VIETNAM GENERAL CONFEDERATION OF LABOUR
**TON DUC THANG UNIVERSITY**
**FACULTY OF INFORMATION TECHNOLOGY**

**REPORT OF THESIS RESEARCH 1**

# PROBABILISTIC WEIGHTED FREQUENT ITEMSET MINING OVER UNCERTAIN DATA STREAMS

*Advised by*: **Dr Nguyễn Chí Thiện**

*Authors*: **Nguyễn Đình Quý – 520H0675**

**Nguyễn Nhất Thống – 52000808**

**HO CHI MINH CITY, 2024**

VIETNAM GENERAL CONFEDERATION OF LABOUR
**TON DUC THANG UNIVERSITY**
**FACULTY OF INFORMATION TECHNOLOGY**

**REPORT OF THESIS RESEARCH 1**

# PROBABILISTIC WEIGHTED FREQUENT ITEMSET MINING OVER UNCERTAIN DATA STREAMS

*Advised by*: **Dr Nguyễn Chí Thiện**

*Authors*: **Nguyễn Đình Quý – 520H0675**

**Nguyễn Nhất Thống – 52000808**

**HO CHI MINH CITY, 2024**

# ACKNOWLEDGMENT

We sincerely thank the Faculty of Information Technology for providing us with the opportunity to access and complete the report. We would like to express our heartfelt gratitude to Dr Nguyen Chi Thien for guiding us in completing the report.

During the process of preparing the report, due to limited knowledge and experience, there may be some shortcomings. We greatly appreciate any feedback from you so that we can learn more skills and experiences, and improve further.

We sincerely thank you!

*Ho Chi Minh City, day month year 2024*

*Author*

*(Signature and full name)*

# THE REPORT WAS COMPLETED
# AT TON DUC THANG UNIVERSITY

I hereby declare that this is our own research work, conducted under the scientific supervision of Dr. Nguyen Chi Thien. The research contents and results in this topic are truthful and have not been previously published in any form. The data presented in tables and figures, serving for analysis, comments, and evaluations, are collected by the author from various sources, clearly referenced in the reference section.

Furthermore, the project also incorporates some comments, evaluations, as well as data from other authors, and different organizations, all of which are appropriately cited and referenced.

**If any misconduct is discovered, I take full responsibility for the content of my project.** Ton Duc Thang University is not liable for any copyright infringements or violations caused by me during the implementation process (if any).

*Ho Chi Minh City, day month year 2024*

*Author*

*(Signature and full name)*

# ABSTRACT


Mining frequent itemsets from uncertain data streams is a critical task in data analysis, yet it poses unique challenges due to the inherent uncertainty and dynamic nature of streaming data. This report presents a novel approach, focusing on probabilistic weighted frequent itemset mining over uncertain data streams.

Our method, PFIT, leverages an efficient in-memory index to manage data synopsis, facilitating real-time output of probabilistic frequent itemsets within sliding windows. We introduce PFIMoS, a dynamic depth-first algorithm, to construct and update PFIT, optimizing runtime and memory usage by estimating probabilistic support ranges. Additionally, we tackle the computational overhead associated with low minimum support thresholds and dense data through PFIMoS+, an error-parameter-guided heuristic algorithm.

Our methods are developed based on the algorithmic proposals found in the study by *Li et al. (2018)[8]*. Furthermore, to enhance flexibility in usage, improve the quality of outcomes, and increase efficiency in processing large datasets, we have incorporated a probability weighting approach based on the research by *Li et al. (2020)[9]*. This integration aims to refine our methodology by considering the importance and uncertainty associated with each data item, thereby enriching our analysis and enabling more nuanced interpretations of complex data environments.

Through empirical evaluation, we demonstrate the effectiveness and efficiency of our approach, offering insights into scalable probabilistic frequent itemset mining in uncertain data stream environments.

# CONTENTS

# LIST OF FIGURES, DIAGRAM

# LIST OF TABLE

# ABBREVIATIONS

| | |
|---|---|
| FEMP | Fast and Exact Mining of Probabilistic |
| FP-tree | Frequent Pattern tree |
| H-Struct | Hyper-linked data structure |
| PFIMoS | Probabilistic Frequent Itemset Mining over Streams |
| PFIMoS+ | Probabilistic Frequent Itemset Mining over Streams Plus |
| PFIMUDS | Probabilistic Frequent Itemset Mining over Uncertain Data Streams |
| PFIT | Probabilistic Frequent Item-set Tree |
| ProApriori | Probabilistic Apriori |
| ProFPGrowth | Probabilistic Frequent Pattern Growth |
| ProFP-tree | Probabilistic Frequent Pattern tree |
| PWFIMoS | Probabilistic Weighted Frequent Itemset Mining over Streams |
| PWFIMoS+ | Probabilistic Weighted Frequent Itemset Mining over Streams Plus |
| PWFIT | Probabilistic Weighted Frequent Item-set Tree |
| UML | Unified Modeling Language |
| w-PFIs | Weighted Probabilistic Frequent Itemsets |

# SUMMARY OF MATHEMATICAL NOTATIONS

| | |
|---|---|
| $\lambda$ | Minimum support |
| $\tau$ | Minimum probability |
| $\mathbf{x}$ | Itemset |
| X, x | Uncertain item, item |
| $\mathbf{X}$ | Uncertain itemset |
| $\mathbf{T}$ | Uncertain Transaction |
| $\mathbf{D}$ | Uncertain Database |
| $\eta$ | Sliding window |
| pr(x) | Occurrence probability of item |
| $\mathrm{Sup}(\mathbf{x})$ | Support of itemset |
| $\mathrm{Exp}(\mathbf{x})$ | Expected Support of itemset |
| $\mathrm{wt}(\mathbf{x})$ | Weight of itemset |
| $\mathrm{Pr}(\mathbf{x})$ | Probability support of itemset |
| $\mathrm{Pr}_{\mathrm{Sup}(\mathbf{x})\geq i} > \tau$ | Probability support in itemset but greater than minimum probability |
| $\mathrm{lb}(\mathrm{Pr}(\mathbf{x}))$ | Lower bound of probability support |
| $\mathrm{ub}(\mathrm{Pr}(\mathbf{x}))$ | Upper bound of probability support |
| $\mathrm{wP}(\mathbf{x})$ | Probability support with weight |

# 1. Introduction

In the field of data mining, uncovering frequent itemsets within streams of data is crucial for extracting meaningful insights. The exploration of probabilistic frequent itemsets in uncertain data streams, as presented by ***Li et al. (2018)[8]*** in "Probabilistic frequent itemset mining over uncertain data streams" in Expert Systems With Applications, marks a significant advancement in this domain. Their work introduces the PFIMoS and PFIMoS+ algorithms, which efficiently mine such itemsets by employing an innovative in-memory index, PFIT, and by estimating the range of probabilistic support to reduce computational demands.

Building upon the foundation laid by ***Li et al. (2018)[8]***, this project seeks to extend the existing methodology by introducing weights to the frequent probability itemsets. This modification aims to enhance the granularity of the analysis by accounting for the varying significance of item occurrences within itemsets, thereby offering a more refined understanding of data stream patterns. The weighted approach promises to improve the algorithms' applicability across different contexts, enabling a prioritized analysis of itemsets based on their relevance or importance to specific applications.

