Article

Efficient Top-K Identical Frequent Itemsets Mining without Support Threshold Parameter from Transactional Datasets Produced by IoT-Based Smart Shopping Carts

Saif Ur Rehman 1, Noha Alnazzawi 2, Jawad Ashraf 3, Javed Iqbal 1 and Shafiullah Khan 4,*

1 Department of Computer Science, University of Peshawar, Peshawar 25120, Pakistan
2 Department of Computer Science and Engineering, Yanbu Industrial College, Royal Commission for Jubail and Yanbu, Yanbu Industrial City 41912, Saudi Arabia
3 Institute of Computing, Kohat University of Science and Technology, Kohat 26000, Pakistan
4 Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huai’an 233003, China
* Correspondence: skkust@gmail.com

Abstract: Internet of Things (IoT)-backed smart shopping carts are generating an extensive amount of data in shopping markets around the world. This data can be cleaned and utilized for setting business goals and strategies. Artificial intelligence (AI) methods are used to efficiently extract meaningful patterns or insights from such huge amounts of data or big data. One such technique is Association Rule Mining (ARM) which is used to extract strategic information from the data. The crucial step in ARM is Frequent Itemsets Mining (FIM) followed by association rule generation. The FIM process starts by tuning the support threshold parameter from the user to produce the number of required frequent patterns. To perform the FIM process, the user applies hit and trial methods to rerun the aforesaid routine in order to receive the required number of patterns. The research community has shifted its focus towards the development of top-K most frequent patterns not using the support threshold parameter tuned by the user. Top-K most frequent patterns mining is considered a harder task than user-tuned support-threshold-based FIM. One of the reasons why top-K most frequent patterns mining techniques are computationally intensive is the fact that they produce a large number of candidate itemsets. These methods also do not use any explicit pruning mechanism apart from the internally auto-maintained support threshold parameter. Therefore, we propose an efficient TKIFIs Miner algorithm that uses depth-first search strategy for top-K identical frequent patterns mining. The TKIFIs Miner uses specialized one- and two-itemsets-based pruning techniques for topmost patterns mining. Comparative analysis is performed on special benchmark datasets, for example, Retail with 16,469 items, T40I10D100K and T10I4D100K with 1000 items each, etc. The evaluation results have proven that the TKIFIs Miner is at the top of the line, compared to recently available topmost patterns mining methods not using the support threshold parameter.

Keywords: IoT; sensors; frequent itemsets mining; artificial intelligence; data mining

1. Introduction

The IoT is a comprehensive and much researched direction nowadays, and it can be categorized into two main components: embedded systems using sensors and processing technologies on Wi-Fi sensor networks. IoT is evolving day by day, yet it suffers from three main challenges: architecture [1], interoperability [2], and security [3–5]. Various IoT-enabled heterogeneous devices around us communicate through sensors via the internet to provide smart services. For example, smart health monitoring systems [6], smart parking systems [7], and smart shopping carts [8]. In this paper, we address one of the pertinent issues with smart shopping carts’ transactional data collected on web servers. After preprocessing this coarse-grained data, some artificial intelligence method can be
utilized for effective analysis, such as to study customer buying patterns, effective catalog design, efficient stock management, etc. [9–11]. Frequent itemsets mining is one such method in association rule mining that can serve the aforementioned purpose perfectly. Frequent Itemsets Mining (FIM) is the discovery of those itemsets which coexist in the given dataset more than the user specified limit [12]. The importance of the FIM process can be seen from the fact that it is used for finding customer purchase trends [13], fraud detection in the banking sector [14], most frequent web pages traversal findings [15], health monitoring [16], data profiling systems [17–19], and document clustering [20].

