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

FINAL REPORT DESIGN AND ANALYSIS ALGORITHMS

WEIGHTED PROBABILISTIC FREQUENT ITEMSET IN UNCERTAIN DATABASE

Instructor: **PhD. NGUYỄN CHÍ THIỆN**
Author: **PHAN MINH HOANG – 521H0501**
DO HOANG DUY – 521H0395
NGUYEN SONG HUNG – 521H0399

HO CHI MINH CITY, 2024

VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FALCUTY OF INFORMATION TECHNOLOGY

FINAL REPORT DESIGN AND ANALYSIS ALGORITHMS

WEIGHTED PROBABILISTIC FREQUENT ITEMSET IN UNCERTAIN DATABASE

Instructor: **PhD. NGUYỄN CHÍ THIỆN**
Author: **PHAN MINH HOANG – 521H0501**
DO HOANG DUY – 521H0395
NGUYEN SONG HUNG – 521H0399

HO CHI MINH CITY, 2024

ACKNOWLEDGMENTS

We sincerely thank the Information Technology department for providing us with the opportunity to approach and complete this report. We sincerely thank PhD. Nguyen Chi Thien for guiding us to complete this report.

During the process of preparing the report, due to our limited knowledge and experience, the report could not avoid shortcomings. We look forward to receiving feedback from you so that we can learn more skills, gain more experience, and improve further.

THESIS WAS COMPLETED
AT TON DUC THANG UNIVERSITY

I hereby certify that this is my thesis project and was conducted under the guidance of PhD. Nguyen Chi Thien. The research contents and results presented in this thesis are honest and have not been published in any form before. The data used for analysis, comments, and evaluations were collected by the author from various sources which are clearly indicated in the reference section

Furthermore, this thesis also includes some comments, evaluations, and data from other authors and organizations which are properly cited and referenced

If any fraudulent activity is detected, I take full responsibility for the content of my thesis. Ton Duc Thang University is not liable for any copyright infringement or violation caused by me during the implementation process (if any)

Ho Chi Minh City, January 7, 2024

Author

(Signature and full name)

Phan Minh Hoang

SUMARY

In this research, we will discuss about how to find weight probabilistic frequent itemset in uncertain database

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS

DB	Database
minSup	Minimum support
t	Threshold
a	Scale factor
w	A weight table for the itemset I
I	An itemset of size m
T	Transaction of database
p_{ij}	An existential probability for item I_i appearing in the transaction T_j

Chapter 1 – INTRODUCTION

Frequent itemset mining has long been a fundamental aspect of data mining, extensively studied and applied in conventional methods. However, traditional approaches predominantly operate under the assumption of certainty in data, overlooking the inherent uncertainties present in real-world scenarios. In response to this limitation, recent research efforts have shifted towards addressing the challenges posed by uncertain data, particularly in the context of mining frequent itemset.

In uncertain data scenarios, the certainty of the presence or absence of items in transactions is not guaranteed. This introduces a layer of complexity that necessitates novel models and techniques for extracting meaningful patterns. The authors of the paper under consideration contribute to this emerging field by presenting a sophisticated probability model designed specifically for capturing the frequency of itemset within uncertain datasets.

A key innovation introduced by the authors involves the incorporation of an efficient pruning technique for the generation of candidate itemset. This technique plays a crucial role in streamlining the mining process, ensuring that only the most promising candidates are considered. By addressing the computational challenges associated with uncertain data, the proposed approach enhances the efficiency and effectiveness of frequent itemset mining.

Furthermore, the paper underscores the importance of considering the significance of individual items within specific contexts. Unlike traditional methods that treat all items equally, the authors argue for a nuanced approach that accounts for the relative importance of each item. This consideration is reflected in their probability model, where item weights are assigned to convey their relative significance within the uncertain dataset

Chapter 2 - Related works

In this research, we will find the w-PFIs in the uncertain database using the Apriori algorithm, w-PFI candidate generation and pruning (Algorithm 2), w-PFI candidate generation and pruning based on a probability model (Algorithm 3) as stated in the paper "Efficient weighted probabilistic frequent itemset mining in uncertain databases" by Zhiyang Li, Fengjuan Chen, Junfeng Wu, Zhaobin Liu, and Weijiang Liu in 2020.

We will design classes for searching wPFI in uncertain databases and implement the Apriori algorithm, Algorithm 2 and 3 mentioned in the paper [1] to see if these algorithms are really effective in mining wPFI in uncertain databases, including fully uncertain databases or not.

In addition, we also analyze the complexity of functions that have a major impact on the above algorithms.

