**VIETNAM GENERNAL CONFEDRATION OF LABOUR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**RESEARCH REPORT 1**

**MINING PROBABILISTIC MAXIMAL HIGH-UTILITY FREQUENT ITEMSETS**

*Advised by*:  **Dr. NGUYỄN CHÍ THIỆN**

*Researcher*:  **TRẦN LÊ GIA BẢO – 520H0516**

**NGUYỄN TRẦN QUANG HUY – 52000668**

**HO CHI MINH CITY, YEAR 2024**

**VIETNAM GENERNAL CONFEDRATION OF LABOUR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**RESEARCH REPORT 1**

**MINING PROBABILISTIC MAXIMAL HIGH-UTILITY FREQUENT ITEMSETS**

*Advised by*:  **Dr. NGUYỄN CHÍ THIỆN**

*Researcher*:  **TRẦN LÊ GIA BẢO – 520H0516**

**NGUYỄN TRẦN QUANG HUY – 52000668**

**HO CHI MINH CITY, YEAR 2024**

**Acknowledgment**

We sincerely thank the Faculty of Information Technology for providing us with the opportunity to access and complete the report. We sincerely thank Mr. Nguyen Chi Thien for guiding us in completing the report.

During the report writing process, due to our limited knowledge and experience, the report unavoidably had some shortcomings. We hope to receive your feedback to learn more skills and experiences to improve ourselves.

Once again, we would like to extend our sincere gratitude!

**DECLARATION OF AUTHORSHIP**

I hereby declare that this thesis was carried out by myself under the guidance and supervision of Mr. Nguyen Chi Thien; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

I will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

Ho Chi Minh City, day 10 month 03 year *2024*

*Author*

(signature and full name)

*Nguyễn. Trần Quang Huy*

*Trần Lê Gia Bảo*

**Abstract**

Báo cáo này…

# Contents

[Acknowledgment i](#_Toc160969436)

[Abstract iii](#_Toc160969437)

[Contents 1](#_Toc160969438)

[List of Figures 3](#_Toc160969439)

[List of Tables 4](#_Toc160969440)

[Introduction 5](#_Toc160969441)

[Reason for choosing the topic 6](#_Toc160969442)

[Objectives of implementing the topic 7](#_Toc160969443)

[Related works 8](#_Toc160969444)

[Methods 10](#_Toc160969445)

[Problem Definition 10](#_Toc160969446)

[Algorithm Design 13](#_Toc160969447)

[Candidate Generation 14](#_Toc160969448)

[PMFIs Conforming 20](#_Toc160969449)

[Experiment Setup 25](#_Toc160969450)

[Experiment Results and Discussion 27](#_Toc160969451)

[Conclusion 29](#_Toc160969452)

[References 30](#_Toc160969453)

**ABBREVIATIONS**

TSP Traveling salesman problem

ACO Ant colony optimization

SOS Symbiotic optimization search

TH Trường hợp

UD Uncertain Database

# List of Figures

[Figure 1 CGEB 14](#_Toc160969376)

[Figure 2 Implement Algorithm 1 15](#_Toc160969377)

[Figure 3 Output after implement Algorithm 1 20](#_Toc160969378)

[Figure 4 APFI-MAX 21](#_Toc160969379)

[Figure 5 Implement Algorithm 2 22](#_Toc160969380)

[Figure 6 FM (Frequent Measurement) 23](#_Toc160969381)

[Figure 7 Implement Algorithm 3 24](#_Toc160969382)

[Figure 8 Output after implement Algorithm 2 and 3 25](#_Toc160969383)

# List of Tables

[Table 1 Example: An Example of Uncertain Database 10](#_Toc160968101)

[Table 2 Example: A Simple Example of PWS Generated From An Example of Uncertain Database 11](#_Toc160968102)

# Introduction

In the evolving landscape of data analysis, the mining of frequent itemsets from large datasets has established itself as a cornerstone for uncovering meaningful patterns and associations. Particularly in sensor networks and similar domains, where data is continuously generated, the ability to efficiently and accurately identify these patterns is crucial for event detection and decision-making processes. However, the inherent uncertainty in sensed data, resulting from factors such as noise, measurement errors, and incomplete information, poses significant challenges to traditional mining techniques.