Rakesh Agarwal first highlighted FIM in his seminal work as the “Apriori algorithm”. Subsequently, the research community has floated many proposals for the FIM problem. The proposed solutions for FIM encountered two main issues before and after the discovery of Frequent Itemsets (FIs) [21]. First, the user’s selection of the ideal Support Threshold Parameter (STP) before starting the FIM process. Second, the exponential number of FIs generated as a result of the FIM process. This issue is addressed well in different compact FIs-mining procedures that have been proposed. Hence, the selection of an optimum STP for the FIM process is a harder task as it depends on the characteristics of the datasets. Also, the frequentness of an itemset depends on the STP value that is provided by the user. This value is used by the FIM process to isolate frequent patterns from infrequent patterns. If the user chooses a high threshold, there will be fewer itemsets generated, and choosing a low threshold will generate more patterns. Therefore, the user may tune the FIM process with different STP values to discover the desired FIs. The miss tuning of STP can in turn result in two main problems for the FIM process, i.e., effects on the size of the result space and the size of the search space [21].

Many algorithms have been proposed in the area of mining the topmost frequent itemsets [22–31]. These algorithms are commonly named K-topmost or topmost FIM approaches. Initial topmost FIM algorithms are derivative from the classical user’s-STP-based FIM techniques and inherit all of their demerits, i.e., the Apriori or FP-tree algorithm. There are two main challenges faced by the topmost FIM algorithms. First, the Apriori-based topmost FIM techniques proposed in [23,24,31] have multiple scans, excessive candidate generation, they are search space focused, and have large search space. Second, the FP-tree-based topmost FIM techniques proposed in [22,25,27,28,32] perform at least two scans of the database, suffer from the large search space problem in the case of dense datasets, and have the memory resident construction of FP-tree. In general, the topmost FIs-mining techniques designed based on the aforementioned principles are computationally inefficient [30]. Recent techniques for topmost patterns mining include the mining top-K frequent patterns without support threshold proposed by Salam et al. [30], the top-K identical frequent without support threshold by Rehman et al. [29], and the top-K frequent itemsets mining technique based on equivalence classes by Iqbal et al. [26]. All Apriori-and-FP-growth independent top-K methods mentioned earlier are not viable to perform due to the two main limitations described below.

**a: No top-K specific pruning method:**—All top-K specific methods so far do not use any top-K candidate itemsets pruning technique apart from an automatic support threshold raising strategy. The aforementioned strategy can only determine whether an itemset is frequent or not after computing its support value. On the other hand, we need a pruning mechanism which guarantees that a particular itemset is infrequent without computing its support value.

**b: Computationally intensive methods:**—The Apriori-based top-K methods, the FP-Growth based top-K methods, and the Apriori-and-FP-Growth-free methods are computationally intensive on dense datasets with large value of K. Apriori-and-FP-Growth-free methods produce a large number of candidate itemsets, requiring support computation for every itemset. FP-Growth-based top-K methods are computationally extensive due to the FP-Growth method.
1.1. Research Objectives

In this article, we propose a novel topmost frequent itemsets mining algorithm named TKIFIs Miner. The proposed algorithm is not a derivative of the Apriori or FP-Growth techniques, but it uses a recursive best-first-based depth-first traversal technique on search space to discover topmost frequent itemsets. During this traversal, two novel pruning strategies are used to early prune the portion of the search space not required for topmost patterns mining. Subsequently, support of the remaining candidate patterns using an intersection of TIDs approach is calculated. Finally, the support of the pattern is checked against the least support of the patterns available in the top-K list. If the support of the discovered pattern is greater than the least support of the patterns available in the top-K list, it is added to the top-K list; otherwise, the pattern is not added to the top-K list.

1.2. Organization of the Paper

The article is organized as follows: Section 2 presents previous work related to this area of research, followed by a discussion of the proposed algorithm and a presentation of a few preliminaries about the topmost frequent itemsets mining problem in Section 3. Additionally, a set of examples are presented to highlight the topmost frequent itemsets mining using k as parameter in place of support threshold. In Section 4, the experimental evaluation of the proposed algorithm and the comparison of the TKIFIs Miner with the current benchmark methods are presented. Finally, Section 5 discusses the achievements of this work and future directions that can be unveiled through topmost frequent patterns mining studies.

2. Related Work

The top-K FIM is introduced to avoid user-given threshold parameter tuning. On the other hand, top-K FIM is considered a computationally harder task to perform as compared to support-based FIM. Due to the self-explanatory results, it is applied in different real world application domains such as monitoring users’ activity from their movements data [33], COVID-19 virus strains classification and identification [34], and extraction of metadata from dynamic scenarios [35,36].