Chapter 3 - Preliminaries and Problem Statements

The challenge of mining frequent itemset in uncertain databases arises from the necessity to analyze data with uncertain presence or absence of items in transactions. Traditional methods for frequent itemset mining are not directly applicable to such uncertain data, assuming data certainty. Hence, there is a demand to devise effective algorithms and methodologies for identifying frequent itemset in uncertain databases while accommodating the inherent uncertainty. This entails overcoming obstacles like quantifying itemset frequency in the face of uncertainty The objective is to offer a comprehensive solution for mining frequent itemset in uncertain databases, ensuring accuracy and scalability, and facilitating the extraction of valuable insights uncertain data.

3.1 Definition 1 – Uncertain Database

An uncertain DB is a database that contains uncertain data, which means the data may not be accurate or complete. Unlike traditional databases, uncertain datasets assign an existential probability, denoted as $p_{ij} = Pr(I_i \subseteq T_j)$, to each item I_i, representing the likelihood of I_i being present in transaction T_j [1]

3.1.1 Parly uncertain database

It is an uncertain database where each transaction contains items that are associated with a probability from [0, 1]. Table 1 gives an example of our partly uncertain dataset

CustomerID	Transaction (item, probability)			
1	(PC, 0.5)	(Mouse, 1.0)	(Phone, 0.7)	
2	(PC, 1.0)		(Phone, 0.8)	(DVD, 0.3)

Table 1 – Example of partly uncertain database

3.1.2 Fully uncertain database

It is an uncertain database where each transaction contains items that are associated with a probability from (0, 1) not include 0 and 1. Table 2 gives an example of our fully uncertain dataset

CustomerID	Transaction (item, probability)			
1	(PC, 0.5)	(Mouse, 0.4)	(Phone, 0.7)	(DVD, 0.85)
2	(PC, 0.1)	(Mouse, 0.2)	(Phone, 0.8)	(DVD, 0.3)

Table 2 - Example of fully uncertain database

3.2 Definition 2 – Weight Table

A weight table is a table that assigns a real-valued weight to each item in an itemset, indicating its importance. For example, Table 3 shows a weight table for four items: PC, Mouse, Phone and DVD. The weight of an item can be set by the user according to their domain knowledge or specific requirement. The weight table is used for weighted frequent itemset mining in uncertain databases [1]

Item	Probability
PC	0.78
Mouse	0.54
Phone	0.89
DVD	0.34

Table 3 – Example of weight table

3.3 Definition 3 – Probabilistic Frequent Itemset

Given uncertain dataset DB, a minimum support - $msup$, and a probabilistic frequent threshold - t, an itemset $X \subseteq I$ is a probabilistic frequent itemset if and only if $Pr(sup(X) \geq msup) \geq t$. The $Pr(sup(X) \geq msup)$ is calculated by this formula below: [1]

$$P_{\geq i}(X) = \sum_{S \subseteq T, |S| \geq i} \left(\prod_{t \in S} P(X \subseteq t) . \prod_{t \in T - S} (1 - P(X \subseteq t)) \right) \text{ [2]}$$

3.4 Definition 4 – Itemset Weight

For an itemset X ⊆ I, its weight, denoted as w(X), reflects the average weight of its constituent items. It's calculated by summing the weights of all items in X and dividing by the total number of items. This weight serves as a measure of the importance of itemset based on pre-assigned weights, facilitating the mining of weighted frequent itemset within uncertain databases. [1]

$$W(X) = \frac{1}{|X|} \sum_{x \in X} W(X)$$

3.5 Definition 5 – Weighted probabilistic frequent itemset

Given an uncertain dataset DB, a weight table - w, a minimum support - $msup$, and a probabilistic frequent threshold - t, an itemset $X \subseteq I$ is a weighted probabilistic frequent itemset if and only if [1]

$$w(X) Pr(sup(X) \geq msup) \geq t$$

3.5 Corollary 1

If an itemset X is a w-PFI, then any item I that is added to X to form a larger itemset X' must have a smaller weight than the smallest weight in X, and must not be a member of any size-1 w-PFI. [1]

$$X' = X \cup I_s \text{ is not a } w-\text{PFI if } w(I_s) \geq \min_{x \epsilon X} w(x) \text{ and } I_s \epsilon I - I'$$

where I' = consists of all the size-1 items of $WPFI_{k-1}$

3.6 Corollary 2

If an itemset X is a w-PFI, then any item I that is added to X to form a larger itemset X' must following the condition:

$$Min\{\mu^x, \mu^{I_i}\} \geq \mu\text{^}$$

Where μ^ is the solution of the equation $1 - F(msup - 1, \mu) = t/m$, and $m = max\{w(X), w(Is)\}$ [1]

3.7 Corollary 3

The Corollary 3 is defined as following: If an itemset X is a w-PFI, then any item I that is added to X to form a larger itemset X' must following the condition.

