**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**

**MÔN PHÂN TÍCH VÀ THIẾT KẾ GIẢI THUẬT**

**NGHIÊN CỨU CHỦ ĐỀ**

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

*Người hướng dẫn*:  **TS. NGUYỄN CHÍ THIỆN**

*Người thực hiện*:  **TRẦN VĂN PHÁT – 52100919**

**HUỲNH GIA HUY – 52100803**

**HƯỜNG QUANG HUY - 52100893**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ**

**MÔN PHÂN TÍCH VÀ THIẾT KẾ GIẢI THUẬT**

**NGHIÊN CỨU CHỦ ĐỀ**

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

*Người hướng dẫn*:  **TS. NGUYỄN CHÍ THIỆN**

*Người thực hiện*:  **TRẦN VĂN PHÁT– 52100919**

**HUỲNH GIA HUY– 52100803**

**HƯỜNG QUANG HUY - 52100893**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn khoa Công nghệ thông tin đã tạo điều kiện cho chúng em được tiếp cận và hoàn thành bài báo cáo. Chúng em xin chân thành cảm ơn thầy Nguyễn Chí Thiện đã hướng dẫn hoàn thành bài báo cáo.

Trong quá trình làm bài báo cáo, do kiến thức cũng như kinh nghiệm còn nhiều hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp của thầy để chúng em có thể học hỏi được nhiều kĩ năng, kinh nghiệm và sẽ ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm báo cáo của riêng chúng tôi và được sự hướng dẫn của thầy 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 đồ án 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 … tháng … năm 2024*

*Tác giả*

*(ký và ghi rõ họ tên)*

*Trần Văn Phát*

*Huỳnh Gia Huy*

*Hường Quang Huy*

**TÓM TẮT**

Trong bài báo cáo này chúng tôi 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 tôi 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.

**MỤC LỤC**

[**LỜI CẢM ƠN** i](#_Toc155474093)

[**TÓM TẮT** iii](#_Toc155474094)

[Introduction 3](#_Toc155474095)

[Related works 3](#_Toc155474096)

[Preliminaries and Problem Statements 3](#_Toc155474097)

[Định nghĩa 1 - Uncertain dataset 5](#_Toc155474098)

[Định nghĩa 2 - Weight table 5](#_Toc155474099)

[Định nghĩa 3 - Probabilistic frequent itemset 6](#_Toc155474100)

[Định nghĩa 4 - Weighted probabilistic frequent itemset 6](#_Toc155474101)

[Định nghĩa 5 - Itemset weight 6](#_Toc155474102)

[Definition input and output 7](#_Toc155474103)

[Định nghĩa input 7](#_Toc155474104)

[Định nghĩa output 8](#_Toc155474105)

[Methods 9](#_Toc155474106)

[Sơ lược về phần phân tích 9](#_Toc155474107)

[Phân tích các phương thức hỗ trợ cho phương thức solve() 9](#_Toc155474108)

[Phương thức getWeight(Set<T> X) 9](#_Toc155474109)

[Phương thức isSubset(Set<T> X, Map<T, Double> Ti) 10](#_Toc155474110)

[Phương thức probXInTi(Set<T> X, Map<T, Double> Ti) 12](#_Toc155474111)

[Phương thức probXInUD(Set<T> X, int msup, double t, double wX) 13](#_Toc155474112)

[Phương thức isWPFI(double wX, double probXInUD, double t) 16](#_Toc155474113)

[Phương thức itemsInPrevWPFI(Set<Set<T>> WPFI) 17](#_Toc155474114)

[Phương thức genSize1WPFI() 18](#_Toc155474115)

[Phương thức scanFindKItemset() 19](#_Toc155474116)

[Phương thức genWPFIApriori() 20](#_Toc155474117)

[Phương thức chính của thuật toán Apriori, phương thức solve() 24](#_Toc155474118)

[Experiment Setup 26](#_Toc155474119)

[Experiment Results and Discussion 28](#_Toc155474120)

[Kết quả dựa trên các tham số đã chọn 28](#_Toc155474121)

[Phân tích về mặt lý thuyết 29](#_Toc155474122)

[Phân tích về mặt thực tế 30](#_Toc155474123)

[Conclusion 31](#_Toc155474124)

