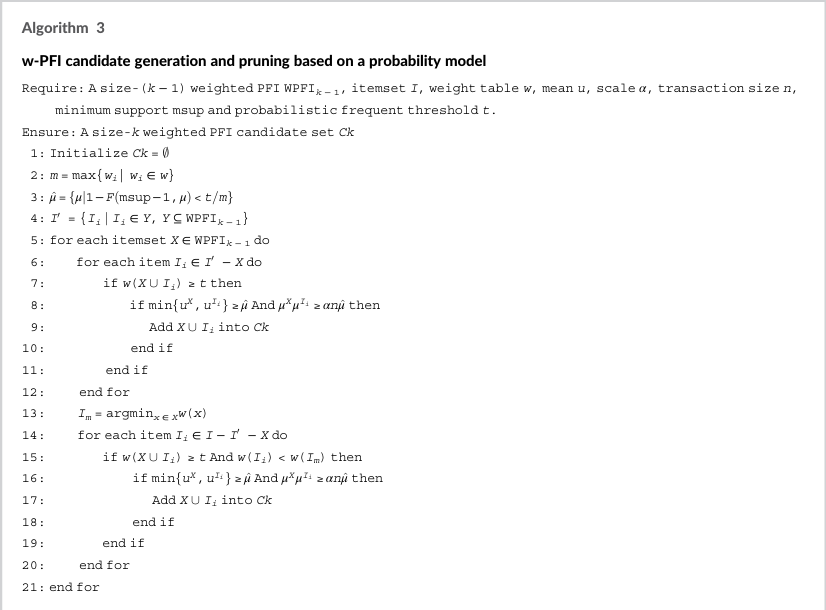
Vòng lặp sẽ lặp lần do phải duyệt hết các wPFI trong vì thế độ phức tạp là

Phương thức getWeight() có độ phức tạp là , với là kích thước của các wPFI trong .

Phương thức proXInUD() có độ phức tạp của worst case là .

Tóm lại, phương thức này có độ phức tạp là (nếu như xem và lớn hơn nhiều so với các số còn lại).

### Phương thức genWPFIApriori()



Ảnh .: Mã giả cho phương thức genWPFIApriori()

A screenshot of a computer program

Description automatically generated

Ảnh .: Phần code đầu của phương thúc genWPFIAPriori()

A screenshot of a computer program

Description automatically generated

Ảnh .: Phần code tiếp theo của phương thức genWPFIApriori()

* Mô tả đầu vào và chức năng của phương thức:

Phương thức này tận dụng các thuộc tính của instance Apriori làm đầu vào bao gồm: uhat, alpha, n, msup, t, WPFIK, UWPFIK, UI.

Phương thức này dựa vào mã giả để sinh ra các itemset tiếp theo có kích thức tăng lên một đơn vị.

* Input size:

là kích thước của tập hợp size-k-itemset cũng như .

là thuộc tính của instance Apriori, biểu thị kích thước của uncertain dataset .

* Basic operation:

Dòng 275, 277, 285, 315.

* Worst case:

Không có worst case.

Dòng 275, tìm các item có trong hiện tại, có độ phức tạp , là size hiện tại của các itemset trong .

Dòng 277, duyệt qua tất cả wPFI trong , có độ phức tạp

Dòng 285, duyệt qua tất cả các item trong Ia – X, cho trung bình kích thước Ia – X là độ phức tạp khi thực hiện là

Dòng 315, duyệt qua tất cả các item trong I – Ia - X, cho trung bình kích thước I – Ia - X là độ phức tạp khi thực hiện là

Tóm lại, phương thức này có độ phức tạp là

(nếu như xem và lớn hơn nhiều so với các số còn lại).

## Phương thức chính của thuật toán Apriori, phương thức solve()

A white text with black text

Description automatically generated

Ảnh .: Mã giả cho phương thức solve()

A screenshot of a computer code

Description automatically generated

Ảnh .: Phần code phương thức solve()

* Mô tả đầu vào và chức năng của phương thức:

Phương thức này tận dụng các thuộc tính của instance Apriori để làm đầu vào.

Phương thức này có nhiệm vụ tìm ra tập hợp các itemset là wPFI và có size đạt tối đa.

* Input size:

là kích thước của tập hợp size-k-itemset cũng như .

là thuộc tính của instance Apriori, biểu thị kích thước của uncertain dataset

* Basic operation:

Dòng 359, 360 và 361.