$X' = X \cup I_s$ is probably not a w – PFI if $\mu^X \, \mu^{I_s} / n < \alpha\mu$ [1]

Chapter 4 - Methods

4.1 Apriori Algorithm (Algorithm 1)

A w-PFI is a set of items that have different importance and occur frequently with high probability in the data. The algorithm works as follows:

- It scans the database once to find the size-1 w-PFIs, which are the items that satisfy the minimum support and probabilistic frequent threshold.
- It generates the size-k w-PFI candidates from the size-(k-1) w-PFIs by adding one item at a time, and prunes the candidates that are not likely to be w-PFIs based on some properties and probability models.
- It scans the database again to verify the size-k w-PFI candidates by computing their weighted frequency probability and comparing it with the probabilistic frequent threshold.
- It repeats the above steps until no more w-PFI candidates can be generated.

Algorithm 1: Apriori
Input: A uncertain dataset **DB**, the dataset size **n**, a weight table **w**, a minimum support **msup**, a probabilistic frequent threshold **t**, a scale factor **a** Output: A weighted probabilistic frequent itemset **WPFI** 1. Initialize WPFI = ∅, I is the set of all the items in DB. 2. $WPFI_1$, μ_1 = Scan_Find_Size_1_wPFI(I, DB, w, msup, t) 3. Add $WPFI_1$ into WPFI. 4. k = 2. 5. while $WPFI_k \neq \emptyset$ do. 6. Candidate C_k = wPFI_Apriori_Gen($WPFI_{k-1}$, I, μ_{k-1}, w, α, n, msup, t) 7. $WPFI_k$, μ_k = Scan_Find_Size_k_wPFI(C_k, DB, w, msup, t) 8. Add $WPFI_k$ into WPFI. 9. k = k + 1. 10. end while

Table 4 Pseudocode of algorithm 1 [1]

To implement algorithm1 we need 3 functions inside it:

- Scan_Find_Size_1_wPFI (using definition 5)
- WPFI_Apriori_Gen (this is algorithm 2 and 3, I will discuss it in next section)
- Scan_Find_Size_k_wPFI (using definition 5)

4.2 WPFI_Apriori_Gen (Algorithm 2)

w-PFI candidate generation and pruning algorithm is the wPFI_Apriori_Gen function in the algorithm 1. It uses the Corollary 1 If an itemset X is a w-PFI, then any item I that is added to X to form a larger itemset X' must have a smaller weight than the smallest weight in X, and must not be a member of any size-1 w-PFI. Based on the corollary 1 we have a pseudocode for generating wPFI candidate like table 5 below.

Algorithm 2: WPFI_Apriori_Gen

Input: A size-(k − 1) w-PFI set $\boldsymbol{WPFI_{k-1}}$, the itemset **I**, the weight table **w**, probabilistic frequent threshold **t**
Output: A size-k w-PFI candidate set **Ck**

1. Initialize Ck = ∅
2. $I' =$ consists of all the size-1 items of $WPFI_{k-1}$
3. for each itemset $X \in WPFI_{k-1}$ do
4. for each item $I_i \in I' - X$ do
5. if $w(X \cup Ii) \geq t$ then
6. Add $X \cup I_i$ into Ck
7. End if
8. End for
9. $I_m = argmin_{x \in X} w(x)$
10. for each item $I_i \in I - I'$ do
11. if $w(X \cup I_i) \geq$ t then
12. Add $X \cup I_i$ into Ck
13. End if
14. End for
15. End for

Table 5 Pseudocode of algorithm 2 [1]

4.3 WPFI_Apriori_Gen (Algorithm 3)

w-PFI candidate generation and pruning based on probability model algorithm is the wPFI_Apriori_Gen function in the algorithm 1. It uses the Corollary 2 and Corollary 3 and we have pseudocode for generating wPFI candidate like table 6 below.

Algorithm 3: WPFI_Apriori_Gen

Input: A size-(k − 1) w-PFI set $\boldsymbol{WPFI_{k-1}}$, the itemset **I**, the weight table **w**, probabilistic frequent threshold **t**, mean $\boldsymbol{\mu}$, scale **α**
Output: A size-k w-PFI candidate set **Ck**

1. Initialize Ck = ∅
2. I' = consists of all the size-1 items of $WPFI_{k-1}$
3. m = $max\{wi|w_i \in w\}$
4. $\mu\text{^} = \mu \mid 1 - F(msup - 1, \mu) = t/m$
5. for each itemset $X \in WPFI_{k-1}$ do
6. for each item $I_i \in I' - X$ do
7. if $w(X \cup Ii) \geq t$ then
8. if $\min(\mu^x, \mu^{I_i}) \geq$ μ^ And $\mu^x.\mu^{I_i} \geq \alpha n \mu\text{^}$ then
9. Add $X \cup I_i$ into Ck
10. End if
11. End if
12. End for
13. $I_m = argmin_{x \in X w(x)}$
14. for each item $I_i \in I - I'$ do
15. if $w(X \cup I_i) \geq$ t then
16. if $\min(\mu^x, \mu^{I_i}) \geq$ μ^ And $\mu^x.\mu^{I_i} \geq \alpha n \mu\text{^}$ then
17. Add $X \cup I_i$ into Ck
18. End if
19. End if
20. End for
21. End for