[References 32](#_Toc155474125)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC CHỮ VIẾT TẮT**

wPFI Weighted probabilistic frequent itemset

FI Frequent itemset

FIM Frequent itemset mining

PFI probabilistic frequent itemset

TH Trường hợp

# Introduction

Khám phá các weighted probabilistic frequent itemsets (wPFIs) là một nhiệm vụ cơ bản trong data mining 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 frequent itemset mining (FIM) truyền thống, trong đó mỗi item có cùng weight, khai thác các weighted frequent itemsets xem xét tầm quan trọng hoặc tầm quan trọng của từng item trong từng itemset. Việc gán weight này cho phép chúng tôi khám phá các itemset có ý nghĩa phản ánh sở thích, mức độ ưu tiên hoặc điểm mạnh cơ bản liên quan đến từng item.

Trong bài báo cáo này, chúng tôi cài đặt các thuật toán và kỹ thuật khác nhau đã được phát triển trong bài báo [1] để thực hiện khai phá các WPFIs. Chúng tôi cũng thảo luận về khác nhau giữ độ phức tạp thực tế so với lý thuyết trong việc cài đặt thuật toán khám phá các WPFIs.

# Related works

Như đã đề cập trong bài báo cáo này chúng tôi chủ yếu triển khai cài đặt các thuật toán đã có trong bài báo [1]. Bài báo này đưa ra framework Aprioi dành cho việc tìm kiếm các wPFIs đây là một trong những thuật toán chính được chúng tôi cài đặt. Một trong những thuật toán quan trọng để loại bỏ các itemsets không phải là WPFI cũng được bài báo này đưa ra và được cài đặt theo thuật toán được đưa ra thông qua bài báo [2] .

# Preliminaries and Problem Statements

Vấn đề phát hiện các wPFI phát sinh khi các kỹ thuật FMI truyền thống không còn phù hợp do các weight bằng nhau được gán cho mỗi item. Trong nhiều tình huống thực tế, tầm quan weight của các items trong các itemsets có thể khác nhau và việc xem xét các weight này trở nên quan trọng để phân tích và ra quyết định chính xác.

Thách thức chính nằm ở việc xác định nhanh chóng các wPFIs. Không giống FIM truyền thống, nó chỉ tập trung vào tần suất, việc FI có weight yêu cầu một cách tiếp cận khác kết hợp thông tin weight. Vấn đề này càng trở nên phức tạp hơn khi kích thước dataset tăng lên và số lượng item cũng như weight liên quan của chúng tăng lên.

Một thách thức khác là xác định ngưỡng (threshold) hoặc giá trị hỗ trợ thích hợp (minimum support) để xác định tầm quan trọng của wPFI. Các biện pháp hỗ trợ truyền thống chỉ dựa trên tần suất có thể không phù hợp cho việc khám phá wPFI này vì các biện pháp truyền thống không xem xét đến weight khác nhau của các items. Do đó, việc xác định một biện pháp hỗ trợ hiệu quả khi gán weight cho item là điều cần thiết để khám phá các wPFIs chính xác và có ý nghĩa.

Việc giải quyết những thách thức trên và phát triển các giải pháp hiệu quả để khám phá các wPFI là rất quan trọng để cho phép các tổ chức, doanh nghiệp... 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.

Trong bài báo cáo này chúng tôi xin dẫn lại một số định nghĩa trong phần Preliminaries của bài báo [1] và giải thích, làm rõ thêm các định nghĩa, phần chính trong bài báo cáo của chúng tôi chỉ thực hiện việc cài đặt thuật toán dựa trên những định nghĩa và pseudocode mà bài báo này đã đề cập.

Trước hết hãy sơ lược qua một vài ý đại khái, một Uncertain dataset UD là một tập hợp các transactions UD = . Mỗi transaction sẽ là một trong các tập con của itemset , đây cũng chính là tập hợp toàn bộ các item xuất hiện trong UD. Khác với dataset thông thường, với mỗi trong sẽ có một xác suất đi kèm . Định nghĩa cơ bản về UD sẽ được nêu dưới đây.