The notion of uncertainty in data analysis is not new; however, the complexity and volume of data generated by contemporary sensor networks demand novel approaches that can adapt to and manage this uncertainty. Traditional frequent itemset mining algorithms, while robust in deterministic environments, fall short in their ability to account for the probabilistic nature of real-world data. This gap necessitates the development of specialized algorithms capable of navigating the uncertain terrain to mine maximal frequent itemsets (MFIs) effectively.

Addressing this need, our research introduces an approximation algorithm designed to mine probabilistic maximal frequent itemsets (PMFIs) from uncertain sensed data. This approach, termed Approximation of Probabilistic Frequent Itemset Mining (APFI-MAX), leverages Possible World Semantics (PWS) to interpret and process the probabilistic data efficiently. By focusing on approximation methods, APFI-MAX seeks to balance the trade-off between computational feasibility and the accuracy of mined itemsets, offering a scalable solution to the challenges presented by uncertain data.

Our methodology revolves around a two-step process: initially generating PMFI candidates before confirming their status as PMFIs through a rigorous approximation process. This dual-phase approach ensures that the algorithm remains computationally efficient while maintaining a high degree of accuracy in the itemsets identified. By applying this method to various datasets, we demonstrate the effectiveness of APFI-MAX in identifying meaningful patterns within uncertain data, paving the way for enhanced event detection and analysis in sensor networks and beyond.

## Reason for choosing the topic

In our report, we would like to elaborate on our collective decision to delve into the research topic "Approximation of Probabilistic Maximal Frequent Itemset Mining Over Uncertain Sensed Data." [1] Our fascination with this subject stems from the growing integration of sensor networks in diverse fields such as environmental surveillance, health care, and industrial operations. These networks accumulate a wealth of data, which is frequently laced with uncertainties due to various factors like noise, sensor malfunctions, or unpredictable environmental changes.

The core challenge that captured our interest is the effective mining of this uncertain data to unearth significant patterns or events crucial for making well-informed decisions. Conventional data mining methods falter when faced with the vast number of potential interpretations of uncertain data, leading to substantial computational complexities and inefficiencies. Given its challenging nature, the innovative solutions it proposes, and its relevance to real-world issues, this topic has become the unequivocal choice for our current research project.

## Objectives of implementing the topic

The core challenge that captured our interest is the effective mining of this uncertain data to unearth significant patterns or events crucial for making well-informed decisions. Conventional data mining methods falter when faced with the vast number of potential interpretations of uncertain data, leading to substantial computational complexities and inefficiencies. Given its challenging nature, the innovative solutions it proposes, and its relevance to real-world issues, this topic has become the unequivocal choice for our current research project.

The core objective of our research is to architect an algorithm capable of adeptly managing and analyzing uncertain data to unearth valuable probabilistic maximal frequent itemsets (PMFIs). Our approach's innovation hinges on employing approximation techniques designed to slash computational complexity while preserving the accuracy of the mined patterns.

To rigorously assess the efficacy and scalability of our algorithm, we have earmarked the T10I4D100K dataset, encompassing 100,000 transactional records, as our testing ground. This dataset will serve as a critical benchmark to validate our algorithm's capability to process and analyze extensive volumes of uncertain data efficiently

To bring our algorithm to fruition, we have chosen Java as our primary programming language due to its robustness, platform independence, and widespread usage in data-intensive applications. To further enhance our development process, we have decided to incorporate Maven, a powerful project management tool that simplifies the build process in Java projects. Specifically, we'll leverage Maven's dependency management capabilities to utilize Java Eclipse Primitive Collections, which offer memory-efficient collections (such as sets, lists, and maps) optimized for primitive types, providing us with a significant performance boost when handling large datasets.

Upon completing the development and testing phases, we intend to graphically represent our algorithm's performance juxtaposed with the benchmarks reported in existing literature.

# Related works

Initially, it points out the efficacy of well-known algorithms like Apriori and FP-growth in exact databases but underscores their limitations in uncertain databases due to the probabilistic nature of itemset support. To navigate these challenges, the document categorizes existing methods into two primary groups based on their distinct definitions of frequent items in uncertain databases

The first group bases its approach on the expected support of itemsets, utilizing algorithms like U-Apriori and UFP-growth, which are essentially extensions of techniques used in exact databases. However, this approach may overlook critical information about item frequencies due to its reliance on expected support alone.