Salam et al. for the first time suggested considering natural requirements of STP-free K-most FIs to design a novel topmost-FIM technique that was not derived from the aforementioned classical approaches [30]. This technique mines topmost maximal frequent itemsets and top-K maximal frequent itemsets from the given transactional dataset. According to this approach, a topmost maximal frequent itemset in a dataset is a frequent itemset of a highest support with item length greater or equal to 3. On the other hand, a top-K maximal frequent itemset set is a set of K distinct highest support maximal frequent itemsets. In this approach, first the association ratio of all 2-length itemsets is calculated, followed by the construction of an Association Ratio graph (AR-graph). Subsequently, the AR-graph is applied to construct an all-path-source-destination tree (ASD-tree). Finally, the ASD-tree is used to find the top-K maximal frequent itemset set. Additionally, the Top-K Miner introduced by Rehman et al. [29] does not extend any classic STP-based mining techniques. It performs only one scan of the dataset to construct frequent itemsets of length 2, and during this scan the top-K list is updated with frequent itemsets of lengths 1 and 2. The topmost frequent itemsets of length 2 from the top-K list whose support is greater or equal to the Kth support value in the top-K list are selected for topmost k frequent itemsets mining of lengths greater than or equal to 3. Subsequently, every 2-itemset is iteratively combined with the selected nodes of a Candidate Itemsets Search tree (CIS-tree) to form the itemsets of any length greater than 2. Their support is calculated and adjusted accordingly in the top-K list. The Top-K Miner finds all topmost FIs as per the tuned parameter, but suffers as memory-expensive due to the Candidate Itemsets Search tree (CIS-tree). Moreover, Iqbal et al. presented the TKFIM algorithm which inherits the Apriori mining technique for the discovery of K-topmost FIs [26]. This algorithm performs excessive candidate generation because it uses common itemsets prefixes of the already
produced topmost frequent itemsets. The frequencies of the itemsets are generated using the diffset support finding method [37]. Apart from the automatic support threshold raising strategy, the algorithm does not adopt any other pruning strategy to perform search space pruning and to avoid excessive candidate generation. Table 1 presents the discussed topmost FIM techniques and their details.

Table 1. Topmost FIM algorithms and their details.

| No | Algorithm Name | Pruning Strategy | Candidate Generation | Technique’s Type | All Patterns |
|----|----------------|------------------|----------------------|-----------------|-------------|
| 1  | Itemset-Loop [24]  
MTK [23]  
Top-N [31] | Auto STP raising | Yes | Apriori | All patterns |
| 2  | BOMO [32]  
COFI [27]  
CRMN [22]  
ExMiner [28]  
TFP [25] | Auto STP raising | No | FP-Growth | All patterns |
| 3  | Slam [30] | No STP | No | Graph-based approach | Few patterns |
| 4  | Top-K Miner [29] | Auto STP raising | Yes | No | All patterns |
| 5  | TKFIM [26] | Auto STP raising | Yes | Apriori | All patterns |

3. Frequent Itemsets Mining

In this section, we will discuss the proposed frequent itemsets mining algorithms, which include three methods: the TKFIs mining method that mines a user-required number of IFIs from the user-given dataset; the Gen-IFIs method that explores the search space to discover TK-IFIs; and the Candidate-IFIs method that generates the set of candidate itemsets for the newly created IFIs. Additionally, some examples will be presented while explaining the frequent itemsets mining methods to give more clarification to the reader. At the beginning, it is worth starting by presenting some preliminaries and definitions.

3.1. Preliminaries and Problem Definitions

The support threshold value provided by the user serves as the border between frequent and infrequent patterns. Our proposed FIM method is not based on a support threshold value supplied by the user. Then, how can our proposed method discriminate between frequent and infrequent patterns to produce the required number of patterns?