Table 1. Bảng mô tả các kí hiệu

|  |  |
| --- | --- |
| **Kí hiệu** | **Mô tả** |
| UD | Một uncertain dataset có kích thước |
|  | Một tập hợp các transation (duy nhất) trong toàn bộ UD |
|  | Một tập hợp các item (duy nhất) trong toàn bộ UD |
|  | Xác suất xuất hiện itemset trong transaction |
|  | Weight table dùng để tra cứu các weight của item có trong UD |
|  | Minimum support dành cho việc tìm kiếm PFI, là một số nguyên trong khoảng |
|  | Ngưỡng xác suất (probabilistic frequent threshold) dành cho việc tìm kiếm PFI, ngưỡng này là một số thực trong khoảng |
|  | Một hệ số quy mô (scale factor) dùng cho việc các tỉa các itemset vừa sinh ra (candidates), scale factor là một số thực trong khoảng |

## Định nghĩa 1 - Uncertain dataset

Một uncertain dataset UD là một hàm có miền xác định cho ra miền giá trị trong khoảng , là tập hợp tất cả các transactions (duy nhất), là tập hợp các items (duy nhất) có trong toàn bộ UD, mỗi giá trị mà hàm cho ra chính là xác suất của item trong transaction .

## Định nghĩa 2 - Weight table

Weight table là một bảng tra cứu weight của các items, có hình dung nó thông qua biểu thị tập hợp , với mỗi là weight của item .

## Định nghĩa 3 - Probabilistic frequent itemset

Cho một uncertain dataset DB, một minimum support , một probabilistic frequent threshold . Một itemset là một PFI khi và chỉ khi .

Ta có thể tính theo công thức sau:

Tuy nhiên, việc đi tìm các tổ hợp của theo điều kiện có thời gian tìm kiếm rất lớn khi dataset ngày càng tăng chính vì thế bài báo [2] đã giới thiệu công thức truy hồi nhằm giảm thời gian tìm kiếm. Việc cài đặt công thức truy hồi theo dynamic programing sẽ được đề cập chi tiết thông qua phần Phương thức probXInUD().

## Định nghĩa 4 - Weighted probabilistic frequent itemset

Cho một uncertain dataset DB, một weight table , một minimum support và một probabilistic frequent threshold , một itemset là một wPFI khi và chỉ khi .

## Định nghĩa 5 - Itemset weight

Cho một itemset , weight của itemset được tính bằng trung bình weight của các items có trong itemset .

# Definition input and output

## Định nghĩa input

Phần này chúng tôi sẽ nói về việc chúng tôi định nghĩa input được bài báo [1] đề cập trong pseudocode wPFI – Apiori algorithm thành code bằng ngôn ngữ lập trình java.

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

Figure 1. Ví dụ input Uncertain dataset với T là Integer



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

Figure 2. Ví dụ về input 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.

Figure 3. Ví dụ về mean của các size-1-itemset với T là Integer



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 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 output

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

Figure 4. Ví dụ về output với T là Integer



# Methods

## Sơ lược về phần phân tích

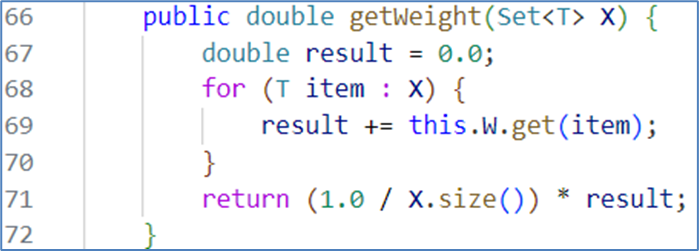
Class Apriori<T> phục vụ chính là tìm ra các itemset là wPFI dựa vào các input được đề cập trong định nghĩa input. Các input này sẽ là argument dành cho việc tạo instance của Apriori. Đối với việc tìm ra các wPFI, chỉ cần gọi phương thức solve() và phương thức solve() này cần nhiều phương thức khác có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

### Phương thức getWeight(Set<T> X)

Figure 5. 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

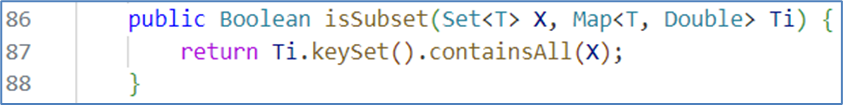
* Worst case:

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)

Figure 6. Phần code phương thức 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)