This report aims to detail the theoretical underpinnings of this modification, its implementation, and the potential impacts it holds for the field of data mining. Through a meticulous examination of the proposed approach and its comparison with the foundational work of ***Li et al. (2018)[8]***, this report endeavors to highlight the innovation and added value brought about by incorporating weights into the mining of probabilistic frequent itemsets.m mining research.

## 2. Related work

To explore deeply into this topic. Two new definitions must be clearly. Firstly, ***Expected Frequent Item-set [5]***. For expected frequent itemset,this definition focuses on calculating the expected support of itemsets. The expected support can be computed with O(n) time complexity and O(1) space complexity. Several algorithms have been devised to discover expected frequent itemsets, mainly based on a priori rules and traditional data structures like ***FP-tree [6][7]*** and ***H-struct [1]***.

However, expected support cannot show the total of probabilistic characteristic of data. Therefore, ***Probabilistic Frequent item-set [3]*** was invented.Mining probabilistic frequent itemsets can better discover probabilistic features. Algorithms like ***ProApriori [3]*** and ***ProFPGrowth [4]*** have been proposed to discover probabilistic frequent itemsets, utilizing a priori rules and the ***ProFP-tree [10]*** data structure to maintain itemsets.

In the realm of uncertain data mining over data streams, algorithms such as ***FEMP [2]*** and ***PFIMUDS [10]*** have been proposed to discover probabilistic frequent itemsets in data streams. These methods focus on computing probabilistic support and employ techniques like a priori rules to reduce computational costs.

Further advancing this domain, **Li et al. (2018)[8]** developed **PFIMoS** and **PFIMoS+** algorithms to efficiently mine probabilistic frequent itemsets in uncertain data streams, using an in-memory index (**PFIT**) and probabilistic support ranges to significantly cut down on computational time and memory use, surpassing previous methods like **TODIS-Stream [11]** and **FEMP [2]**. This breakthrough enhances uncertain data stream mining for practical use. In a subsequent study by **Li, Chen, Wu, Liu, and Liu (2020) [9]**, a new algorithm for mining weighted probabilistic frequent itemsets **(w-PFIs)** in uncertain databases was introduced, employing a novel probability model and three pruning techniques to efficiently reduce the search space. This research advances weighted uncertain data mining by combining weights and probabilistic assessments, providing a thorough analysis of itemset mining's significance and uncertainty.

# 3. Preliminaries and Problem Statements

## 3.1 Preliminaries

- **Definition 1**: Distinct Item (i)
  - **Concept:** The list of different items in uncertain database
  - **Formula:** Where n is the number of distinct items. Available on all data

$$x = \{x_1, x_2, \ldots, x_n\} \tag{3.1}$$

  - **Explanation and Example:** This is 5 different items in uncertain database

$$x = \{A, B, C, D, E\}$$

- **Definition 2:** Uncertain Item
  - **Concept:** It is a random variable. Corresponding to each item, there will be a different occurrence probability
  - **Formula:**

$$X = \{x_1, p_1\} \tag{3.2}$$

  - **Explanation and Example:** This presents the item 'A' and the occurrence probability '0.3'

$$X = \{A, 0.3\}$$

- **Definition 3:** Uncertain itemset
  - **Concept:** It is Mutitional random variables
  - **Formula:** Where n is the number of random variables.

$$\mathbf{X} = \{(x_1, p_1), (x_2, p_2), \ldots, (x_n, p_n)\} \tag{3.3}$$

  - **Explanation and Example:** This example shows the Uncertain itemset with 3 pairs of uncertain items and the probabilities are (A, 0.3), (B, 0.6), (D, 0.8)

$$\mathbf{X} = \{(A, 0.3), (B, 0.6), (D, 0.8)\}$$

- **Definition 4:** Uncertain Transaction
  - **Concept:** Uncertain Transaction is a list of multidimensional random variables. Corresponding to a multi-dimensional random variable, there will be a symbolic ID.
  - **Formula:** Where n is the number of uncertain transactions

$$\mathbf{T} = \{\mathrm{ID}, \{(x_1, p_1), (x_2, p_2), \ldots, (x_n, p_n)\}\} \tag{3.4}$$

  - **Explanation and Example:** This example shows 1 Uncertain transaction consists of a list of multidimensional random variables including (A, 0.3), (B, 0.6), (D, 0.8) and symbolic ID

$$\mathbf{T} = \{\mathrm{ID}, \{(A, 0.3), (B, 0.6), (D, 0.8)\}\}$$

- **Definition 5:** Uncertain Database
  - **Concept:** Uncertain Database is a collection of uncertain transactions.
  - **Formula:** where n is the number of uncertain transactions

$$\mathbf{D} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T_n}\} \tag{3.5}$$

  - **Explanation and Example:** This example shows an uncertain database as a table where each of its rows is an uncertain transaction

| | |
|---|---|
| ID1 | {(A,0.3), (B, 0.6), (D, 0.8)} |
| ID2 | {(C, 0.4), (E, 0.7)} |
| ID3 | {(B, 0.9)} |
| ID4 | {(D, 0.6), (E, 0.7)} |
| ID5 | {(A,0.5), (B, 0.4),(C, 0.6),(D, 0.8), (E, 0.8)} |

Table 3.1 Example of a Uncertain Database

- **Definition 6**: Support
  - **Concept:** Support is the frequency of an itemset appearing in the entire uncertain database.
  - **Formula:**

$$\mathrm{Sup}(\mathbf{x}) = \sum\nolimits_{t \epsilon T} 1_{\{i \epsilon t\}} \tag{3.6}$$

- **Explanation and Example:** It compute the sum of items occur in uncertain database. Similarly, according to ***Table 3.1*,** itemset A occurs in the first Uncertain transaction 1 time so increase to 1, Otherwise, in Uncertain transaction 2, 3, 4 don't have so increase to 0. Finally, in the last uncertain transaction also own 'A' so we continue rising to 1. At the end of computing support, collecting support is 2

- **Definition 7:** Expected Frequent Itemset:
  - **Concept:** An itemset is considered a frequent itemset when expected support > minimum support. Below is the formula to calculate expected support. (Expected support is a form of support but is often used in uncertain data streams)
  - **Formula:**

$$\mathrm{Exp}(\mathbf{x}) = \sum_{i\epsilon \mathrm{UT}} \mathrm{Sup}(\mathbf{x})\mathrm{p(i)} \tag{3.7}$$

  - **Explanation and Example**: Calculate expected frequent itemset based on the sum of each item's probability on each transaction

- **Definition 8:** Probabilistic Frequent Item-set
  - **Concept:** An itemset is considered frequent when probabilistic support > minimum support. Below is the calculation formula
  - **Formula:**

$$\Pr(\mathbf{x}) = Max\{i \middle| P_{\mathrm{Sup}(\mathbf{x})\geq i} > \tau\} \tag{3.8}$$

Where:

$$\Pr_{\mathrm{Sup}(\mathbf{x})\geq \mathrm{i}}(\mathbf{x}) = \sum_{\mathrm{Sup}(\mathbf{x})\geq \mathrm{i}} pr(\mathrm{i}) \tag{3.9}$$

  - **Explanation and Example:** For itemset 'A' we see that with minimum probability is 0.2 so first transaction and five transaction have probability greater than minimum probability so we have probability support equals to 2.