Table 6 Pseudocode of algorithm 3 [1]

Chapter 5 - Experiment Setup

5.1 Mapping

	Mathematic	Memory (Data Structure)
Item	$\boldsymbol{I_i = \{item, prob\}}$	Item class: + int item + double prob
ItemSet	$\boldsymbol{I = \{I_1, I_2, \dots, I_m\}}$	Set<Item>
Weight Table	$\boldsymbol{W = \{e \mid e = \{item, prob\}, item \in I\}}$	Map<Item, Double>
DB	DB = $\{\boldsymbol{T_1, T_2 \dots T_m}\} \mid \boldsymbol{T = T \subseteq \{I_1, I_2, \dots, I_m\}}$	List<ItemSet>
minSup	$\boldsymbol{minSup \in N^*}$	Integer
threshold	$\boldsymbol{t \in \mathbb{R}}$	Double
Scale factor	$\boldsymbol{\alpha \in \mathbb{R}}$	Double

Table 7 Mapping table

5.2 Design Java Class

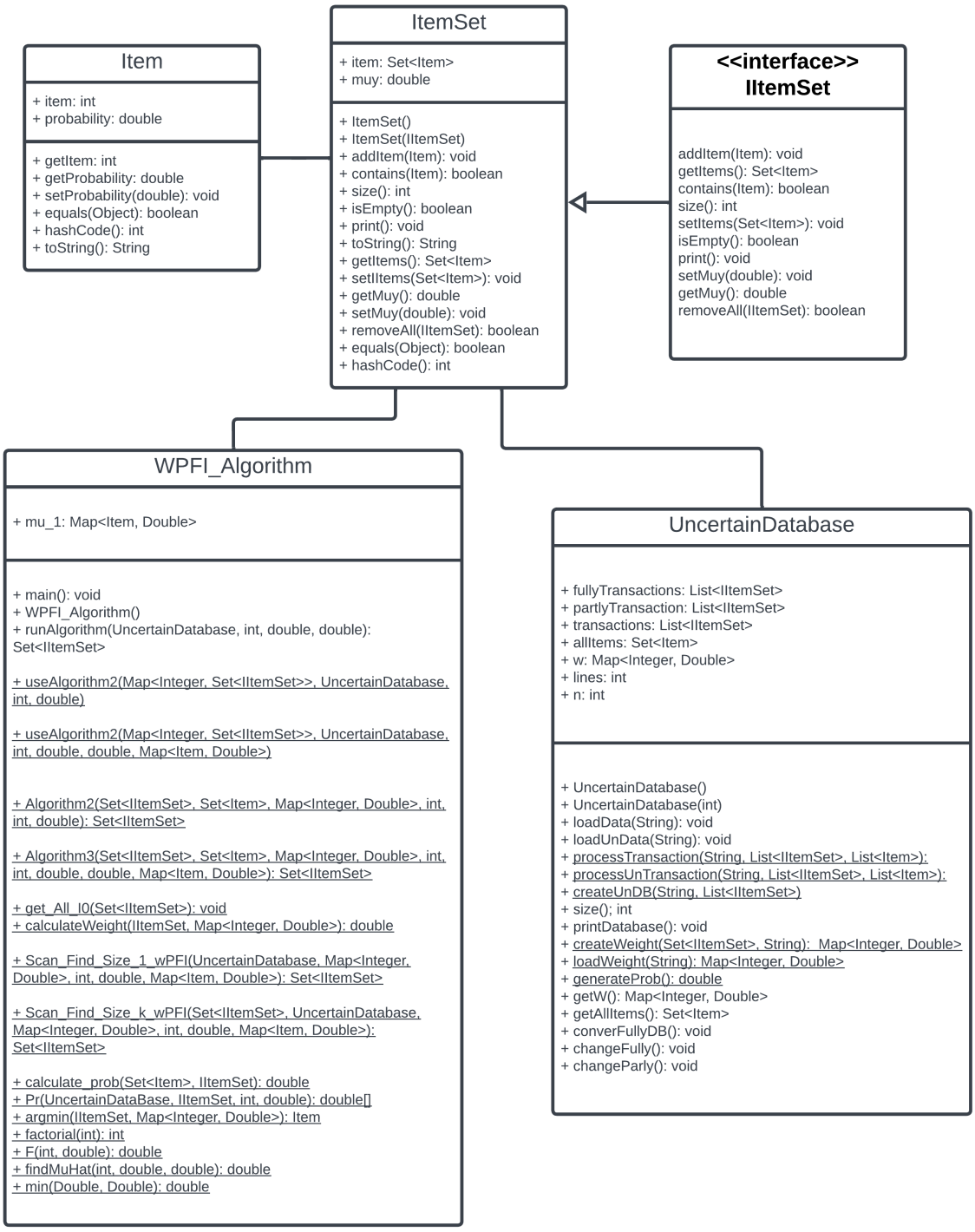

Figure 1 – Overall architecture of Java class diagram for uncertain mining

5.3 Experimental Setting

We conduct the experiments using four dataset which come from the Frequent Itemset Mining (FIMI) repository and the Open-Source Data Mining Library (SPMF) repository. The characteristics of datasets are shown in Table 8 below, all of which are actually deterministic databases

Dataset	No. of transaction	No. of items	Average Length	minSup	t	a
Connect	67,557	129	43	0.2n	0.7	0.6
Accidents	340,183	468	33.8	0.1n	0.7	0.6
USCensus	1,000,000	396	48	0.1n	0.7	0.6
T40I10D100K	100,000	942	39.6	0.01n	0.7	0.6

Table 8 Characteristics of datasets

In order to create uncertainties in these databases, we allocate a probability of existence to each item in the transactions. The probabilities are derived from a Gaussian distribution with a mean of 0.5 and a variance of 0.125. Concurrently, the weight table ‘w’ for the items in each database is generated from a uniform distribution in the range (0, 1]. [1]

We will use the first 10K transactions of the above four databases in practice to make the runtime more reasonable.