Figure 7. Phần code phương thức probXInTi()

A screen shot of a computer code

Description automatically generated

* 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)

Figure 8. Phần code phương thức probXInUD()

A screenshot of a computer code

Description automatically generated

* 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

Figure 9. Bảng quy hoạch động mô tả lại công thức truy hồi trên (Nguồn: [2])

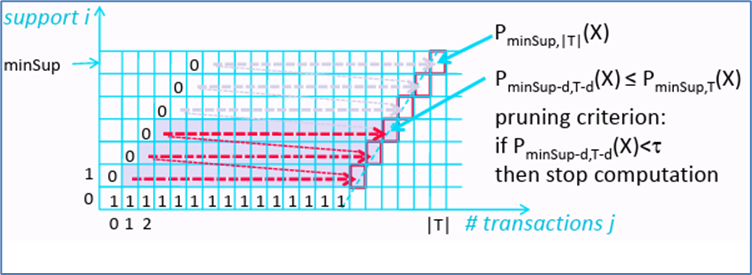
A graph of a function

Description automatically generated

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

Figure 10. Bảng quy hoạch động thể hiện việc tỉa (Nguồn: [2])



* 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:

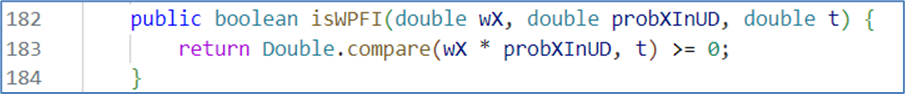
Dòng số 146 và 148.

* 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)

Figure 11. 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

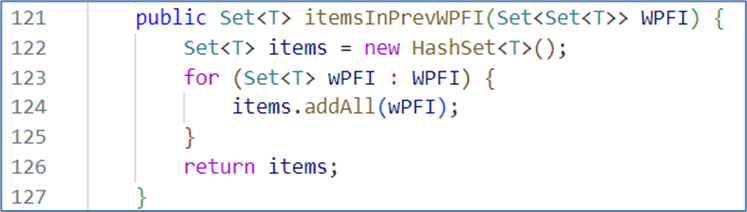
* Worst case:

Không có.

Độ phức tạp của phương thức này là .

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

Figure 12. 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.

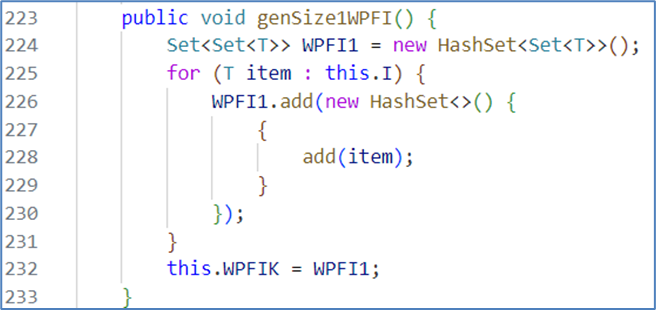
* Worst case:

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()

Figure 13. 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.

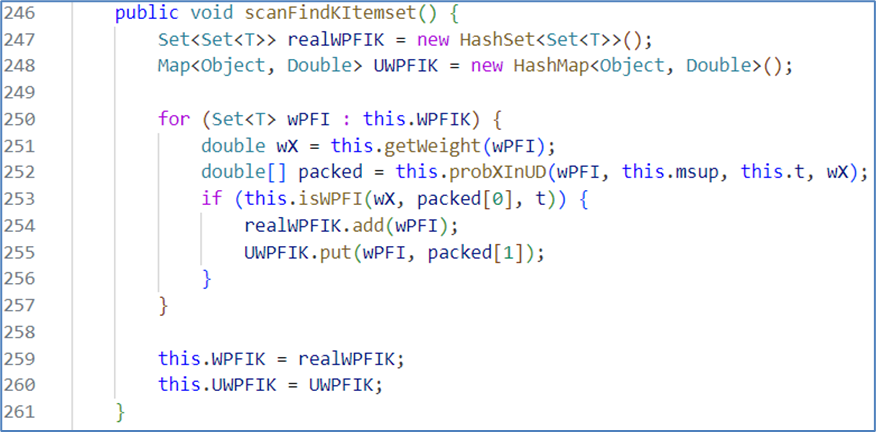
* Worst case:

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()

Figure 14. 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()

Figure 15. Pseudocode cho phương thức genWPFIApriori() (Nguồn: [1])

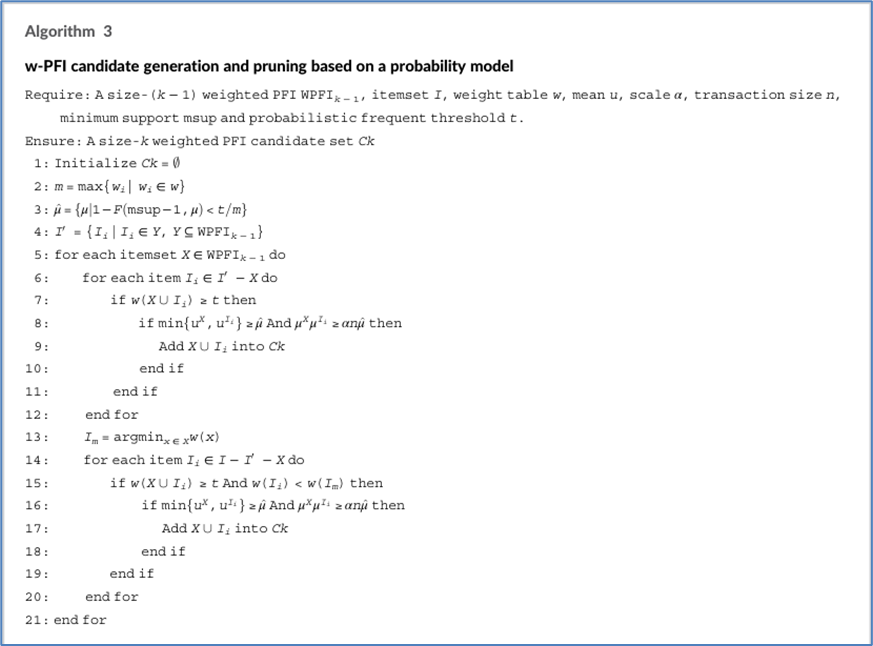


Figure 16. Phần code đầu của phương thức genWPFIApriori()

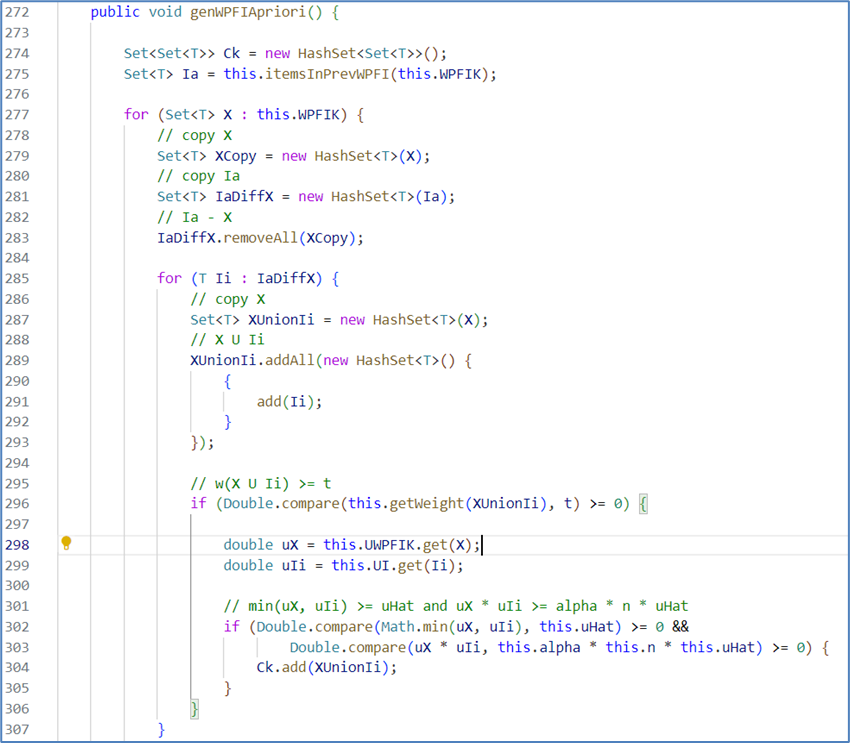


Figure 17. Phần code tiếp theo của phương thức genWPFIApriori()