In response to the limitations of expected support-based methods, a new definition of frequent items, termed probabilistic frequent itemsets (PFIs), is introduced. This definition aims to capture more intrinsic information by summing up the frequencies of itemsets across all possible data interpretations under the Possible World Semantics (PWS) framework. This leads to the development of algorithms like DP and DC, designed to mine PFIs more effectively.

Despite these advancements, the document acknowledges the inefficiency of these methods due to the exponential number of possible worlds they need to consider. It highlights recent efforts focused on mining maximal frequent itemsets, which can represent all frequent itemsets efficiently, thus reducing computational costs and memory requirements.

Finally, The document also addresses the significant computational challenge posed by the need to compute probability mass functions (pmf), which has a time complexity of O(nlogn), making it time-consuming for large datasets

# Methods

## Problem Definition

**Uncertain Transaction Database**:

An uncertain transaction database is a type of uncertain database where data uncertainty is represented in transactions. Uncertain data refers to data collected from sensors or similar sources that have inherent uncertainties due to various factors such as noise, errors in measurement, or incompleteness.

Table 1 Example: An Example of Uncertain Database

|  |  |  |
| --- | --- | --- |
| **ID** | **Attributes** | **Probability** |
| T1 | ABCD | 0.5 |
| T2 | BCD | 0.6 |
| T3 | ABD | 0.7 |

**Possible World Semantics:**

Possible World Semantics is employed to interpret this uncertainty by conceptualizing an uncertain database as a collection of several exact databases, each representing a "possible world." Each possible world is a realization of the uncertain database where each item's presence is determined based on its associated probability.

Table 2 Example: A Simple Example of PWS Generated From An Example of Uncertain Database

|  |  |  |  |
| --- | --- | --- | --- |
| **PWid** | **Transactions** | **Set of Items** | **Probability** |
| PW0 | null | null | (1 - 0.5) \* (1 - 0.6) \* (1 - 0.7) = 0.06 |
| PW1 | T1 | ABCD | 0.5 \* (1 - 0.6) \* (1 - 0.7) = 0.06 |
| PW2 | T1, T2 | ABCD, BCD | 0.5 \* 0.6 \* (1 – 0.7) = 0.09 |
| PW3 | T1, T3 | ABCD, ABD | 0.5 \* (1 - 0.6) \* 0.7 = 0.14 |
| PW4 | T1, T2, T3 | ABCD, BCD, ABD | 0.5 \* 0.6 \* 0.7 = 0.21 |
| PW5 | T2 | BCD | (1 - 0.5) \* 0.6 \* (1 - 0.7) = 0.09 |
| PW6 | T2, T3 | BCD, ABD | (1 - 0.5) \* 0.6 \* 0.7 = 0.21 |
| PW7 | T3 | ABD | (1 - 0.5) \* (1 - 0.6) \* 0.7 = 0.14 |

**Probabilistic Frequent Itemsets**

Probabilistic Frequent Itemsets (PFIs) in the context of uncertain databases are defined by considering the sum of their frequent probabilities across all possible interpretations of the data under Possible World Semantics (PWS).

First, the support X (Sup(X)) of an itemset X is the number of times X occurs in the dataset, the support of X is at least as large or equal as the minimum support threshold (minsup)

An itemset X is deemed a Probabilistic Frequent Itemset if the sum of the probabilities (P) that its support exceeds a given minimum support threshold in these possible worlds is greater than or equal to a minimum probabilistic frequent threshold (minpro)

**Probabilistic Maximal Frequent Item**

A Probabilistic Maximal Frequent Itemset (PMFI) is defined as an itemset that is not only frequent but also maximal in a probabilistic sense within an uncertain database. This means that a PMFI is an itemset that meets the minimum support threshold with a probability greater than or equal to a specified probabilistic threshold, and there is no superset of this itemset that also meets these criteria.

**TODIS-MAX**

TODIS-MAX is its top-down approach for the confirmation of PMFIs candidates, starting from the longest itemsets and progressing to shorter ones. This framework significantly improves over traditional methods by quickly yielding potential PMFIs during the candidate generation phase.

However, one of the primary challenges associated with TODIS-MAX is the computation of the probability mass function (pmf) of an itemset, which is essential for confirming its frequency. This computation has at least a time complexity of O(nlogn), making it less efficient when dealing with large-scale data.