There is an important observation related to the selection of a support threshold as far as our proposed method is concerned. The parameter required by the proposed method is K, which determines the number of topmost frequent patterns that are required to be discovered from search space. The value of the K parameter guides the proposed method for the automatic adjustment of the border between the number of required topmost frequent and infrequent patterns. The support count value of the patterns decides the aforementioned border. This is automatically determined or adjusted by the machine based on K value rather than given by the user. Tuning the value of the K for finding the required number of patterns is simple and unlike a support threshold value which is not dependent on the characteristics of the given dataset. Therefore, based on the above discussion, the following definitions are presented for further groundwork:

1. Identical Itemsets (IIs): One or more itemsets are called IIs if and only if all of them have the same support count that is \( I_{\text{IIs}} = \{ p_i | \forall p_i \in X \land \text{supp}(p_i) = s \} \), where \( s \in S \) is the support count of every pattern \( p_i \) found in the set of support count \( S \).
2. Top-1 Identical Frequent Itemsets (IFI\(_1\)): One or more itemsets are called IFI\(_1\) if and only if all of them have same support count and their support is the first highest most
support in the set of support count S that is $I_{IFI_1} = \{ p_i | \forall p_i \in X \land \text{supp}(p_i) = s_1 \}$, where $s_1 \in S$ is the first highest support count in the set of support count S.

3. Top-K Identical Frequent Itemsets ($I_{IFI_K}$): One or more itemsets are called IFIs if and only if all of them have same support count and their support is the Kth highest most support in the set of support count S that is $I_{IFI_K} = \{ p_i | \forall p_i \in X \land \text{supp}(p_i) = s_K \}$, where $s_K \in S$ is the Kth highest support count in the set of support count S.

Suppose I is the set of all items in a database, i.e., $I = \{ i_1, i_2, i_3, \ldots, \ldots, i_n \}$, where $i$ is an item. The database $D = \{ t_1, t_2, t_3, \ldots, \ldots, t_m \}$ is a set of m transactions where every $t$ is a transaction. An itemset $p \subseteq I$ containing one or more transactions presents support of $p$, i.e., $\text{Supp}(p)$. Let $X$ be a set of all $p_x$ found in transactions of database D. That is, $X$ contains all the itemsets that can be generated from D. Let $S$ be the set of support counts of all the itemsets contained by $X$. The problem of topmost frequent itemsets mining is to find all IFIs of highest support to Kth support where K is a user specified number of topmost IFIs, i.e., $T_{IFI_K} = \{ I_{IFI_1}, I_{IFI_2}, I_{IFI_3}, \ldots, I_{IFI_K} \}$.

3.2. $T_{IFI_K}$ Mining Method

In this section we introduce $T_{IFI_K}$ Miner method that mines the user required number of IFIs from the user-given dataset D based on one simple parameter K as shown in step 1 in the above Algorithm 1. The database is scanned once to copy the tid in the corresponding position of the 1-itemset indexed with item $i$. It is done by performing the horizontal scan of the given datasets as shown in step 3 to 5 of the above Algorithm 1. The next step is copying all 1-itemsets of the highest support to the Kth support into set of candidate itemset $C$, and into the $T_{IFI_K}$ list. In order to perform the aforesaid task, first topmost highest support itemset is discovered and copied into itemset $i$, followed by copying this itemset in $C$ as shown in step 6a and step 6b of the above Algorithm 1. The topmost selected itemset $i$ in step 6a is copied into the $T_{IFI_K}$ list in step 6c of the above Algorithm 1. This itemset $i$ is removed from the list of 1-itemsets in step 6d, as shown above in Algorithm 1. In step 7 of the above Algorithm 1, we used 1-itemsets of highest frequency to Kth frequency for producing frequencies of 2-itemsets. In step 8, we use the $\text{Gen}_{IFI}$ algorithm for the top-K IFIs mining from the already trimmed search space. For more explanation, we give Example 1 to illustrate the algorithm by applying it to a transactional dataset that is given in Table 2.