- **Definition 9**: Lower bound
  - **Concept**: The smallest value that a random variable, a set of numbers, or a quantity can take or not exceed towards the bottom. In the context of an

algorithm or a probability distribution, the lower bound has can indicate the minimum value below which the probability of finding a value of the variable is very low or non-existent.

- **Formula**:

$$\mathrm{lb}(\Pr(\mathbf{x})) = Max(\mathrm{lb}'(\Pr(\mathbf{x})), 0) \tag{3.10}$$

Where:
$$\mathrm{lb}'(\Pr(\mathbf{x})) = \mathrm{Exp}(\mathbf{x}) - \sqrt{-2\mathrm{Exp}(\mathbf{x})ln(1-\tau)} \tag{3.11}$$

- **Explanation and Example**: Base on Expected support and minimum probability we compute the lower bound and check the condition if it smaller than 0 we mark it 0

- **Definition 10**: Upper bound
  - **Concept**: The maximum value that a random variable, a set of numbers, or a quantity can reach or not exceed. In an algorithm or model, an upper limit can be used to indicates the value above which the probability of finding the value of the variable is very low or non-existent.
  - **Formula**:

$$\mathrm{ub}(\Pr(\mathbf{x})) = Min(\mathrm{ub}'(\Pr(\mathbf{x})), \mathrm{Sup}(\mathbf{x})) \tag{3.12}$$

Where:
$$\mathrm{ub}'(\Pr(\mathbf{x})) = \frac{2\mathrm{Exp}(\mathbf{x}) - ln\tau + \sqrt{ln^2\tau - 8\mathrm{Exp}(\mathbf{x})ln\tau}}{2} \tag{3.13}$$

  - **Explanation and Example**: Base on Expected support and Support we compute the lower bound and check the condition if it smaller than 0 we mark it support

## 3.2 Problem Statements

- The paper did not have weight calculations
- The code from the paper applies recursion-based calculation in each function but it is not efficient compared to large data sets
- With data, the paper applied a gaussian distribution to each item on each transaction and caused an overflow situation.

# 4. Solutions

## 4.1 Revisions

**Weighted:** Because of the revision of this report is that adding weights to each item to enhance the granularity of the analysis by accounting for the varying importance of item occurrences in item sets, the report There will be updates on how to calculate weight on each item to test its effectiveness.

**DELTRANS function**: Our approach in the DELTRANS function is a little different from the paper. Specifically, we combine the weights assigned to each probabilistic support, ensuring more accurate analysis results. Furthermore, we optimized our method to avoid recursion, which is especially important for processing large data sets. Instead, we leverage the strategy of introducing new nodes and the post-tree structure of child nodes, thereby streamlining list reuse and improving execution efficiency.

**Processing data**: To address the lack of item probabilities and weights in the dataset, we used a **Gaussian distribution** to aggregate the probabilities, applying a **sigmoid function** to prevent overflow, following ***Li et al. (2018)[8]*** and assign random weights between 0 and 1 according to ***Li et al. (2020)[9]***.

## 4.2 The weight of probability support

- **Definition 11:** Probability Support weight
  - **Concept**: Is a variation of probability support. But we will calculate more with weighted to block the conditions in ADDTRANS and DELTRANS functions
  - **Formula**:

$$\mathrm{wPr} = \mathrm{w}(\mathbf{x}) Max\{i \,|\, Pr_{\mathrm{Sup}(\mathbf{x})\geq i} > \tau\} \tag{4.1}$$

Where:

$$\mathrm{Pr}_{\mathrm{Sup}(\mathbf{x})\geq \mathrm{i}}(\mathbf{x}) = \sum_{i \epsilon \mathrm{UT}, \mathrm{Sup}(x)\geq i} pr(\mathrm{i}) \tag{4.2}$$

  - **Explanation and Example**:Similar to probability support but we will multiply each Probability support with weight of itemset

- **Definition 12:** Weighted
  - **Concept:** Is in the number of each itemset
  - **Formula:**

$$w(\mathbf{x}) = \frac{1}{|\mathbf{x}|}\sum_{i \epsilon x} w(i) \tag{4.3}$$

  - **Explanation and Example**: We will compute the weight in each itemset base on the average weight of each item.