# Algorithm Design

APFI-MAX for mining Probabilistic Maximal Frequent Itemsets (PMFIs) from uncertain data, particularly focusing on noisy sensor data. This method employs an approximation strategy to enhance efficiency. It is structured into two main phases:

1. **Candidate Generation**: The process begins with generating PMFI candidates. It innovatively utilizes a bound on the expectation of support for itemsets to narrow down potential candidates effectively. This step is aimed at reducing the candidate set size significantly, making the subsequent steps more manageable and less time-consuming.
2. **Confirmation of PMFI**s: After narrowing down the candidates, the algorithm confirms which among them are actual PMFIs. This involves estimating the frequency of itemsets, a crucial step where the algorithm approximates the probability mass function (pmf) instead of calculating it exactly, which is typically resource-intensive. This approximation is inspired by the Central Limit Theorem, allowing for a more efficient computation.

## Candidate Generation

**Pseudocode:**

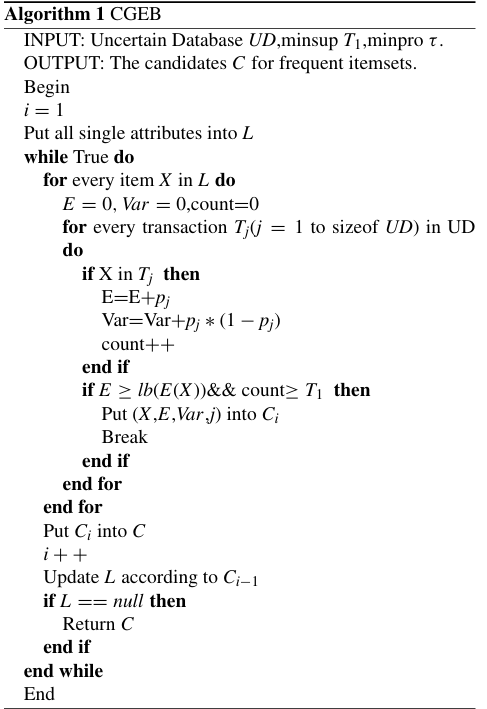
****

Figure 1 CGEB

**Implement code in java:**

****

Figure 2 Implement Algorithm 1

**Explain the Algorithm 1 (CGEB):**

Given Table (Table 2 Example):

* T1: Attributes = ABCD, Pro = 0.5
* T2: Attributes = BCD, Pro = 0.6
* T3: Attributes = ABD, Pro = 0.7

Probabilistic: {A = 0.5 \* 0.7 = 0.35 ; B = 0.5 \* 0.6 \* 0.7 = 0.21 ; C = 0.5 \* 0.6 = 0.3; D = 0.5 \* 0.6 \* 0.7 = 0.21}

1. **Initialization**:

* List L = {A, B, C, D}
* minsup = 1 ; minpro 0.3
* Calculate lowbound = 0.5604628493881512

1. **Calculate Expectation for Single Items**:

* \*A\*: Exp(A) = 0.35 + 0.35 = 0.7
* Count = 2 (Present in T1 and T3).
* Since Count >= minsup(1) and Exp(A) = 0.7 > lb(1, 0.3) ≈ 0.56, A become a candidate itemset.
* \*B\*: Exp(B) = 0.21 + 0.21 + 0.21 = 0.63.
* Count = 3 (Present in T1,T2 and T3).
* Since Count >= minsup(1) and Exp(B) = 0.63 > lb(1, 0.3) ≈ 0.56, B become a candidate itemset.
* \*C\*: Expc(C) = 0.3 + 0.3 = 0.6.
* Count = 2 (Present in T1 and T3).
* Since Count >= minsup(1) and Expc(C) = 0.6 > lb(1, 0.3) ≈ 0.56, C become a candidate itemset.
* \*D\*: Exp(D) = 0.21 + 0.21 + 0.21 = 0.63.
* Count = 3 (Present in T1,T2 and T3).
* Since Count >= minsup(1) and Exp(D) = 0.63 > lb(1, 0.3) ≈ 0.56, D become a candidate itemset.
* **Single Items: A, B, C and D.**

1. **Generate and Calculate Expectation for Pair Itemsets**:

* Combine items from C1 to form pairs: {AB, AC, AD, BC, BD, CD}.
* \*AB\*: Exp(AB) = Exp(A) \* Exp(B) = 0.7 \* 0.63 = 0.441.
* Count = 2 (Present in T1 and T3).
* Since Count >= minsup(1) and Exp(AB) = 0.441 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*AC\*: Exp(AC) = Exp(A) \* Exp(C) = 0.7 \* 0.6 = 0.42.
* Count = 1 (Present only in T1).
* Since Count >= minsup (1) and Exp(AC) = 0.42 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*AD\*: Exp(AD) = Exp(A) \* Exp(D) = 0.7 \* 0.63 = 0.441.
* Count = 2 (Present in T1 and T3).
* Since Count >= minsup(1) and Exp(AD) = 0.441 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*BC\*: Exp(BC) = Exp(B) \* Exp(C) = 0.63 \* 0.6 = 0.378.
* Count = 2 (Present in T1 and T2).
* Since Count >= minsup(1) and Exp(BC) = 0.378 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*BD\*: Exp(BD) = Exp(B) \* Exp(D) = 0.63 \* 0.63 = 0.3969.
* Count = 3 (Present in all transactions).
* Since Count >= minsup(1) and Exp(BD) = 0.3969 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*CD\*: Exp(CD) = Exp(C) \* Exp(D) = 0.6 \* 0.63 = 0.378.
* Count = 2 (Present in T1 and T2).
* Since Count >= minsup(1) and Exp(BD) = 0.378 is NOT greater than lb(1, 0.3) ≈ 0.56.
* **Pair Itemsets: ∅.**

1. **Generate and Calculate Expectation for Triplet Itemsets**:

* Combine items from C1 to form triplets: {ABC, ABD, ACD, BCD}.
* \*ABC\*: Exp(ABC) = Exp(A) \* Exp(B) \* Exp(C) = 0.7 \* 0.63 \* 0.6 = 0.265.
* Count = 1 (Present only in T1).
* Since Count >= minsup(1) and Exp(ABC) = 0. 265 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*ABD\*: Exp(ABD) = Exp(A) \* Exp(B) \* Exp(D) = 0.7 \* 0.63 \* 0.63 = 0.278.
* Count = 2 (Present in T1 and T3).
* Since Count >= minsup (1) and Exp(ABD) = 0. 278 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*ACD\*: Exp(ACD) = Exp(A) \* Exp(C) \* Exp(D) = 0.7 \* 0.6 \* 0.63 = 0. 265.
* Count = 1 (Present only in T1).
* Since Count >= minsup(1) and Exp(ACD) = 0. 265 is NOT greater than lb(1, 0.3) ≈ 0.56.
* \*BCD\*: Exp(BCD) = Exp(B) \* Exp(C) \* Exp(D) = 0.63 \* 0.6 \* 0.63 = 0.238.
* Count = 2 (Present in T1 and T2).
* Since Count >= minsup(1) and Exp(BCD) = 0.238 is NOT greater than lb(1, 0.3) ≈ 0.56.
* **Triplet Itemsets: ∅.**

1. **Generate and Calculate Expectation for Quadruplet Itemsets**:

* Since we have all four items A, B, C, and D present in T1, we can form the quadruplet {ABCD}.
* \*ABCD\*: Exp(ABCD) = Exp(A) \* Exp(B) \* Exp(C) \* Exp(D) = 0.7 \* 0.63 \* 0.6 \* 0.63 = 0.167.
* Count = 1 (Present only in T1).
* Since Count >= minsup(1) and Exp(BCD) = 0.167 is NOT greater than lb(1, 0.3) ≈ 0.56.
* **Quadruplet Itemsets: ∅.**

1. **Output**:

* The candidate itemsets from this process are {A, B, C, D}.