Chapter 6 - Experiment Results and Discussion

6.1 Test with partly uncertain database

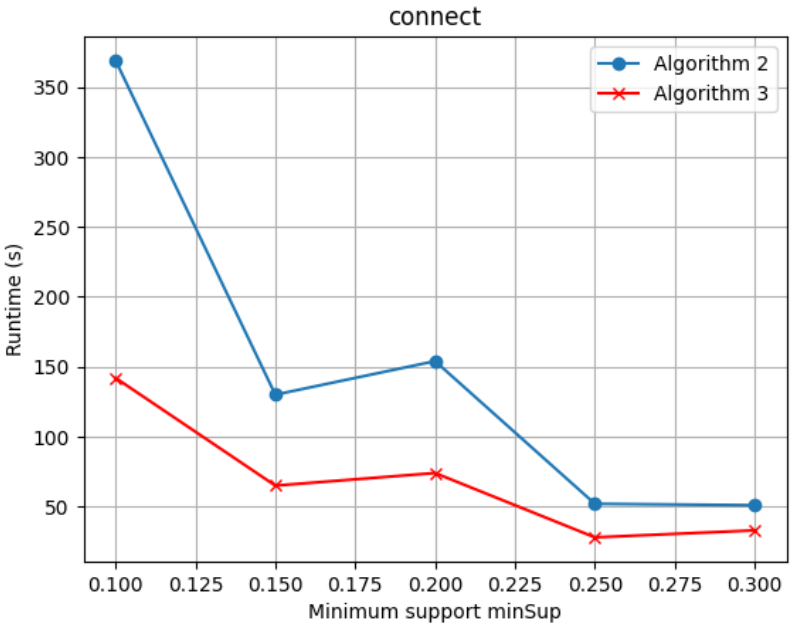

Figure 2 Test with connect dataset

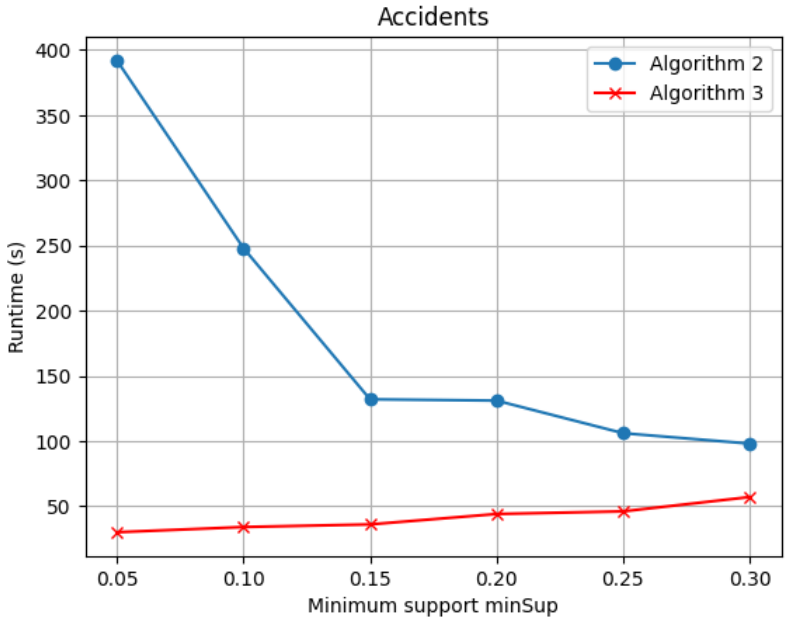

Figure 3 Test with accidents dataset

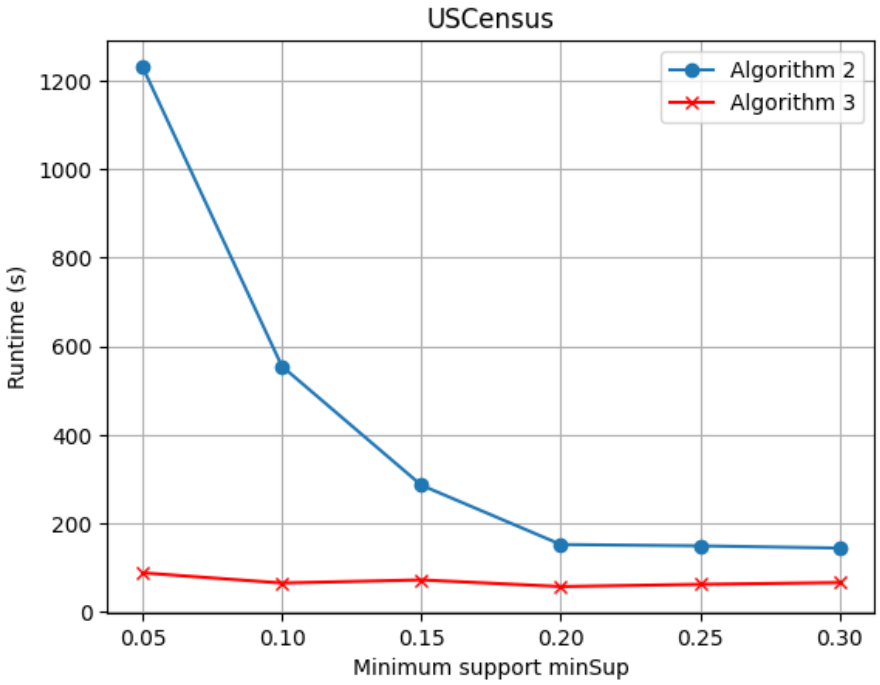

Figure 4 test with USCensus dataset

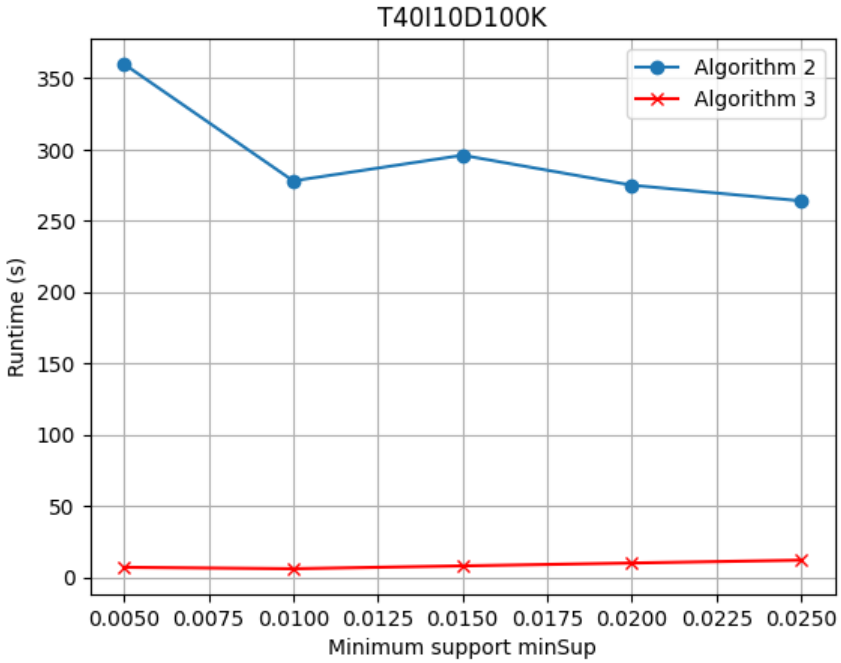

Figure 5 Test with T40I101D100k

Figure 2, 3, 4, 5 illustrates the performance of algorithms 2 and 3 in relation to the minimum support, minSup. As a rule, as minSup decreases, the execution time of the algorithms increases. This is because a lower minSup leads to the discovery of more weighted-PFIs (w-PFIs), which in turn necessitates the verification of more w-PFI candidates. The probabilistic frequent threshold, t, is set to the default value of 0.7.