| A | B | C | D | E |
|---|---|---|---|---|
| 0.2 | 0.4 | 0.1 | 0.9 | 0.7 |

Table 4.1 Table describe the result of calculating weight of each item

## 4.3 UML Class Diagram

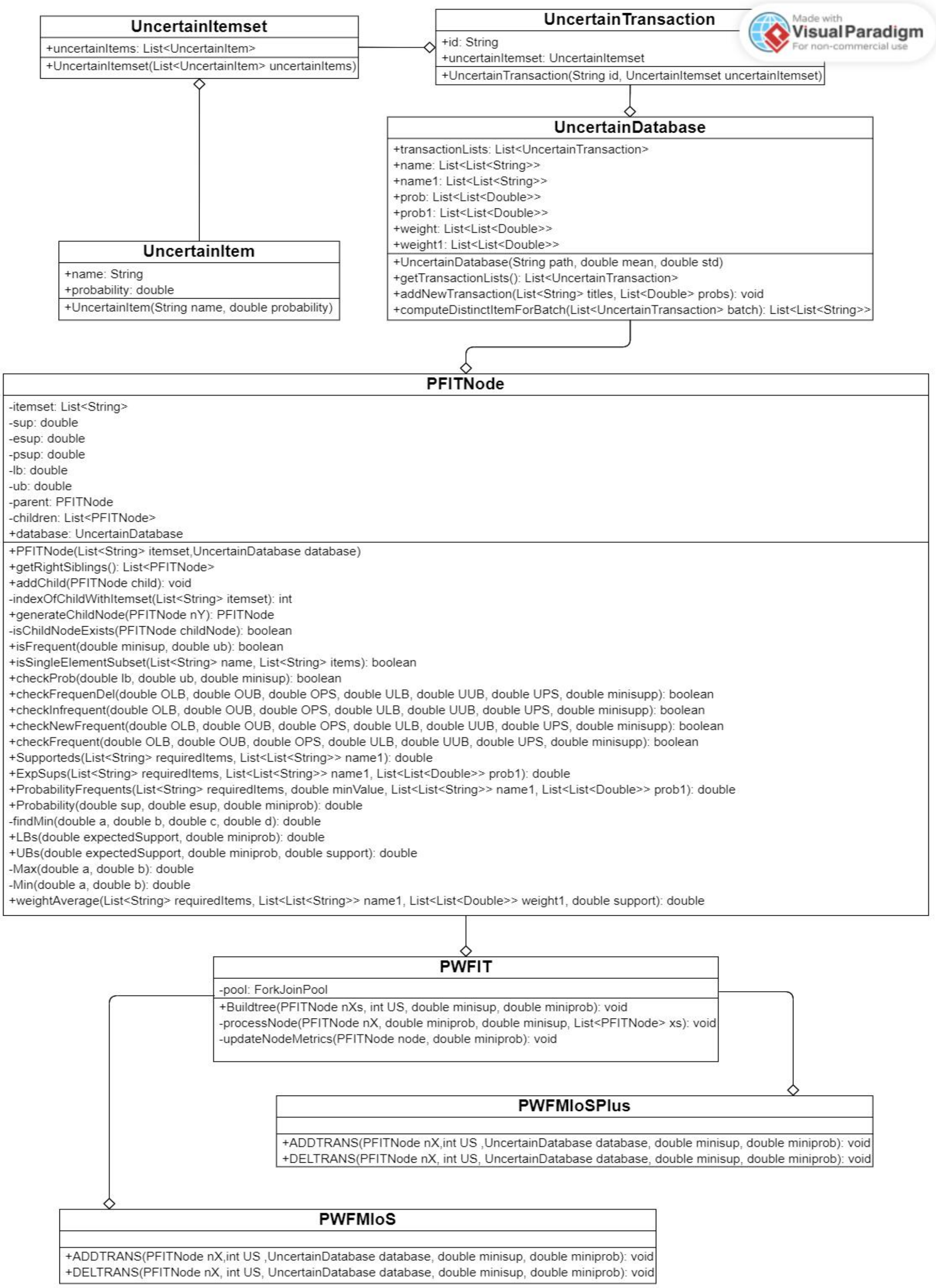



Figure 4.1 UML Class Diagram

## 4.4 Entities and Relationships

- **Uncertain item** : Save each item and probability in pairs
- **Uncertain itemset**: Save a list of uncertain items
- **Uncertain transaction**: Save a list of uncertain items and ID in pairs (ID represents the order of transactions)
- **Uncertain database**: Save a list of uncertain transactions.
  - The **getTransactionList()** function is responsible for getting the items and probabilities stored in each uncertain transaction → uncertain itemset → uncertain items. Attributes Name, weight, probability variable are used to store the value of the entire list when reading the file
  - The **ComputeDistinctItem()** function to define separate items to easily calculate itemset values
- **PFITNode**: Takes as input the Uncertain Database and itemsets. Its task is to handle functions in building a tree based on different itemsets, adding important values such as support, expected support, probability support with weight, lower bound, and upper bound during processing.
- **PWPIT**: Takes as input an empty node containing child nodes representing different itemsets. It includes the BUILDTREE function. Its task is to construct a tree by traversing the child nodes of the parent node. Upon completion, it returns a list of child nodes from the initial empty node.
- **PWFMIoS**: Takes as input the node after BUILDTREE. This node will be traversed through its child nodes, updating the necessary values when adding a new uncertain transaction as well as altering the structure of the tree. Additionally, it deletes old nodes. It returns a new tree with new nodes after processing a newly added uncertain transaction.
- **PWFMIoS+**: Takes as input a node after BUILDTREE. This node will process similarly to PWFMIoS but changes the calculation of Probability Support using Heuristicrules. It returns a tree with new nodes after processing a newly added uncertain transaction.

## 4.5 PFITNode operations

- **getRightSibling():** Returns the list of elements on the right. Check if these right nodes are at the same level as the current node. If so, returns the list of nodes on the right.
- **generateChildNode(requiredItem):** Returns nodes by combining requiredItem with the current nodes. This is achieved by adding the required nodes to the current nodes.
- **isFrequent(minimum support, support):** Returns true or false by checking if an itemset is frequent or not. It determines this by verifying if the support of the itemset is greater than the minimum support.
- **isSingleElementSubset(name, items):** Returns True or False indicating whether name is a subset of items.
- **checkFrequentDel(OLB,OUB,OPS,ULB,UUB,UPS):** Returns true or false by checking conditions to determine if an itemset is infrequent.
- **checkInfrequent(OLB,OUB,OPS,ULB,UUB,UPS):** Returns true or false by checking conditions to determine if an itemset is still frequent in DELTRANS.
- **checkNewfrequent(OLB,OUB,OPS,ULB,UUB,UPS):** Returns true or false by checking if itemsets transitioned from infrequent to frequent.
- **checkFrequent(OLB,OUB,OPS,ULB,UUB,UPS):** Returns true or false by checking if itemsets remain frequent in ADDTRANS.
- **Supported(requiredItem, name1):** Returns the frequency of an itemset by traversing through the list name1 and checking if requiredItems are present; if so, it increments the count by 1.
- **ExpSup(requiredItem, name1, prob1):** Returns the probability of an itemset by traversing through the list name1 and checking if requiredItem is present; if so, it adds the probability.
- **ProbabilityFrequents(requiredItem, name1, prob1):** Returns the probability support of an itemset by traversing through the list name1 and checking if requiredItem is present.

- **weighAverage(requiredItem,name1,weight1, support):** Returns the avarage weight of an itemset. It traverses through name1 and checks if requiredItem is present; if so, it calculates the weight of that itemset.
- **Probability(sup, esup,miniprob):** Returns the smallest value based on Heuristicrules.
- **LBs(expectedsupport, miniprob):** Returns the minimum of lowerbound and 0.
- **UBs(expectedSupport, miniprob, support):** Returns the maximum of upperbound with support.

# 5. Methods

## 5.1 PWFIT

This function is responsible for creating a tree with parameters including: An empty node as input including child nodes as itemset; a sliding window used to reformat the group of transactions you want to process; minimal support for frequency determination; With the probability of items having 1 element and use the minimum probability to calculate and determine the probability support.

It’s performed by going through the child nodes and then checking the frequency of each itemset. If it's infrequent, it checks the same-level nodes on the right to see if they're frequent. If so, it will create the corresponding child nodes and calculate the necessary parameters. The output will be a list of nodes with important values as support, expected support, lower bound, upper bound and weight of probabilistic support (due to redefinition when updating or deleting transactions).

**Function:** Buildtree

**Require:** nXs: PFITNode, US: int, minisup: double, miniprob: double

```
Initialize a new RecursiveAction
    Define compute() method
    Initialize an empty list xs of PFITNode
    For each child nX of nXs in parallel do
        Set nX's support, expected support, LB, UB
        If nX is not frequent then
            Continue to the next child
        Set the probability of nX
        For each right sibling node of nX in parallel do
            If the sibling node is frequent then
                Generate a child node
                Set child's support, expected support, LB, UB
                If the child node's LB <= minisup and UB >=
                minisup then
                    Set the probability of the child
                Synchronize access to xs
                    Add the child node to xs
    Add all nodes in xs to the children of nXs
```

Table 5.1 Pseud code Buildtree function in PFIT with weight

**Example:** List itemset includes ‘A’, ‘B’, ‘C’, ‘D’,’E’ in ***Table 3.1*** can see that, with minimum support (lamda) is 2, minimum probability (t) is 0.6. itemset ‘A’ has support 2 and it is greater than or equal minimum support so it define frequent. If itemset is not frequent, it will continue to check the right parents. The righr parent is frequent so continuing generate children with the itemset not frequent. The condition lowerbound and upper bound keep computing probability support less consume running time cost. In this probability we multiply probability support with weight of itemset. Finally, we add all childnode to root.