Table 2. Transactional dataset.

| TID | Transactions |
|-----|--------------|
| 1   | A B D F      |
| 2   | A B F        |
| 3   | A D F        |
| 4   | B C D E F    |
| 5   | B C D E F    |
| 6   | A B F        |
| 7   | A B D F      |
| 8   | A B F        |
| 9   | A B C D F    |
| 10  | A B C E F    |

Example 1: Consider the transactional dataset given in Table 2. The dataset consists of 10 transactions and 6 distinct items. For the given value of $K = 4, 5$, the dataset is mined for top-4, top-5 IFIs as shown in Table 3.
Table 3. Top-4 IFIs, Top-5 IFIs.

| Top-4 IFIs       | Top-5 IFIs       |
|------------------|------------------|
| (1) F = 10       | (1) F = 10       |
| (2) B = 9, BF = 9| (2) B = 9, BF = 9|
| (3) A = 8, AF = 8| (3) A = 8, AF = 8|
| (4) AB = 7, FBA = 7 | (4) AB = 7, FBA = 7 |
| (5) D = 6, DF = 6|                  |

Algorithm 1. TK$_{IFIs}$ Miner (Data Set: D, User Input K).

Candidate Itemset : C
Set of 1 − Itemsets : 1_itemsets, Set of 2 − Itemsets : 2_itemsets

transaction : tr

Top − K IFIs List Structure : TK$_{IFIs}$

Steps
(1) input K, D // input the number of TK$_{IFIs}$ to mined from dataset D.
(2) TK$_{IFIs}$ = ∅, C = ∅, IFI = ∅;
(3) foreach transaction tr ∈ D
(4) foreach item i ∈ tr
(5) 1_itemsets$_i$ = 1_itemsets$_j$. tid ∪ tr(i).tid;
(6) Repeat Until |TK$_{IFIs}$| = K // copy top − K 1 − itemset into TopK$_{IFIs}$ list structure
  a. i = max_SupportItem(1_itemsets);
  b. C = C ∪ 1_itemsets$_i$;
  c. Append (TK$_{IFIs}$, 1_itemsets$_i$);
(7) foreach i : 1 to |C| − 1 // 2-itemsets’s support used for pruning purpose in Gen$_{IFIs}$
  a. sprt = support(C$_i$, C$_{i+1}$);
  b. 2_itemsets[C$_i$, C$_{i+1}$] = sprt;
(8) Gen$_{IFIs}$(IFI = ∅, C, TK$_{IFIs}$, K, 2_itemsets);
(9) return TK$_{IFIs}$;

Table 3 represents the topmost patterns mined for K= 4, and 5, from the transactional dataset given in Table 2. It is clear from Table 3 that the only input required from the user is the K value to produce the desired number of topmost patterns. In the top-K result sets in Table 3, it can be observed that all the subsets of an FI in top-K IFIs are placed at the same, or ranked at higher, positions due to Apriori property [1]. For example, all the subsets of a frequent itemset {FBA = 7} in top-5 IFIs in Table 3 are ranked at top-4 or higher positions.

3.3. TK$_{IFIs}$ Production Method

This section presents the Gen$_{IFIs}$ method which is a search space exploration algorithm that discovers the TK$_{IFIs}$ based on the number of topmost-K IFIs required by the user. It is a recursive best-first-based depth-first search TK$_{IFIs}$ discovery approach. Every time Gen$_{IFIs}$ is called recursively, it discovers one or more IFIs. The generated IFIs are then either accommodated in the currently maintained TK$_{IFIs}$ list or dropped due to lesser support count. The Gen$_{IFIs}$ procedure uses four parameters during the TK$_{IFIs}$ discovery process. The first parameter is current highest support IFI$_o$. At the start, when Gen$_{IFIs}$ is called, it is passed as empty. The second parameter is candidate itemsets list C$_0$ which is used for the extension of current IFI$_o$. Before calling Gen$_{IFIs}$ for the first time, the top-K 1-itemsets are copied in C$_0$, and subsequently it is passed to Gen$_{IFIs}$ as the second parameter. The parameter TK$_{IFIs}$ maintains the current top-K IFIs produced before calling and within the Gen$_{IFIs}$ calls. The 2 − itemsets is a last parameter used in Gen$_{IFIs}$ calling. This parameter is used in pruning the candidate patterns in the Candidate_IFIs procedure in step 5 of Algorithm 2.
The GenIFIs procedure extends the current IFI with the item i from the candidate itemset list \( C_o \) iteratively as shown in step 1 of Algorithm 2. The minimum support threshold is raised automatically with the addition of IFIs in the TK_{IFIs} list. Therefore, before appending item i from the candidate list to the current IFI, its support count is checked with minimum support of the TK_{IFIs} list, as shown in step 2 of Algorithm 2. If the support of item i is less than the minimum support of IFIs in the TK_{IFIs} list, then the current item i is pruned from combining it with the current IFI. In step 3, the selected item i is combined with current IFI \( IFI_n \) to form a new IFI, IFI_{n+1}. In step 4, all the 1-itemsets in \( C_o \) whose support value is greater than or equal to the current item i are selected/copied in the possible candidate itemset list \( P_c \).