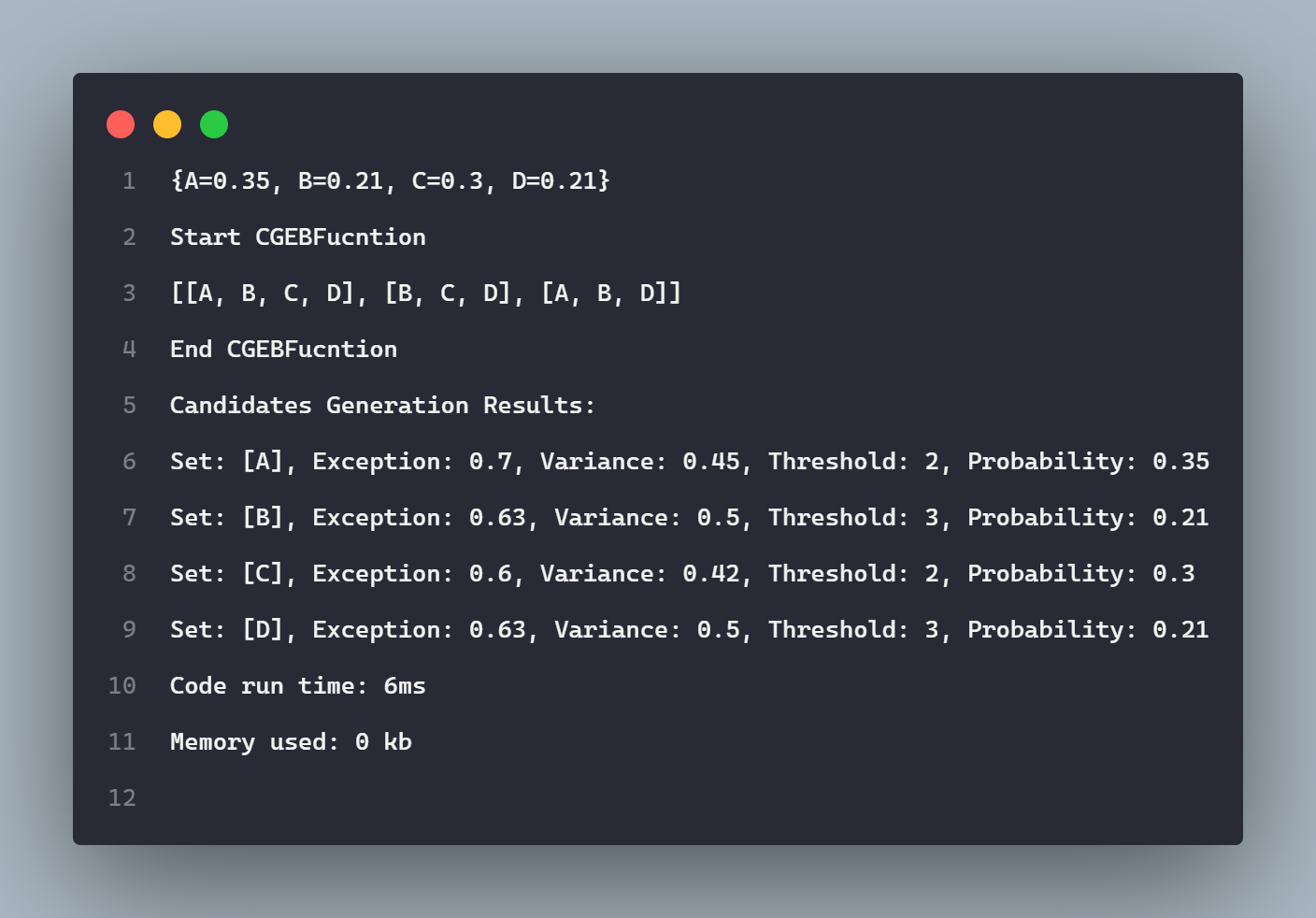


Figure 3 Output after implement Algorithm 1

## PMFIs Conforming

In this section there will be two algorithms (APFI-MAX and FM):

