10.18686/aitr.v2i1.3860

# The Research on the Improvement of FP-growth Algorithm

**Zixian Wu, Gang Fang**

**Chongqing Three Gorges University, Wanzhou, Chongqing, 404020**

***Abstract:*** This paper focuses on the issue of low efficiency in the FP-growth algorithm for frequent pattern mining and proposes an improved algorithm, ICFM-growth. Experimental results demonstrate that the improved algorithm outperforms the FP-growth algorithm in terms of both runtime and space utilization. By studying the classical frequent pattern algorithms Apriori and FP-growth, the latter is an improved algorithm based on Apriori to address the problems of generating a large number of candidate itemsets and consuming significant memory space. While FP-growth exhibits superior mining efficiency compared to Apriori, it faces challenges when dealing with large and long transactional databases due to the construction of numerous FP-trees, which increases computational tasks and prolongs runtime, leading to lagging mining efficiency. To address this issue, the improved algorithm ICFM-growth is proposed. It constructs a co-occurrence frequency matrix to perform preliminary screening on the transaction set, focusing on high-frequency and co-occurring item pairs, thereby reducing unnecessary computations. In the initial stage, as important item pairs have already been filtered, the algorithm can directly operate on these item pairs and items, rather than the entire dataset, thereby reducing the search space and computational complexity. Additionally, the structure of the FP-tree enables the algorithm to store and process data more efficiently, avoiding frequent scanning of the entire database, as seen in traditional Apriori algorithms. Finally, through simulation experiments on publicly available datasets such as Movie Data and House Data, validated using cross-validation, the ICFM-growth algorithm proves to be significantly superior to FP-growth in terms of time and space efficiency. It demonstrates faster runtime, lower memory consumption, and superior mining efficiency compared to FP-growth.

***Keywords:*** Frequency matrix; Frequent patterns; Data mining

## 1. Introduction

Based on the initially proposed Apriori algorithm, which is simple and easy to implement, challenges arise when dealing with dense databases. The need for multiple scans to generate numerous candidate itemsets leads to significant memory consumption. In response to this issue, the later-developed FP-growth algorithm addresses the shortcomings of Apriori by requiring only two scans. It represents transactions using a compressed tree, effectively reducing the need for frequent database scans. However, when dealing with long transactional databases, FP-growth still encounters challenges, as there are substantial potential candidate itemsets that occupy significant space and increase runtime. To address these shortcomings, this research focuses on improving the algorithm by reducing the number of database accesses, optimizing the generation of candidate itemsets, and enhancing support calculation methods to increase mining efficiency. Additionally, the study explores the optimization of data storage space by altering data structures and storage methods.

In response to the low efficiency issue, this paper proposes an improved algorithm, ICFM-growth, based on the FP-growth algorithm. ICFM-growth aims to address the challenges posed by dense and long transactional databases by refining the process of mining frequent patterns. The proposed improvements include reducing the frequency of database accesses, optimizing the generation of candidate itemsets, and refining the support calculation method. Furthermore, the study delves into changes in data structure and storage methods to optimize data storage space. Through these enhancements, ICFM-growth seeks to overcome the limitations of existing algorithms and provide a more efficient solution for frequent pattern mining in the context of diverse and challenging database scenarios.

## 2. Problem Statement

### 2.1 Association Rules

Association rules have significant applications in daily life, such as analyzing shopping baskets in supermarkets and exploring implicit associations in sales transactions, like the combination sales of beer and diapers. The expression format of association rules is represented as X→Y, where Support and Confidence are two crucial metrics for association rules.

### 2.2 Frequent Itemsets

In daily life, a collection of frequently occurring items is commonly referred to as a "frequent itemset." The essential metric for frequent itemsets is the support, which serves as the threshold for identifying such sets. Only those itemsets that meet the minimum support threshold are considered frequent itemsets. Transactions, items, itemsets, and support are the fundamental elements in frequent pattern mining.

Let I = {i1, i2, i3, i4, ..., im} be a set composed of m distinct items. Each item may represent a disease or a medication in a healthcare context. The sets formed by different combinations of items are referred to as itemsets, and a collection of k-itemsets is known as a k-itemset. D represents the database, where a transaction T in D is a collection of items from I, denoted as T ∈ D. In this context, T signifies the set of medications purchased by a patient in a single buying instance. The minimum support threshold is user-specified, and only itemsets that satisfy the minimum support (minsup) are recognized as frequent itemsets.