* Worst case:

Phụ thuộc vào dòng 352.

Dòng 359, khi n + 1, tức là kết quả trả về của phương thức này là một tập hợp chứa một tập hợp tất cả các item có trong UD.

Dòng 360, phương thức genWPFIApriori(), có độ phức tạp là

Dòng 361, phương thức scanFindKItemset(), có độ phức tạp là .

Tóm lại, phương thức này có độ phức tạp trường hợp worst case là

# EXPERIMENT

## Sinh bộ dữ liệu và kết quả khi thực thi thuật toán

### Sinh bộ dữ liệu

Vì lý do không gian trình bài, ở đây chúng tôi chỉ có thể sinh ra bộ dữ liệu có 30 transactions, với kích thước transaction nằm trong khoảng từ 3 đến 20.

A close-up of a computer screen

Description automatically generated

Ảnh .: Lệnh sinh ra Uncertain datatset, Weight table, Mean size-1-itemsets

|  |
| --- |
| {16=0.4343089922507179, 2=0.463223854706176, 18=0.29844949115426134, 6=0.74819843387256, 7=0.35107364065390595, 9=0.8393897618166071, 12=0.32529382372621785, 13=0.2764813249279423}, {2=0.41533148185765734, 3=0.5335752806193016, 4=0.7632804271192443, 5=0.4792103765092857, 6=0.6334415580578474, 7=0.39974635328699, 8=0.1, 10=0.4926057377731124, 11=0.32613686800130703, 12=0.6772725221561255, 14=0.6034624741161424, 17=0.1, 18=0.4141548064731761, 20=1.0}, {16=0.7463986743778437, 18=0.17022478813109715, 6=0.5165057760061147, 7=0.7247392846707543, 8=0.13459227552729086, 10=0.9051627120136001}, {16=1.0, 17=0.7244773515490246, 1=1.0, 2=0.23731190503052696, 19=0.8707225200902007, 3=0.836147277314512, 7=0.25941330653483835, 8=0.5719699819389901, 10=0.1, 15=0.757082638726758}, {1=0.2530120895412218, 18=0.500460540338406, 3=0.1, 20=0.5771071414309304, 6=0.38731216154776443, 15=0.1}, {1=0.1, 18=0.9384915653033572, 20=0.1, 6=0.4676271494790903, 7=0.2869953505844114, 8=0.1, 9=0.5659823615576997, 10=0.1, 12=0.1, 15=0.1}, {1=0.9107468150093077, 4=0.8730283654402409, 5=0.32853245011303656, 7=0.6105579644859014, 10=0.4904967223511343, 12=0.1, 13=0.1, 14=0.5078123126078224, 15=0.8665642246547276}, {16=0.9789262922575546, 1=0.6301984763949088, 2=0.4151009483812223, 5=0.3570691864178733, 6=0.28714809934465463, 8=0.5672656317093514, 9=0.6737413509191651, 11=0.5531268641266187, 12=0.4275313415052458, 13=0.5534743113419976, 15=0.9826052666479914}, {16=0.5772442813268532, 18=0.44963906558802463, 3=0.32191012143281916, 8=0.5864091215194734, 12=0.4352284220021878, 13=0.1}, {1=0.49547453065935654, 18=0.16416137539699127, 20=0.23523853760822977, 4=0.40649122624932243, 5=0.8810609076819756, 6=1.0, 9=0.2234870849332759, 12=1.0, 14=0.1}, {1=0.3175871933833575, 4=0.9888872155893571, 7=0.8805661862877735, 8=0.5016001331408717, 10=0.6562409503566915}, {17=0.5740721575673136, 19=0.5947135708417284, 5=0.6558883405957541, 13=0.9375201250661225, 14=0.6440131975969854}, {1=0.47481289310167235, 3=0.1, 11=0.5857344456531287}, {2=0.9007138271496957, 19=0.1, 20=0.6808135527865093, 9=0.1}, {16=0.2352017497965363, 2=0.10050407880171841, 5=0.8777665677554394, 7=0.2852316258227561}, {2=0.9429021722563725, 19=0.585575887331818, 12=0.34734190147806465}, {16=0.37784893178959916, 2=0.5331924487190567, 20=0.6276031782901018, 4=0.10320984419744594, 6=0.5775164086506075, 8=0.25374934015535344, 10=0.4321212911842155, 11=0.1, 13=0.38844125636353094}, {1=0.36732984502182625, 2=0.525129743727016, 3=0.7902532152181276, 20=1.0, 4=0.551399764022791, 5=0.3761480318347783, 6=0.7900408646450281, 8=0.19893244463728732, 12=1.0, 13=0.735147226065208}, {7=0.8908008020675334, 10=0.3263079375736372, 11=0.34216500098120317}, {16=0.1581762007790169, 1=0.5469906910145269, 18=0.49299443443688257, 3=1.0, 5=0.6267808646024755, 6=0.1, 9=1.0, 11=0.1, 12=0.1, 13=0.33805938870242236, 14=0.439639765347537}, {16=0.4471225361984901, 1=0.31161698181520897, 18=0.6221226413166827, 3=0.5354633639676666, 19=0.3221826254417913, 6=0.1962714527265978, 8=1.0, 10=0.5919825918785756, 11=0.1, 12=0.1}, {19=1.0, 3=0.3950517719342931, 4=0.4568557877705702, 6=0.17381534173743096, 7=0.17044581978175444, 9=0.1, 10=0.1, 11=0.8530443559169463, 12=0.49413331592215376, 13=0.9817377222920376, 14=0.3598629622568478}, {5=0.31773059024254857, 6=1.0, 14=0.30725876106845706}, {16=0.9758716599833219, 17=0.44595613080724894, 3=0.1, 5=1.0, 6=1.0, 9=1.0, 13=0.5181870735567264, 14=0.56598562243081}, {1=0.8168962138117664, 2=0.7800827188055177, 5=1.0, 7=0.4047826737276672, 9=0.6157427682325466, 10=0.907708287439143, 11=1.0, 12=1.0, 13=0.4471731856396224, 14=0.29881924293109347, 15=0.3465184958761106, 18=0.9819443525094192, 19=0.5267590970296238}, {16=0.1, 17=0.1, 1=0.6395876763791866, 3=0.5958454184945285, 20=0.7747514120265678, 6=0.805825507681873, 8=0.42986201080140984, 10=0.7013749610939082, 11=0.2606421264645869, 12=0.7557371160678824, 13=0.3926155606430889}, {16=0.3442382952537705, 1=0.8020200564529931, 18=0.39615326394510764, 4=0.9208361055391983, 5=0.6426921971359085, 6=0.3897334615370779, 7=0.1, 9=0.3546028509605447, 10=0.664949631780541, 12=0.6013546012968357, 14=0.18959957988899678}, {17=0.3681459274617964, 1=0.1, 2=0.1, 5=0.9307914174994552, 6=0.1772957173341993, 10=0.6095993785401036, 14=0.5172777616707022, 15=0.5312520386695241}, {18=0.3084487824733062, 19=0.14684775015790313, 3=0.46571698694660457, 6=0.7424182162565032, 7=0.5210940217274179, 8=0.8009484792199943, 12=0.6325415781369164, 13=0.94060060090369}, {16=0.1, 1=0.7268169683041369, 4=0.56371057370517, 5=0.1, 11=0.7265316199946894, 12=0.1241414313461578} |