**Pseudocode**:

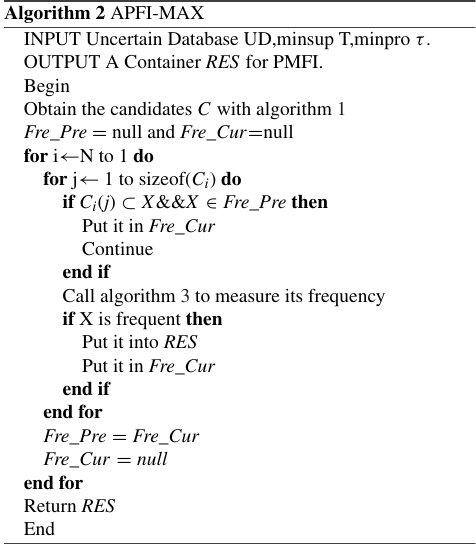


Figure 4 APFI-MAX

**Implement code in java:**

****

Figure 5 Implement Algorithm 2

**Explain Algorithm 2 (APFI MAX)**

1. **Initialization**:
   * **Fre\_Pre** is initially set to null because there are no previously found frequent itemsets.
   * **Fre\_Cur** is also set to null at the start.
2. **Processing Candidates**:
   * Since **Fre\_Pre** is null initially, all candidates from {A, B, C, D} can be added directly to **Fre\_Cur**.
3. **Frequency Measurement**:
   * For each itemset in **Fre\_Cur** (which now contains D, C, B, A), call Algorithm 3 to measure its frequency.

Algorithm 3 will check if the expected support E(X) for each itemset X is greater than or equal to the upper bound (ub) calculated.

**Pseudocode**:

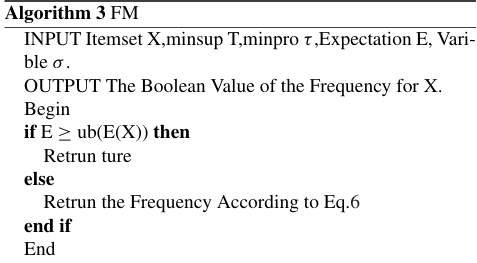


Figure 6 FM (Frequent Measurement)

**Implement code in java:**

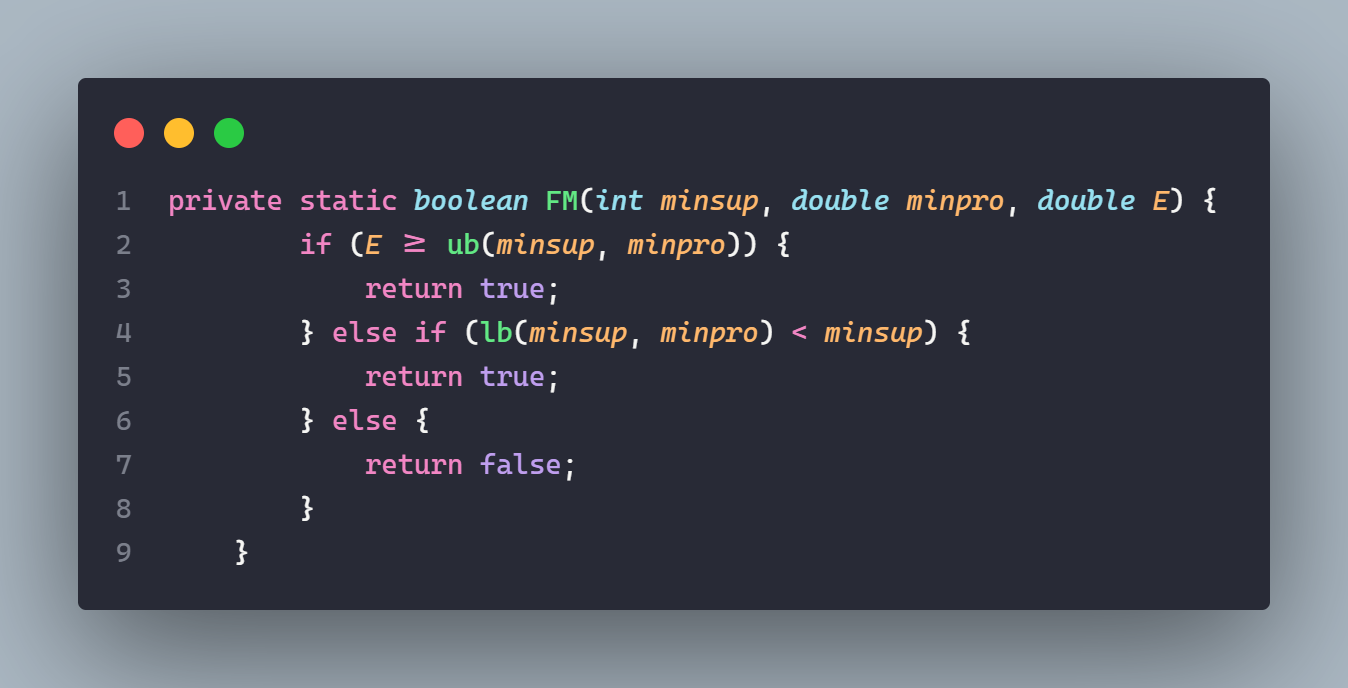


Figure 7 Implement Algorithm 3

**Explain Algorithm 3 (FM- Frequent Measurement)**

* **Input**: Each itemset X, **minsup = 1**, **minpro = 0.3**, and the calculated upper bound of 1.94.
* **Output**: The boolean value of the frequency for X.