### 2.3 Frequent Pattern Growth Tree

The FP-tree is designed for discovering frequent patterns rather than generating candidate itemsets. It involves constructing the FP-tree tree and mining frequent patterns. The FP-tree operates by sequentially reading transactions and mapping them to paths in the FP-tree. Different transactions are placed on distinct paths, where many distinct items may appear on the same path. To optimize memory space, the items on the same path are compressed into a single path, and the count of duplicated nodes is incremented.

## 3. Algorithm Description of ICFM-growth

Based on the FP-growth algorithm, an improved algorithm named ICFM-growth is proposed with the objectives of reducing the mining search space, conserving memory, shortening runtime, improving the quality of frequent pattern mining, and enhancing mining efficiency. Firstly, an Item Co-occurrence Frequency Matrix is constructed, and matrix frequencies are calculated. Secondly, the correlation between items is computed, and a threshold is set for filtering, determining important item pairs. Lastly, important item pairs are selected, and frequent itemsets are extracted from the FP-tree, aiming to mine high-quality frequent patterns.

The specific process description of the ICFM-growth algorithm is as follows: Input: Minimum support threshold (minsup), transaction database D, matrix threshold Output: Full set of frequent patterns Procedure:

(1) Begin by constructing the item co-occurrence frequency matrix using the transaction dataset. Traverse the entire dataset, tally the occurrence frequency of each item, where the frequency represents the support count of the item. The item co-occurrence frequency matrix is used to describe the co-occurrence relationships between items, facilitating the analysis of the simultaneous occurrence frequency of items in the dataset. Definition: Assuming there are N items, a matrix M of size N×N can be constructed, where M[i][j] represents the number of times items i and j co-occur or their co-occurrence frequency. For example, consider a supermarket transaction dataset with multiple transaction records, each representing a shopping basket, and each item representing a product in the basket. Transactions: {A, B, C}, {A, C, D}, {B, C, E}, {A, B, C, E}. Based on this dataset, construct an item co-occurrence frequency matrix, where rows and columns represent different items, and matrix elements indicate the corresponding co-occurrence frequency of item pairs. In this example, a possible item co-occurrence frequency matrix may look like the following:

**Table 3.1 Item Co-occurrence Frequency Matrix**

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 3 | 2 | 3 | 1 | 2 |
| B | 2 | 3 | 3 | 0 | 2 |
| C | 3 | 3 | 4 | 1 | 2 |
| D | 1 | 0 | 1 | 1 | 0 |
| E | 2 | 2 | 2 | 0 | 2 |

In the dataset, the number of times items A and B co-occur is 2, and the number of times items C and D co-occur is 1, and so forth. Based on the item co-occurrence frequency matrix, preliminary filtering is performed. This involves setting a threshold to filter out item pairs with co-occurrence counts below the threshold, thereby reducing the scale of mining. The focus is on items with higher frequencies.

(2) Compute the matrix threshold. Set a threshold to determine which item pairs are important. This threshold is based on the average co-occurrence frequency of item pairs. If the value of (M_{ij}) is greater than or equal to this threshold, the item pair ((i, j)) is considered important.

Let Mij represent the co-occurrence frequency of items i and j. Set a threshold T, which is determined based on the average co-occurrence frequency of item pairs. Define the matrix's average value as the average co-occurrence frequency. The set of important item pairs I can be represented as:

$$\bar{M} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} C[i][j] \quad (3.1)$$

$$I = \{(i, j) \mid M_{ij} \geq T\} \quad (3.2)$$

In the equation, i and j are indices or identifiers of items;

$M_{ij}$ is the co-occurrence frequency of items i and j;

T is the set threshold;

$\bar{M}$ is the ave rage co-occurrence frequency of item pairs.

(3) Item Pair Filtering. Assuming I is the set of important item pairs composed of co-occurrence frequencies greater than or equal to the threshold T, introduce filter(I) as a filtering function used to select highly correlated item pairs. The filtering result F can then be expressed as:

$$F=filter(I) \quad (3.3)$$

In the equation, I is the set of important item pairs, i.e., I = {(i, j) | Mij ≥ T};

F is the set of highly correlated item pairs after filtering.

Specifically, filter(I) represents the process of filtering item pairs from I, retaining only highly correlated item pairs. This filtering process may involve evaluating co-occurrence frequencies and other metrics to determine which item pairs are considered highly correlated.