Ảnh .: Uncertain dataset khi được sinh ra

|  |
| --- |
| {1=0.4682768261630882, 2=0.711970503893518, 3=0.027687006794852254, 4=0.18916163672875563, 5=0.7346072164304528, 6=0.8315309228859298, 7=0.3494463875284145, 8=0.9071311677362454, 9=0.4645006606311396, 10=0.6437897800945606, 11=0.20610313676844738, 12=0.9153593917857517, 13=0.8032683503066909, 14=0.08458608285236513, 15=0.7223033877154695, 16=0.062305459144174735, 17=0.2072870376400241, 18=0.6192657673261658, 19=0.3524464850356216, 20=0.5714542414448268} |

Ảnh .: Weight table khi được sinh ra

|  |
| --- |
| {1=8.49309043088947, 2=5.413493179434959, 3=5.773963435927853, 4=5.62769930963334, 5=8.573670930388529, 6=9.99315014887735, 7=5.885447029631704, 8=5.245329418650022, 9=5.472946178419839, 10=7.078550201984662, 11=4.94738128113848, 12=8.220576053637789, 13=6.709437775502389, 14=4.5337316799153955, 15=3.684022664575111, 16=6.475337614013704, 17=2.3126515673853834, 18=5.737245107066712, 19=4.146801450893065, 20=4.995513822142339} |

Ảnh .: Mean size-1-itemsets khi được sinh ra

### Kết quả khi thực thi thuật toán

Lựa chọn các tham số minimum support là , threshold là , alpha là .