6.2 Test with fully uncertain database

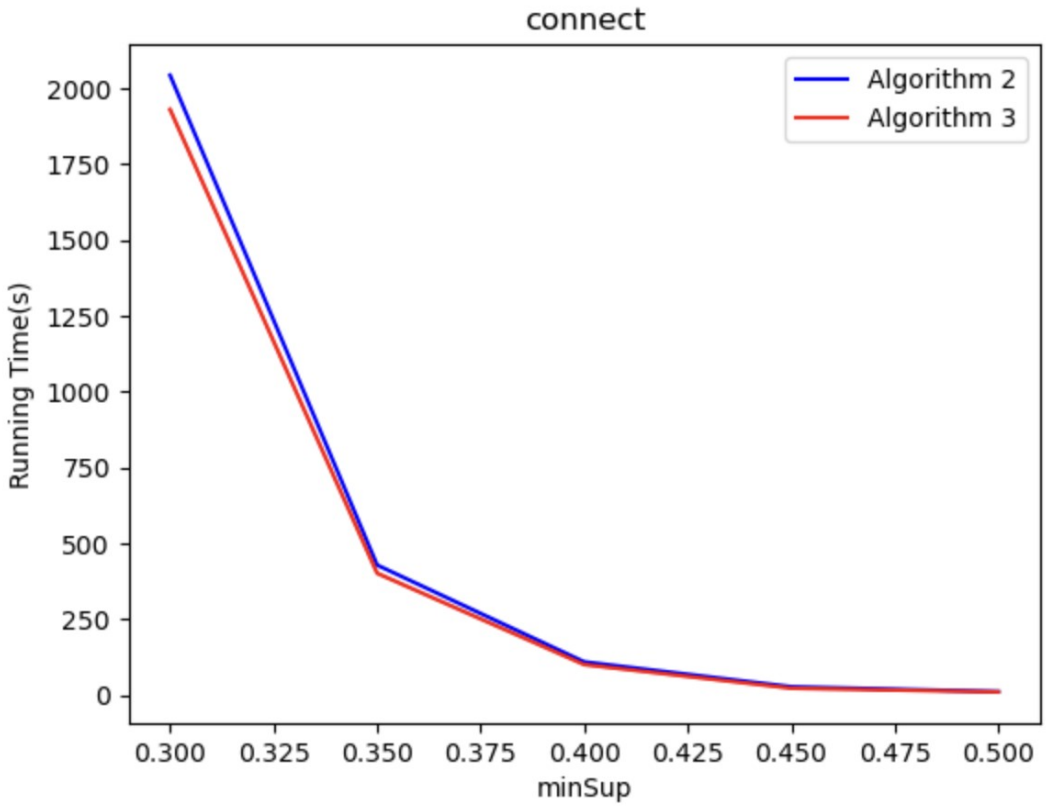

Figure 6 Test with fully uncertain connect dataset

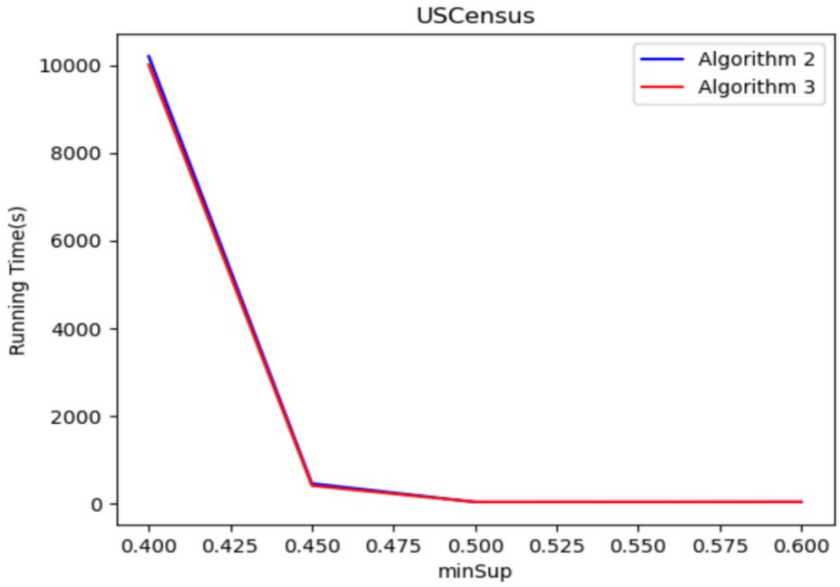

Figure 7 Test with fully uncertain USCensus dataset

Figures 6 and 7 show the performance of algorithms 2 and 3 in mining wPFI in a fully uncertain dataset with corresponding minimum support values, minSup. It can be seen that mining wPFI takes significantly more time compared to partly uncertain databases, and the runtime of the algorithm will increase if minSup decreases, similar to partly uncertain databases. The probabilistic frequent threshold, t, is set to 0.8

However, while the difference in runtime between algorithms 2 and 3 in partly uncertain databases is clearly visible, the difference in runtime of the two algorithms in a fully uncertain database is very small.