(4) Mine Frequent Itemsets. Based on the filtered items from the co-occurrence matrix, use the FP-Growth algorithm to extract frequent itemsets from the FP-tree. The FP-Growth algorithm constructs an FP-tree (Frequent Pattern Tree), a special tree structure, to store information about frequent itemsets. This process avoids the generation of numerous candidate itemsets, thus improving efficiency. The algorithm starts by sorting all items by their frequency and then transforms the dataset into an FP-tree. Subsequently, for each item (starting from the least frequent), its conditional pattern base is generated, which is a small FP-tree representing all paths containing that item. These conditional pattern bases are then used to generate more frequent itemsets. This process can be recursive until no new frequent itemsets are generated.

## 4. Performance Comparison of ICFM-growth Algorithm

The experiment was conducted on the PyCharm platform version 2023.2.1, utilizing an 11th Gen Intel(R) Core(TM) i5-1135G7 processor. Two public datasets, namely "house data" and "movie data," were employed for the simulation experiment.

Three algorithms, namely ICFM-growth, Apriori, and FP-growth, were evaluated on the two datasets under varying support threshold values. The correctness and scientific validity of the ICFM-growth algorithm were verified by setting different support threshold values. The runtime of each algorithm was recorded under different support threshold settings, with shorter times indicating higher mining efficiency.

Support thresholds were set at 0.20, 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, and 0.29. The experimental results are presented in Tables 3.8 and 3.9 (execution time unit: seconds). A comparative analysis was performed between the ICFM-growth algorithm and classical frequent pattern mining algorithms (Apriori and FP-growth). The comparison focused on mining efficiency, runtime, and the number of generated frequent itemsets. The correctness of the algorithms was tested by comparing the frequent itemsets mined under the same support threshold. Additionally, the efficiency and runtime of the three algorithms were compared under the same support threshold and across different datasets.

As shown in Table 3.8, the improved algorithm demonstrated higher time efficiency compared to classical algorithms Apriori and FP-growth. Apriori had the lowest runtime efficiency, with longer runtimes compared to the improved algorithm and FP-growth on the same dataset. ICFM-growth had the shortest runtime and highest efficiency, consistent across different support threshold settings.

**Table 3.8: Comparison of Execution Times for Three Algorithms on the Movie Data Dataset**

| Support | 0.20 | 0.21 | 0.22 | 0.23 | 0.24 |
|---|---|---|---|---|---|
| ICFM-growth | 0.077 | 0.066 | 0.050 | 0.040 | 0.033 |
| FP-growth | 0.122 | 0.094 | 0.083 | 0.071 | 0.058 |
| Apriori | 4.820 | 4.670 | 4.671 | 4.687 | 4.711 |
| Support | 0.25 | 0.26 | 0.27 | 0.28 | 0.29 |
| ICFM-growth | 0.028 | 0.025 | 0.024 | 0.019 | 0.019 |
| FP-growth | 0.053 | 0.050 | 0.047 | 0.047 | 0.040 |
| Apriori | 4.665 | 4.659 | 4.739 | 5.015 | 4.724 |

Transforming the experimental data from the table into line charts, due to the different units of measurement for execution times between the Apriori algorithm and the ICFM-growth and FP-growth algorithms, the performance of the ICFM-growth algorithm and the FP-growth algorithm on the same line chart is not distinguishable and nearly overlaps. Therefore, separate images are used to represent them, as shown in Figures 3.3 and 3.4.