For each itemset X (B, D, BD):

1. Calculate E(X) based on their occurrence probabilities from the transactions (as per the details from Algorithm 1).
2. Compare E(X) to the calculated ub (1.94):
   * If E(X) ≥ ub, then the itemset X is considered frequent. No candidates generation satisfied.
   * Otherwise, refer to Eq.6 (lb < minsup) to determine the frequency of X. So It is true.

**Output:**

* PMFIs Conforming: D, C, B, A.
* Running code:



Figure 8 Output after implement Algorithm 2 and 3

# Experiment Setup

**Programming language**: Java

**Method:**

* 1. Java Object
  2. Using Maven (Java Eclipse primitive collections)

**For the implementation using Java Eclipse primitive collections**:

In the first implementation method, we used object data types (Integer, Double, Boolean, etc.). Using these data types would require boxing and unboxing operations (boxing is the process of converting a primitive type (int, double, boolean) to its corresponding object wrapper class (Integer, Double, Boolean), and unboxing is the reverse process). This leads to additional runtime costs due to object creation and destruction during each operation. While the overhead may be negligible for small amounts of data, it can become substantial for large data sets that are frequently updated. Therefore, by using primitive types where possible to avoid boxing and unboxing, the application runtime performance can be significantly improved.

Primitive types allow operations to be performed directly on the underlying values without implicit object instantiation during boxing/unboxing. For throughput-sensitive systems handling vast amounts of numeric data, primitive collections can provide a notable performance boost compared to their object-oriented equivalents. This optimization is especially important for applications that perform frequent insertions, deletions or updates on large datasets.

**Datasets**: T10I4D100K with 100,000 transactions in this file

**Computer configuration used for experiments**:

* + CPU: Intel Core i5-5200U
  + Memory: 8 GB RAM
  + Operating system: Windows 10 Home
  + Storage: 512 GB SSD

# Experiment Results and Discussion

1. The first graph shows data for "APFT-MAX" as a single series.
2. The second graph also represents "APFT-MAX" but splits the data into two series: "Java Object" and "Java Eclipse Primitive
3. In the first graph, "APFT-MAX" running times increase significantly with the size of the data, reaching up to 25,000 ms for the 100k size.
4. In the second graph, both "Java Object" and "Java Eclipse Primitive" show an increase in running time with data size, but the scale is different, with a maximum running time of about 17,000 ms for the "Java Object" at 100k size. While “Java Eclipse Primitive” only takes about 10,000 ms at 100k size

* "Java Eclipse Primitive" is more memory-efficient as it has lower memory usage compared to "Java Object."

# Conclusion

The paper presents a method called APFI-MAX for mining frequent itemsets from uncertain databases to ultimately minimize computational complexity. Specifically, APFI-MAX utilizes a lower bound constraint on the expected support of candidate PFMIs to reduce the candidate set generation.

Then, an estimation method is proposed to measure the frequencies of candidates instead of accurately computing the probability mass function, thus reducing the computational complexity from O(nlogn) to O(n).

In summary, the paper proposes a method for mining frequent itemsets that is optimized for time complexity and suitable for large databases. This method promises applications in many areas such as environmental monitoring, healthcare, e-commerce, etc. where efficient analysis of large uncertain datasets is needed. By reducing the candidate generation and support estimation steps, APFI-MAX can extract frequent patterns from massive uncertain data in a scalable manner. The lower bound pruning and frequency estimation techniques work together to improve runtime performance for uncertain frequent pattern mining.

# References

|  |  |
| --- | --- |
| [1] | L. N. X. T. Z. L. a. L. Z. S. Chen, "Approximation of Probabilistic Maximal Frequent Itemset Mining Over Uncertain Sensed Data," IEEE Access, 2020. |