* 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()

Figure 18. Pseudocode của phương thức solve()

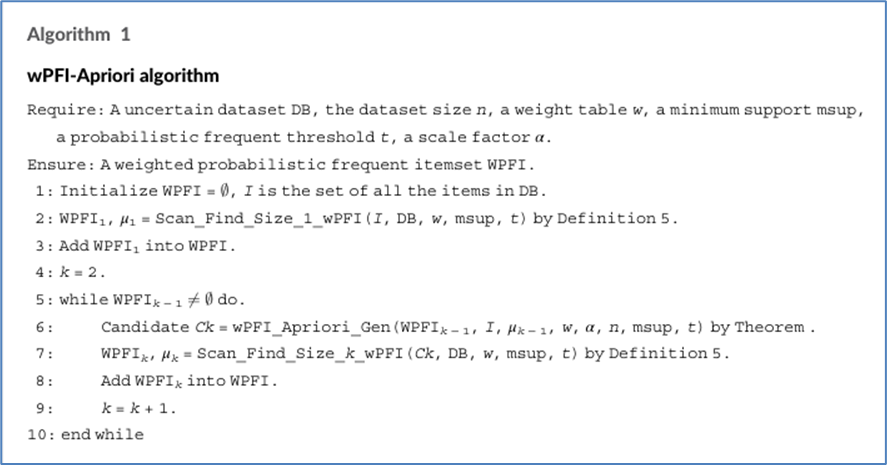


Figure 19. Phần code của phương thức solve()



* 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 Setup

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.

Figure 20. Lệnh sinh ra Uncertain datatset, Weight table, Mean size-1-itemsets

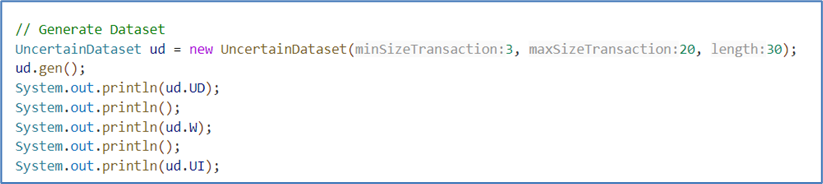


Figure 21. Uncertain dataset khi được sinh ra



Figure 22. Weight table khi được sinh ra



Figure 23. Mean size-1-itemsets khi được sinh ra

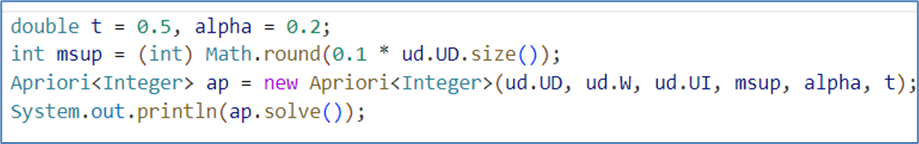


# Experiment Results and Discussion

## Kết quả dựa trên các tham số đã chọn

Lựa chọn các tham số minimum support là 0.1×30=3, threshold là 0.5, alpha là 0.2.

Figure 24. Lựa 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:

Figure 25. kết quả sau khi thực thi



## Phân tích về mặt lý thuyết

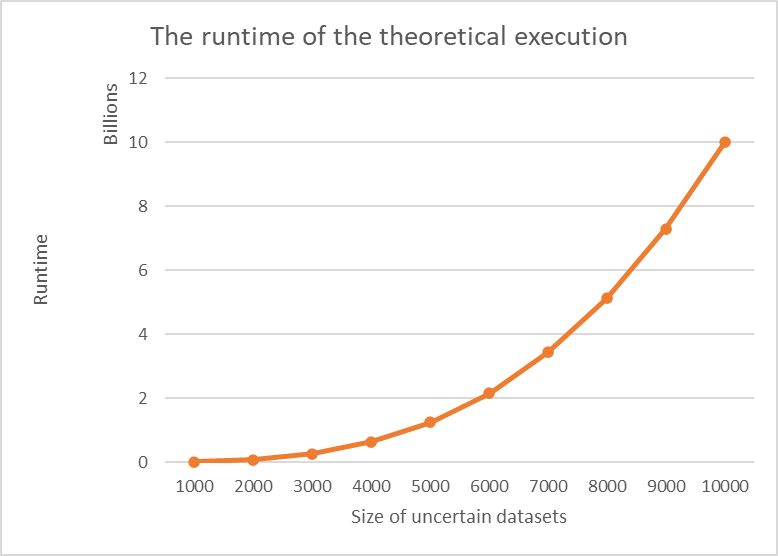
Thực hiện việc vẽ biễu đồ thực về độ phức tạp tuân theo hàm sau:

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

Table 2. Kết quả số thao tác với kích thước dataset tăng dần theo lý thuyết

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size** | **Runtime** | **msup** | **t** | **alpha** |
| 1000 | 10000000 | 10 | 0.5 | 0.6 |
| 2000 | 80000000 | 20 | 0.5 | 0.6 |
| 3000 | 270000000 | 30 | 0.5 | 0.6 |
| 4000 | 640000000 | 40 | 0.5 | 0.6 |
| 5000 | 1250000000 | 50 | 0.5 | 0.6 |
| 6000 | 2160000000 | 60 | 0.5 | 0.6 |
| 7000 | 3430000000 | 70 | 0.5 | 0.6 |
| 8000 | 5120000000 | 80 | 0.5 | 0.6 |
| 9000 | 7290000000 | 90 | 0.5 | 0.6 |
| 10000 | 10000000000 | 100 | 0.5 | 0.6 |

Table 3. Biểu đồ thời gian chạy theo lý thuyết



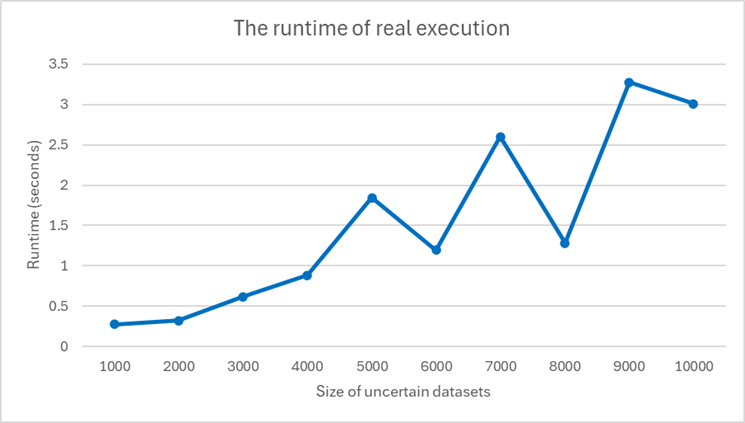
## Phân tích về mặt thực tế

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.

Table 4. Kết quả thời gian chạy với kích thước dataset tăng dần theo thực tế

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size** | **Runtime (seconds)** | **msup** | **t** | **alpha** |
| 1000 | 0.277592 | 10 | 0.5 | 0.6 |
| 2000 | 0.320824 | 20 | 0.5 | 0.6 |
| 3000 | 0.617004 | 30 | 0.5 | 0.6 |
| 4000 | 0.882285 | 40 | 0.5 | 0.6 |
| 5000 | 1.842558 | 50 | 0.5 | 0.6 |
| 6000 | 1.193744 | 60 | 0.5 | 0.6 |
| 7000 | 2.60024 | 70 | 0.5 | 0.6 |
| 8000 | 1.283437 | 80 | 0.5 | 0.6 |
| 9000 | 3.274696 | 90 | 0.5 | 0.6 |
| 10000 | 3.012027 | 100 | 0.5 | 0.6 |

Table 5. Biểu đồ thời gian chạy theo thực tế



# Conclusion

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.

# References

[1] Li Z, Chen F, Wu J, Liu Z, Liu W. Efficient weighted probabilistic frequent itemset mining in uncertain databases. Expert Syst 2021;38. https://doi.org/10.1111/exsy.12551.

[2] Bernecker T, Kriegel H-P, Renz M, Verhein F, Zuefle A. Probabilistic frequent itemset mining in uncertain databases. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA: ACM; 2009, p. 119–28. https://doi.org/10.1145/1557019.1557039.