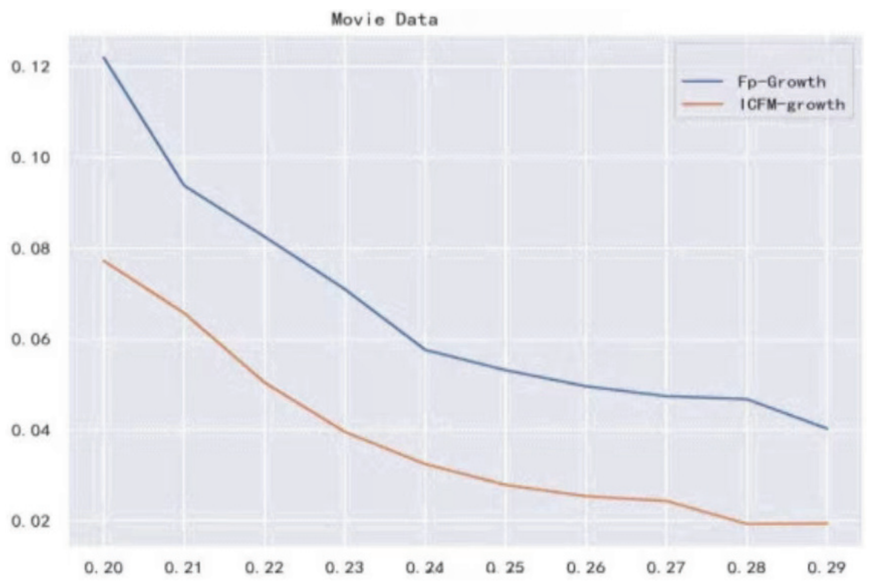

**Figure 3.3: Comparison of Execution Times for Two Algorithms on the Movie Data Dataset**

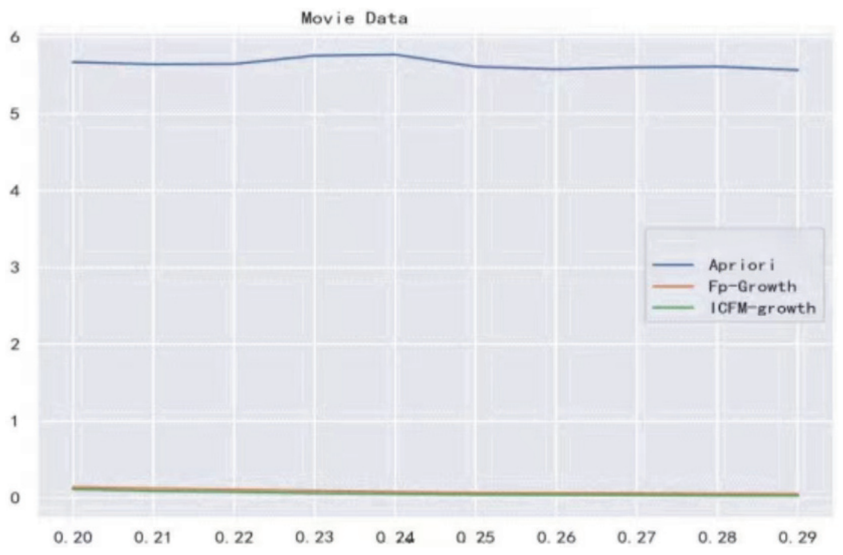

**Figure 3.4: Comparison of Execution Times for Three Algorithms on the Movie Data Dataset**

From Figure 3.3, it can be observed that at a support threshold of 0.20, the runtime of the ICFM algorithm is 0.077, while the runtime of the FP-growth algorithm is 0.122. The ICFM-growth algorithm outperforms the original algorithm significantly. As the support threshold increases, the runtime of both algorithms gradually decreases. However, the runtime of the ICFM-growth algorithm is significantly less than that of the original algorithm. This indicates that the mining efficiency of the ICFM-growth algorithm is notably higher than the classical FP-growth algorithm, demonstrating a significant improvement in the effectiveness of the improved algorithm.

**Table 3.9: Comparison of Execution Times for Three Algorithms on the House Data Dataset**

| SUPPORT | 0.20 | 0.21 | 0.22 | 0.23 | 0.24 |
|---|---|---|---|---|---|
| ICFM-growth | 0.063 | 0.048 | 0.044 | 0.040 | 0.031 |
| FP-growth | 0.118 | 0.099 | 0.085 | 0.071 | 0.060 |
| Apriori | 3.661 | 3.436 | 3.431 | 3.450 | 3.560 |
| Support | 0.25 | 0.26 | 0.27 | 0.28 | 0.29 |
| ICFM-growth | 0.029 | 0.027 | 0.025 | 0.024 | 0.021 |
| FP-growth | 0.058 | 0.056 | 0.054 | 0.051 | 0.044 |
| Apriori | 3.499 | 3.522 | 3.501 | 3.476 | 3.492 |

Transforming the experimental data from the table into line charts, due to the different units of measurement for execution times between the Apriori algorithm and the ICFM-growth and FP-growth algorithms, the performance of the ICFM-growth algorithm and the FP-growth algorithm on the same line chart is not distinguishable and nearly overlaps. Therefore, separate images are used to represent them, as shown in Figures 3.5 and 3.6.