**Output:** This is the calculation result from ***Table 3.1***

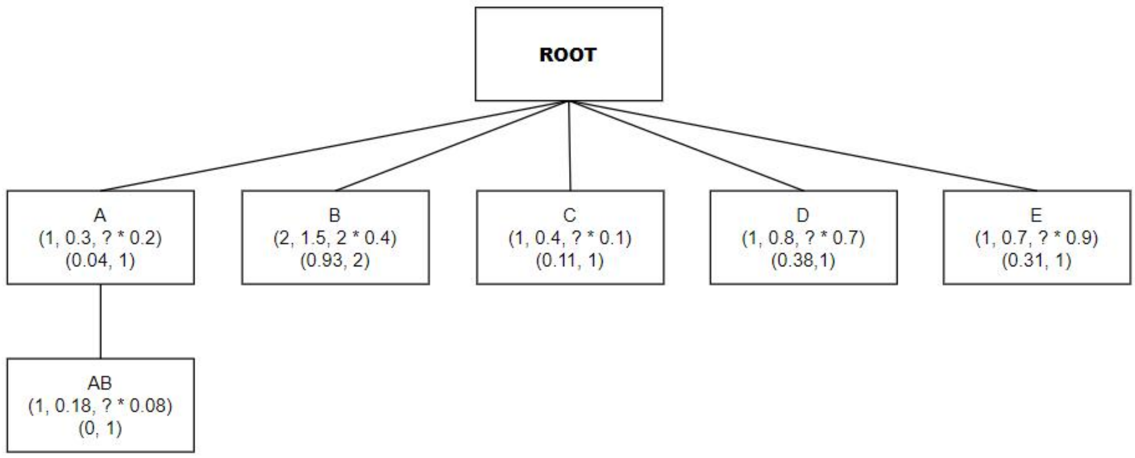



Figure 5.1 Diagram describe the example's result of Buildtree Function

## 5.2 PWFIMoS

PWFIMoS has two tasks: add transactions then update the tree and delete transactions then delete unnecessary transactions. Those 2 tasks are performed by the ADDTRANS and DELTRANS functions below

### *5.2.1 ADDTRANS function*

The ADDTRANS function operates by taking input from the output of the PWFIT entity. Initially, it scrutinizes the child nodes within the existing structure. Upon adding a new transaction, it  updates the values associated with each node, encompassing support, expected support, weight of probability support, lower bound, and upper bound.

Following this, the function proceeds to examine whether any itemsets have transitioned from being infrequent to frequent due to the introduction of the new transaction. If such transitions occur, the function dynamically generates trees from these nodes.

Subsequently, for itemsets that remain frequent, the function extends its scrutiny to the right of each node to identify any emerging frequent nodes. Should new frequent nodes emerge, the function amalgamates them with the existing frequent ones to form child nodes.

With each iteration of child node creation, the function ensures that the values associated with them, such as support metrics and bounds, are accurately updated. This process guarantees the seamless and efficient updating of the transactional dataset while preserving the integrity of support values and tree structures for frequent itemsets.

Here the table describes the conditions for an itemset to change its status from frequent to infrequent and vice versa (use only for ADDTRANS function)

| | Original Lower-bound | Original Upper-bound | Updated Lower-bound | Updated Upper-bound | Original Probabilistic Support | Updated Probabilistic Support | |
|---|---|---|---|---|---|---|---|
| 1 | $< \lambda$ | $\geq \lambda$ | $< \lambda$ | $\geq \lambda$ | $\geq \lambda$ | | $f \rightarrow f$ |
| 2 | $< \lambda$ | $\geq \lambda$ | $\geq \lambda$ | | $\geq \lambda$ | | |
| 3 | $\geq \lambda$ | | | | | | |
| 4 | $< \lambda$ | $\geq \lambda$ | $< \lambda$ | $\geq \lambda$ | $< \lambda$ | $\geq \lambda$ | $i \rightarrow f$ |
| 5 | $< \lambda$ | $\geq \lambda$ | $\geq \lambda$ | | $< \lambda$ | | |
| 6 | | $< \lambda$ | | $\geq \lambda$ | | $\geq \lambda$ | |

Table 5.2 State change based on different conditions(ADDTRANS function)

**Function:** ADDTRANS

**Require:** nX: PFITNode, US: int, database: UncertainDatabase, minisup: double, miniprob: double

**Retrieve** the last value, probability, and weight **from** the database

**Initialize** three lists: childrenCopy, newfre, and frequent

**If** nX has no children **then**

```
        Return
    For each child nY of nX do
        Store nY's original lower bound, upper bound, and probability
        If nY is a subset of the transaction (value) then
            Update nY's support, expected support, lower bound, upper
            bound, and reset probability
            If nY's updated bounds indicate it's potentially frequent then
                Update nY's probability
        If nY transitions to a frequent state then
            Add nY to newfre list
            Generate new child nodes for all right siblings of nY
            Update support and expected support for each child
            Add each child to childrenCopy and as a child of nY
        If nY remains frequent and is a subset of the transaction then
            Add nY to frequent list
        For each node in frequent then
            For each right sibling that is newly frequent then
                Generate and update a new child node
                Add this new child to childrenCopy and as a child of the
                current node
        Add all nodes in childrenCopy to the children of nX
```

Table 5.3 Pseudo code ADDTRANS function in PFMIoS with weight

**Example**: The itemset 'D' is included in the range of the newly added uncertain transaction. Therefore, we update its support, which increases to 2, the expected support is incremented with the new probability and becomes 1.4, the lower bound and upper bound are updated to 0.85 and 2, respectively. Additionally, since the minimum support falls within the range of the lower bound and upper bound, we update the probability support and multiply it by the weight to obtain 2 * 0.9 = 1.8. After updating these necessary values, we check each itemset to see if there are any newFrequent (i - f) ones, such as itemset 'D' transitioning from infrequent to frequent. In this case, we reconstruct the tree based on this 'D' itemset and obtain itemset 'E', which is frequent. Consequently, we combine itemset 'D' with itemset 'E' to form itemset 'D,E'. Upon creating a new subset, we update the necessary values. Furthermore, if an itemset remains unchanged as frequent (f - f) after adding an uncertain transaction, these itemsets will traverse through their right siblings and create a tree. For example, if itemset 'B' remains frequent and itemset 'D' becomes newfrequent, we reconnect them to form 'B,D' and 'B,E', and update the tree accordingly.