6.3 Weaknesses

- Both algorithms perform well when searching for wPFI in partly uncertain databases and perform quite poorly in fully uncertain databases.
- Algorithms 2 and 3 do not have much difference in runtime when used in fully uncertain databases.
- The dynamic programming computation using the double type can produce incorrect results when the number is too small or too large.

Conclusion

In conclusion, the performance of algorithms 2 and 3 in mining weighted-PFIs (w-PFIs) is influenced by the minimum support value, minSup, threshold t, scale factor a, and the nature of the database. In both partly and fully uncertain databases, a decrease in minSup leads to an increase in execution time due to the discovery and verification of more w-PFI candidates. In most cases, algorithm 3 runs faster than algorithm 2.

However, the nature of the database also plays a significant role. Mining w-PFIs in fully uncertain databases takes significantly more time compared to partly uncertain databases. This is likely due to the increased complexity and uncertainty involved in fully uncertain databases.

Interestingly, while the difference in runtime between algorithms 2 and 3 is clearly visible in partly uncertain databases, this difference is minimal in fully uncertain databases. This suggests that the performance of these algorithms may converge under conditions of full uncertainty

References

[1] Zhiyang Li, Fengjuan Chen, Junfeng Wu, Zhaobin Liu, Weijiang Liu, "Efficient weighted probabilistic frequent itemset mining in uncertain databases," 2020.

[2] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, Andreas Zuefle, "Probabilistic Frequent Itemset Mining in Uncertain Databases," 2009.

APPENDIX

Pr() function

Input:

- DB: ArrayList of Maps representing the database.
- X: Set of items of type E.
- MinSup: Minimum support for frequent itemsets.

Output:

- Returns a double representing the probability of the given item set in the database.
- Behavior:
- Computes the probability of the given item set in the context of the database using dynamic programming.

Time Complexity:

- Basic operation: P[i][j] = P[i-1][j-1] * probabilities[j-1] + P[i][j-1] * (1 - probabilities[j-1]);
- Loop over transactions in DB: $O(|N|)$, where $|N|$ is the number of transactions.
- Pre-calculation loop: $O(|N|)$.
- Dynamic programming loop: $O(minSup * |N|)$.
- Overall: $O(minSup * |N| + |N|) = O(minSup * |N|)$.

Space Complexity:

- Storage of P, probabilities: $O(minSup * N)$.

Scan_Find_Size_1_wPFI

Input:

- I: Set of generic type E representing the item set.
- minSup: Minimum support for frequent itemsets.
- DB: ArrayList of Maps, where each map represents a transaction with items of type E and associated probabilities

Output:

- Returns a set of sets (Set<Set<E>>) representing frequent itemsets of size 1.

Behavior:

- Scans the the itemset I for frequent itemsets of size 1.
- Computes probabilities and weights.
- Filters itemsets based on minimum support (minSup) and a threshold (t).

Time Complexity:

- Bacsic operation: double[] pr_mu = Pr(this.DB, itemSet, this.minSup);
- Loop over I: $O(|I|)$, where $|I|$ is the number of unique items.
- Calculating calculateWeight, Pr: $O(minSup * N)$.
- Overall: $O(minSup * N * |I|)$.

Space Complexity:

- Storage of $L1$, mu_k, mu_1: $O(M)$, where M is the number of unique items.

Scan_Find_Size_k_wPFI

Inputs:

- Ck: Set of item sets (Set<IItemSet>) representing candidate itemsets of size k.
- DB: ArrayList of Itemset, where each map represents a transaction with items of type E and associated probabilities
- w: Map<Integer, Double> representing the weights for each item.
- MinSup: Minimum support for frequent itemsets.
- T: Threshold value.

Output:

- Returns a set of item sets (Set<IItemSet>) representing frequent itemsets of size k.

Behavior:

- Scans the database (DB) for frequent itemsets of size k using candidate itemsets (Ck).
- Computes probabilities and weights.
- Filters itemsets based on minimum support (minSup) and a threshold (t).

Time Complexity:

- Basic operation: Pr(DB, itemSet, minSup).
- Loop over Ck: $O(K)$, where K is the number of sets in Ck.
- Calculating calculateWeight, Pr: $O(minSup * N)$.
- Overall: $O(minSup * N * K)$, where K is the number of sets in Ck.

Space Complexity:

- Storage of Lk, mu_k: $O(K * M)$, where K is the number of sets in Ck.