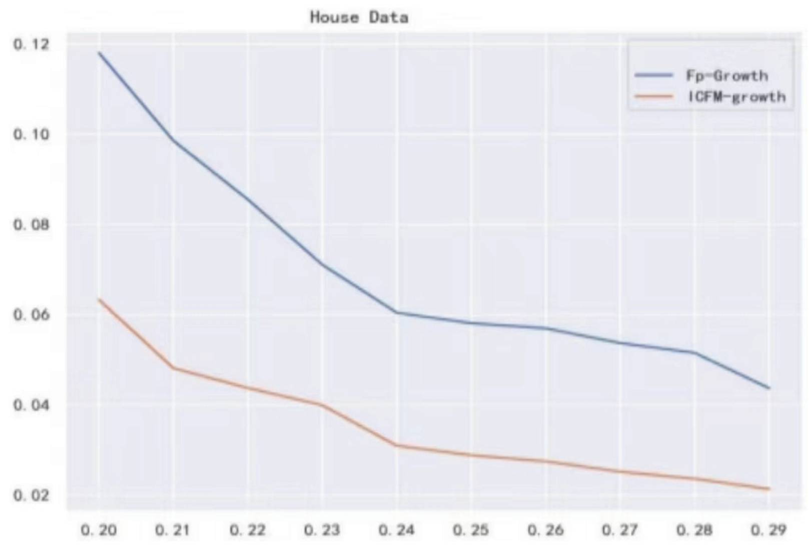

**Figure 3.5: Comparison of Execution Times for Two Algorithms on the House Data Dataset**

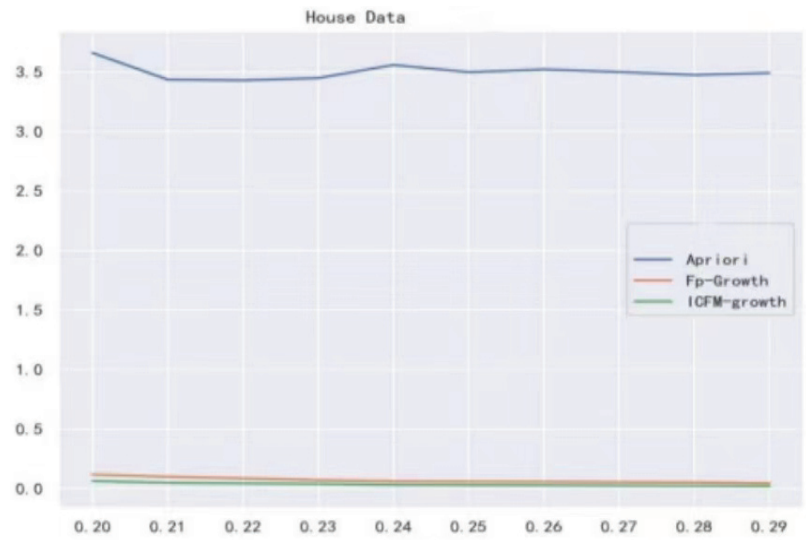

**Figure 3.6: Comparison of Execution Times for Three Algorithms on the House Data Dataset**

This dataset has fewer instances than the Movie Data dataset, but it has more feature attributes, resulting in relatively shorter runtimes. However, the ICFM-growth algorithm still exhibits the shortest runtime. The combination of co-occurrence matrix and FP-Growth algorithm significantly reduces unnecessary computations by focusing on high-frequency co-occurring item pairs. In the FP-Growth phase, since important item pairs have already been filtered, the algorithm can directly operate on these pairs rather than the entire dataset, thereby reducing search space and computational complexity. Additionally, the structure of the FP-tree enables the algorithm to more efficiently store and process data, avoiding the frequent scanning of the entire database in traditional Apriori algorithms.

## 5. Conclusion

This algorithm simplifies the computation of the FP-tree by focusing on high-frequency co-occurring item pairs. It optimizes the initial steps of the algorithm, avoiding the generation of numerous candidate itemsets, reducing unnecessary calculations in the FP-growth algorithm, enhancing the algorithm's mining capabilities, and improving the efficiency of frequent pattern mining. It is particularly suitable for dense and large-scale datasets.

## References

[1] Hu Shichang. Research and Improvement of Apriori Algorithm [D]. Qingdao University, 2019.

[2] Qu Rui, Zhang Tianjiao. Improved Apriori Algorithm Based on Matrix Compression[J]. Computer Engineering and Design, 2017, (8): 2127-2131.

[3] Wei Kun, Wang Fang, Huang Shucheng. Improved Frequent Pattern Mining Algorithm[J]. Computer and Digital Engineering, 2021, 49(11): 2175-2179.

[4] Malarvizhi SP, Sathiyabhama B. Frequent Pagesets from Web Log by Enhanced Weighted Association Rule Mining[J]. Cluster Computing, 2016, 19(1): 269-277.

[5] Wang Meng, Zou Shurong, Fang Rui. An Improved Apriori Algorithm Based on Matrix[J]. Information Technology, 2018, (3): 150-154, 158.

[6] Wei Kun. Research and Application of MGFP-growth Improved Algorithm Based on FP-growth Association Rules [D]. Jiangsu University of Science and Technology, 2020. DOI:10.27171/d.cnki.ghdcc.2020.000636