**Output:** This output is the calculation result between ***Table 3.1*** and the output of the **Buildtree function** in ***Figure 5.1***

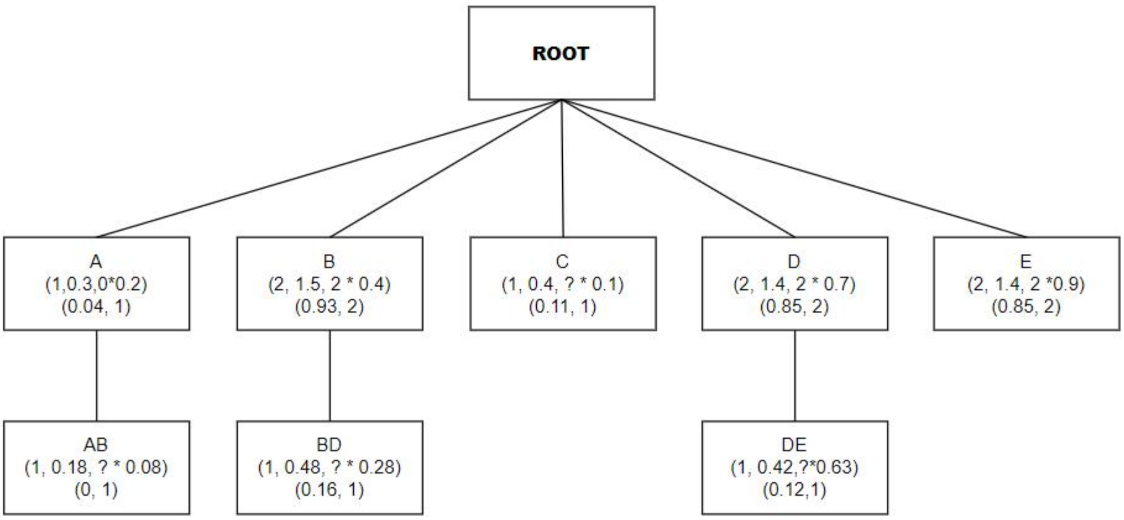



Figure 5.2 Diagram describe the example's result of ADDTRANS Function

### *5.2.2 DELTRANS function*

The DELTRANS function is designed to handle the output of the ADDTRANS function efficiently. It begins by removing the old uncertain transaction, specifically targeting the first one in the queue. Subsequently, it updates crucial values associated with each item set affected by the removal. The next step involves scrutinizing these item sets to determine if any have transitioned from being frequent to infrequent, or vice versa.

Should an item set change from frequent to infrequent, the function promptly removes the child nodes associated with that particular node. On the other hand, if an item set remains frequent but its composition has shifted, the function carefully inspects the neighboring values. If any resulting item sets are now infrequent, they are promptly eliminated to maintain data integrity.

Here the table describes the conditions for an itemset to change its status from frequent to infrequent and vice versa (use only for DELTRANS function)

| | Original Lower-bound | Original Upper-bound | Updated Lower-bound | Updated Upper-bound | Original Probabilistic Support | Updated Probabilistic Support | |
|---|---|---|---|---|---|---|---|
| 1 | $< \lambda$ | $\geq \lambda$ | $< \lambda$ | $\geq \lambda$ | $\geq \lambda$ | $\geq \lambda$ | $f \rightarrow f$ |
| 2 | $\geq \lambda$ | | $\geq \lambda$ | | | | |
| 3 | $\geq \lambda$ | | $< \lambda$ | | | $\geq \lambda$ | |
| 4 | $< \lambda$ | $\geq \lambda$ | $< \lambda$ | $\geq \lambda$ | $\geq \lambda$ | $< \lambda$ | $i \rightarrow f$ |
| 5 | $< \lambda$ | $\geq \lambda$ | | $< \lambda$ | $\geq \lambda$ | | |
| 6 | $\geq \lambda$ | | $< \lambda$ | | | $< \lambda$ | |

Table 5.4 State change based on different conditions(DELTRANS function)

**Function:** DELTRANS

**Require:** nX: PFITNode, US: int, database: UncertainDatabase, minisup: double, miniprob: double

1: **Retrieve** the first set of names, probabilities, and weights **from** the database

2: **Initialize** an infre list to track infrequent nodes

3: **If** nX has no children **then**

4: **Return**

**Copy** nX's children for iteration

**Remove** the first set of names, probabilities, and weights **from** the database

**For each** node nY in the copied list of children **do**

    **Store** nY's original lower bound, upper bound, and probability

    **If** nY's items are a subset of the transaction (list) **then**

        **Update** nY's support, expected support, lower bound, and upper bound without database access

        **Update** nY's probability based on the new bounds

        **Set** nY's probability to 0 if it is not in the frequent bounds anymore

    **For each** child nZ of nY **do**

        **If** nZ becomes infrequent after the transaction removal **then**

            **Remove** nZ **from** nX's children

            **Add** nZ **to** the infre list

        **If** nZ remains frequent but needs updates due to the transaction removal **then**

            **Remove** all children of nZ **from** nX's children if they are related to the transaction

Table 5.5 Pseudo code DELTRANS function in PFMIoS with weight

**Example:** The itemset 'B' is found within the range of the uncertain transaction to be deleted. We will adjust the support values from 2 to 1, expected support from 1.5 to 0.9, lower bound from 0.93 to 0.46, and upper bound from 2 to 1. However, we do not calculate the probability support because the minimum support lies outside the range of the lower bound and upper bound. Subsequently, since we observe that this itemset has become infrequent, we delete its child nodes such as 'B,D' due to the transition from frequent to infrequent (f - i). Next, we proceed to check if any itemsets remain frequent, like 'E', which still maintains its frequency (f - f) after deleting an uncertain transaction. However, there are no right sibling nodes that are all infrequent, so it's not possible to delete the child nodes created by 'E' with infrequent.

**Output:** This output is the calculation result between ***Table 3.1,*** the output of the **Buildtree function** in ***Figure 5.1*** and the output of the **ADDTRANS function** in ***Figure 5.2***

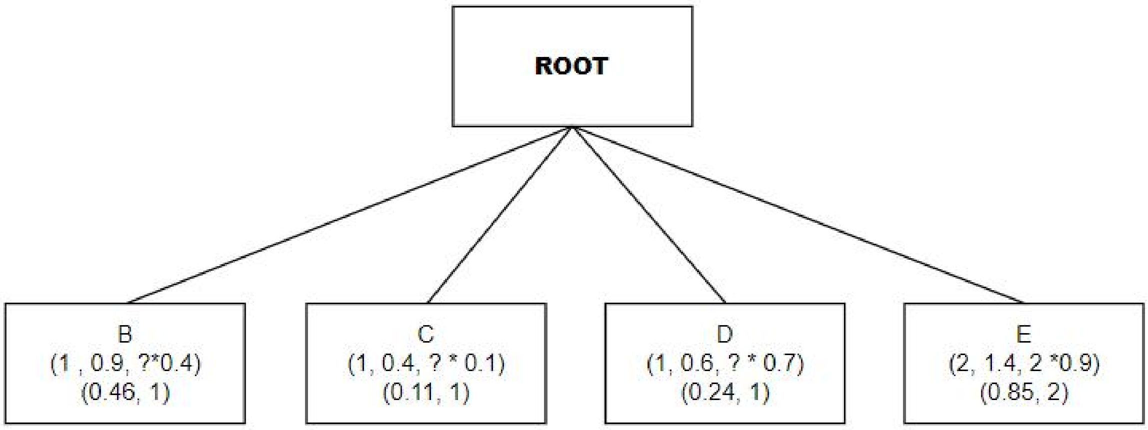



Figure 5.3 Diagram describe the example's result of DELTRANS Function

## 5.3 PWFIMoS+

Similar to PFMIoS but we compute again Probability support which base on ***Heuristicrules*** in document ***Li et al. (2018)[8]***. The new thing we change in this is about add itemset's weight in each probability support to make time consuming in the condition ADDTRANS and DELTRANS more efficient than the older. However, it reduce quite small compared to the older without weight. Here is the formula, we use for this function:

$$\xi = min\begin{pmatrix} \frac{2\sqrt{-2\varepsilon ln(1-\tau)} - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2\eta}, \\ \frac{2\varepsilon - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2\eta}, \\ \frac{\Lambda(X) - \varepsilon + \sqrt{-2\varepsilon ln(1-\tau)}}{\eta}, \\ \frac{\Lambda(X)}{\eta} \end{pmatrix} \tag{5.1}$$