A computer code with text

Description automatically generated with medium confidence

Ảnh .: Lưạ chọn các tham số cho thuật toán Apriori và thực thi bằng cách gọi phương thức solve()

Kết quả sau khi thực thi ta có được các tập hợp các size-2-itemsets có thỏa điều kiện wPFI:

|  |
| --- |
| [[5, 12], [6, 12], [6, 13], [12, 13], [5, 6]] |

Ảnh .: Kết qủa sau khi thực thi

## Vẽ biểu đồ theo phân tích lý thuyết của phương thức solve() thực thi

Thực hiện việc vẽ biễu đồ thực về độ phức tạp tuân theo hàm sau:

Vẽ biểu đồ thực nghiệm các tập Uncertain Dataset tăng dần kích thước từ 1000 đến 10000.



Ảnh .: Kết quả thực thi khi phân tích theo lý thuyết

Ảnh .: Biểu đồ thời gian chạy khi phân tích theo lý thuyết

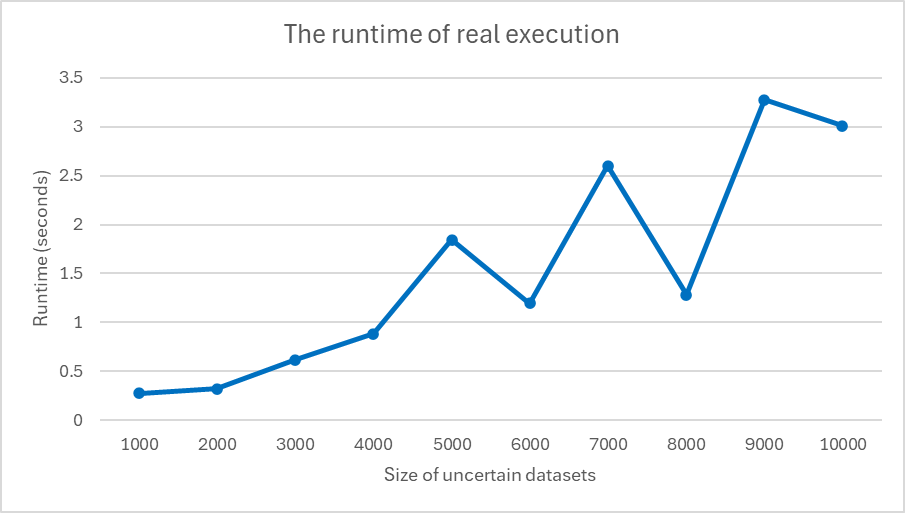
## Vẽ biểu đồ theo thực tế phương thức solve() thực thi

Ở file Plotting.java, sẽ thực hiện việc xuất một fil csv chứa bảng biểu được mô tả dưới đây về kết quả chạy thực tế đối với các tập Uncertain datasets có kích thước từ 1000 đến 10000.

Phần tiếp theo đây là việc thực hiện vẽ biểu đồ thực nghiệm các tập Uncertain Dataset tăng dần kích thước từ 1000 đến 10000, với thời gian thực thi thực tế của thuật toán Apriori đã cài đặt.



Ảnh .: Kết quả thời gian chạy theo thực tế



Ảnh .: Biểu đồ thời gian chạy theo thực tế

# KẾT LUẬN

Sở dĩ có sự khác biệt ở một vài mốc của biểu đồ thời gian chạy theo thực tế so với thời gian chạy theo lý thuyết là do những thành phần khi phân tích ở phần lý thuyết đã bị loại bỏ do không thể xác định xấp xỉ giá trị như kích cỡ của Ia – X chẳng hạn. Ngoài ra các tham số hay trong bài này là thuộc tính của class Apriori giúp loại bỏ các candidate mới sinh ra như alpha, t không thể đo được chúng thật sự hỗ trợ cho việc gia tốc để có thể cho ra kết quả nhanh nhất là bao nhiêu vì uncertain dataset liên tục thay đổi với mỗi lần được sinh ra.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | H.-P. K. M. R. F. V. Thomas Bernecker, "Probabilistic Frequent Itemset Mining in Uncertain," pp. 122-124, 2009. |
| [2] | F. C. J. W. Z. L. a. W. L. Z. Li, "Efficient weighted probabilistic frequent itemset mining in," *Expert Systems,* vol. 38. |