# 6. Experiment setup

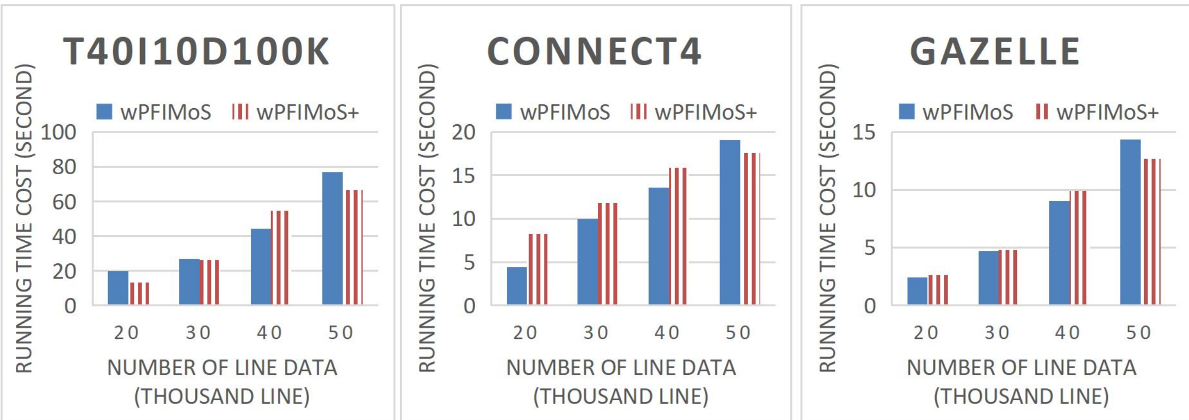
- **Programming language**: Java
- **Compile environment**: java 21.0.2 2024-01-16 LTS, Java(TM) SE Runtime Environment (build 21.0.2+13-LTS-58), Java HotSpot(TM) 64-Bit Server VM (build 21.0.2+13-LTS-58, mixed mode, sharing)
- **IDE used for compilation**: Visual Studio Code
- **Device**: Laptop LENOVO AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz and 8.00 GB of main memory
- **Source code: GitHub**
- **Datasets**:

| Datasets | Transactions count | Mean | Variance |
|---|---|---|---|
| T40I10D100K | 100 000 | 0.79 | 0.61 |
| CONNECT4 | 67 557 | 0.78 | 0.65 |
| GAZELLE | 59 602 | 0.94 | 0.08 |

Table 6.1 Details about the datasets used for experiment

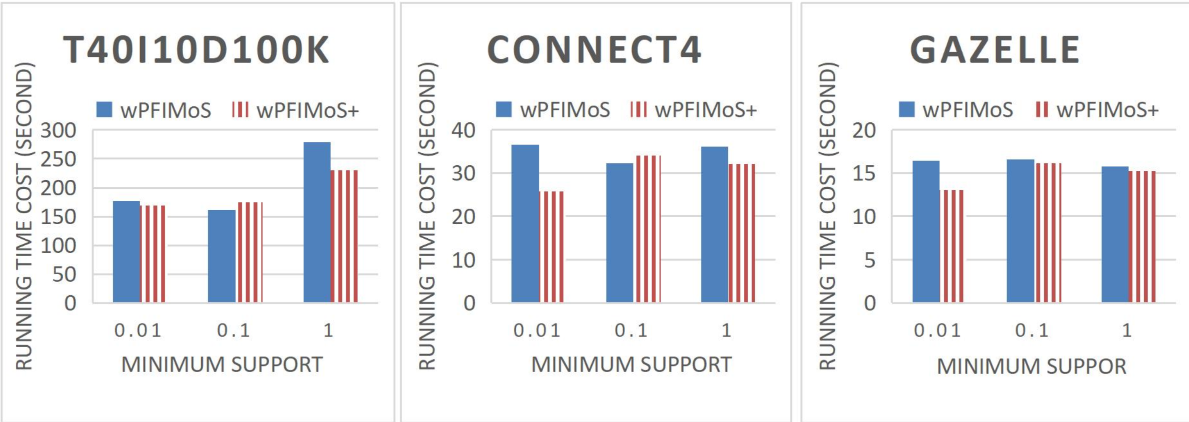
# 7. Experiment result and discussion

## 7.1 Experiment result




a. $\lambda_r = 0.9, \eta = 10K$ b. $\lambda_r = 0.9, \eta = 10K$ c. $\lambda_r = 0.9, \eta = 10K$

Figure 7.1 Effect of Number of Line Data(runtime cost)




d. $\tau = 0.9, \eta = 10K$ e. $\tau = 0.9, \eta = 10K$ f. $\tau = 0.9, \eta = 10K$

Figure 7.2 Effect of Number of Minimum Support(runtime cost)

## 7.2 Discussion

- **According to *Figure 3.1:***
  - There is a clear upward trend in running time cost as the number of line data increases for both algorithms, which is a typical behavior as more data generally requires more processing time.
  - PWFIMoS+ shows an advantage over PWFIMoS in handling larger datasets, particularly evident in the T40I10D100K dataset at 50 thousand lines of data.
  - The incremental running time cost with increasing data volume appears to be more linear for PWFIMoS+, while PWFIMoS exhibits a steeper increase, especially in the T40I10D100K dataset. This indicates better scalability of PWFIMoS+ under heavier data loads.
- **According to *Figure 3.2***:
  - Across all datasets, PWFIMoS+ consistently performs better (i.e., lower running time) than PWFIMoS. This suggests that the enhancements made in the PWFIMoS+ algorithm effectively reduce computation time compared to its predecessor.
  - The running time cost tends to decrease for both algorithms as the minimum support increases from 0.01 to 1. This is an expected behavior because a higher minimum support threshold usually results in fewer itemsets being considered frequent, thus reducing the computational workload.

- The T40I10D100K dataset appears to be more computationally demanding than the CONNECT4 and GAZELLE datasets, as indicated by the overall higher running times for both algorithms on this dataset.
- The relative difference in performance between PWFIMoS and PWFIMoS+ is most pronounced at the lower minimum support level (0.01), particularly in the T40I10D100K dataset. This suggests that the improvements in PWFIMoS+ are especially beneficial when dealing with denser datasets and more stringent (lower) minimum support thresholds.
- While all datasets show improved running times at higher minimum support levels, the T40I10D100K dataset demonstrates a less pronounced improvement compared to the other datasets. This could indicate scalability issues with larger or more complex datasets and might be an area for further optimization.

Overall, these diagrams suggest that the PWFIMoS+ algorithm is more efficient than PWFIMoS, particularly when dealing with larger datasets and when the minimum support threshold is high. Additionally, the diagrams also indicate that algorithm performance can vary significantly depending on the characteristics of the dataset being processed.

# 8. Conclusion

Our research into Probabilistic Weighted Frequent Itemset Mining over Uncertain Data Streams has presented a approach that integrates the concepts of weight and probability in the analysis of uncertain data streams. This integration is pivotal in addressing the dynamic and inherently uncertain nature of streaming data, which poses significant challenges in the field of data mining.

The introduction of the PFIT, PFIMoS, and PFIMoS+ algorithms represents a significant leap forward in our ability to efficiently and effectively mine weighted frequent itemsets from such streams. The 'weighted' aspect of these algorithms is particularly noteworthy, as it allows for a more nuanced understanding of the data by considering not only the occurrence frequency of itemsets but also their respective weights. This approach acknowledges that not all itemsets contribute equally to the insights derived from the data, hence offering a more sophisticated analysis tool that aligns more closely with real-world complexities.

Through our experimental evaluations, we have demonstrated that the incorporation of weights into the probabilistic mining process significantly enhances the performance and accuracy of the mining task. The weighted approach enables the algorithms to prioritize the analysis of itemsets based on their relevance or importance, which is crucial for applications where the significance of the data varies.

Moreover, our experiments using synthetic and real-life datasets have unequivocally shown the superior efficacy of our proposed methods in terms of runtime and memory usage. This not only validates our approach but also emphasizes the practical utility of weighted itemset mining in uncertain data streams, offering potential applications in a wide array of domains such as market basket analysis, real-time monitoring systems, and beyond.

In conclusion, the weighted dimension introduced in our probabilistic frequent itemset mining methodology has opened up new avenues for more refined and relevant data analysis in the face of uncertainty. The success of this approach underscores the importance of continuing to explore and optimize weighted mining techniques, with future work aimed at further enhancing the efficiency and applicability of these algorithms in other complex data environments. Our research thus provides a substantial contribution to the ongoing development of data mining technologies, offering a robust, efficient, and adaptable solution for uncovering the intricate patterns hidden within uncertain data streams.

## REFERENCES

[1] Aggarwal, C. C., Li, Y., Wang, J., & Wang, J. (2009). Frequent pattern mining with uncertain data. In Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining (pp. 29–38). ACM

[2] Akbarinia, R., & Masseglia, F. (2013). Fast and exact mining of probabilistic data streams. In Machine learning and knowledge discovery in databases (pp. 493–508).Springer.

[3] Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., & Zuefle, A. (2009). Probabilistic frequent itemset mining in uncertain databases. In Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining (pp. 119–128). ACM.

[4] Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., & Züfle, A. (2012). Probabilistic frequent pattern growth for itemset mining in uncertain databases. In Scientific and statistical database management (pp. 38–55). Springer

[5] Chui, C.-K., Kao, B., & Hung, E. (2007). Mining frequent itemsets from uncertain data. In Advances in knowledge discovery and data mining (pp. 47–58). Springer.

[6] Cuzzocrea, A., & Leung, C. K. (2016). Computing theoretically-sound upper bounds to expected support for frequent pattern mining problems over uncertain big data. In International conference on information processing and management of uncertainty in knowledge-based systems (pp. 379–392). Springer.

[7] Leung, C. K.-S., & MacKinnon, R. K. (2014). Blimp: A compact tree structure for uncertain frequent pattern mining. In Data warehousing and knowledge discovery (pp. 115–123). Springer.

[8] Li, H., Zhang, N., Zhu, J., Wang, Y., & Cao, H. (2018). Probabilistic Frequent Itemset Mining over Uncertain Data Streams. Proceedings of the 2018 International Conference on Database Systems for Advanced Applications, 11382, 646–661.

[9] Li, Z., Chen, F., Wu, J., Liu, Z., & Liu, W. (2020). Efficient weighted probabilistic frequent itemset mining in uncertain databases. Expert Systems, e12551. https://doi.org/10.1111/exsy.12551

[10] Lixin, L., Xiaolin, Z., & Huanxiang, Z. (2014). Mining of probabilistic frequent item  sets over uncertain data streams. In Web information system and application con  ference (wisa), 2014 11th (pp. 231–237). IEEE.

[11] Sun, L., Cheng, R., Cheung, D. W., & Cheng, J. (2010). Mining uncertain data withprobabilistic guarantees. In Proceedings of the 16th acm sigkdd international conference on knowledge discovery and data mining (pp. 273–282). ACM.

[12] Mike Novey, Tülay Adali, Anindya Roy.(2010). A Complex Generalized Gaussian Distribution— Characterization, Generation, and Estimation. In IEEE Transactions on Signal Processing ( Volume: 58, Issue: 3). 10.1109/TSP.2009.2036049

[13] Jun Han, Claudio Moraga. (2005). The influence of the sigmoid function parameters on the speed of backpropagation learning. In Computational Models of Neurons and Neural Nets. https://doi.org/10.1007/3-540-59497-3_175

[14] Seok-Ho Chang, Pamela C. Cosman, Laurence B. Milstein. (2011) . Chernoff-Type Bounds for the Gaussian Error Function. In IEEE Transactions on Communications. 10.1109/TCOMM.2011.072011.100049

[15] Lin, J.CW., Gan, W., Fournier-Viger, P. et al. Weighted frequent itemset mining over uncertain databases. Appl Intell 44, 232–250 (2016). https://doi.org/10.1007/s10489-015-0703-9

[16] Toon Calders, Calin Garboni, Bart Goethals. (2010). Approximation of Frequentness Probability of Itemsets in Uncertain Data. In 2010 IEEE International Conference on Data Mining. 10.1109/ICDM.2010.42.

[17] Chengjie Luo, Clement Yu, and Jorge Lobo, Gaoming Wang, Tracy Pham, Clement Yu. (1996). Computation of Best Bounds of Probabilities from Uncertain Data. https://doi.org/10.1111/j.1467-8640.1996.tb00276.x

[18] Razieh Davashi. (2021). UP-tree & UP-Mine: A fast method based on upper bound for frequent pattern mining from uncertain data. https://doi.org/10.1016/j.engappai.2021.104477

[19] Carson Kai-Sang Leung, Boyu Hao. (2009). Mining of Frequent Itemsets from Streams of Uncertain Data. In 2009 IEEE 25th International Conference on Data Engineering. 10.1109/ICDE.2009.157

[20] Islam, M.S., Kar, P.C., Samiullah, M. et al. Discovering probabilistically weighted sequential patterns in uncertain databases. Appl Intell 53, 6525–6553 (2023). https://doi.org/10.1007/s10489-022-03699-7

[21] Gan, W., Lin, J.CW., Fournier-Viger, P., Chao, HC. (2016). Mining Recent High Expected Weighted Itemsets from Uncertain Databases. In: Li, F., Shim, K., Zheng, K., Liu, G. (eds) Web Technologies and Applications. APWeb 2016. Lecture Notes in Computer Science(), vol 9931. Springer, Cham. https://doi.org/10.1007/978-3-319-45814-4_47