The Candidate_IFIs procedure is used here to return a new candidate list \( C_n \) for the new IFI, IFI_{n+1}, as shown in step 5 of Algorithm 2. It uses the new IFI, IFI_{n+1}, formed in step 3, and possible candidate list \( P_c \), to generate a new candidate itemsets list, \( C_n \). In step 6, the GenIFIs algorithm is called recursively for further processing. The used search space strategy is explained below. For FIM, the search space is arranged in the form of a Set Enumeration tree (SE-tree), as presented in Figure 1. Since FIM methods discover patterns or itemsets in a search space, the SE-tree is one of the search space presentation methods used here to enumerate all of the possible subsets. The intuitions behind adopting the SE-tree for search space presentation is given below.

1. The problems where search is a subset of power set. It represents irredudant complete search space.
2. The SE tree is a structure that can be traversed recursively.
3. For the FIM techniques adopting depth-first traversal of the search space, SE-tree can be used to model their search space efficiently.

![Set enumeration tree for 5 items.](image-url)
Algorithm 2. \texttt{GenIFI}(IFI\textsubscript{o}, C\textsubscript{o}, TK\textsubscript{IFIs}, 2 \text{- itemsets})

\textbf{Input}
IFI\textsubscript{o}: Old IFI (current node head)
C\textsubscript{o}: Old candidate itemset list (current node tail)
IFI\textsubscript{n}: New IFI (new node’s head portion)
C\textsubscript{n}: New candidate itemset list (new node’s tail portion)
TK\textsubscript{IFIs}: Current list of Top-K IFIs discovered so far
2-itemsets: set of 2-items formed from \( K \) highest support 1-itemset for pruning purpose
P\textsubscript{c}: Possible candidate itemset list

\begin{enumerate}
\item \texttt{foreach item} \( i \in C\textsubscript{o} \) \texttt{// select item} \( i \) \texttt{with highest support count from} \( C\textsubscript{o} \)
\item \texttt{if} (sprt\((i) < \text{min}_{\text{sprt}}(TK\textsubscript{IFIs})) \texttt{continue} \texttt{// prune selected item from the list}
\item \( \text{IFI}\textsubscript{n} = \text{IFI}\textsubscript{o} \cup \{i\} \);
\item \( P\textsubscript{c} = \{ \forall x: x \in C\textsubscript{o} \text{ and support}(x) \leq \text{support}(i) \} \);
\item \( C\textsubscript{n} = \text{Candidate}\_\text{IFIs}(\text{IFI}\textsubscript{n}, P\textsubscript{c}, TK\textsubscript{IFIs}, 2 \text{- itemsets})\);
\item \texttt{GenIFI}(IFI\textsubscript{n}, C\textsubscript{n}, TK\textsubscript{IFIs}, 2 \text{- itemsets});
\item \texttt{Return} TK\textsubscript{IFIs};
\end{enumerate}

We now give Example 2 to illustrate the \texttt{GenIFI} method by applying it to the transactional dataset that was given in Table 2.

\textbf{Example 2:} Consider the transactional dataset Table 2. Generate all 1-itemset for \( K = 5 \) and corresponding 2-itemsets from it. The given transactions dataset is scanned transaction by transaction using iterative step 3 of the Algorithm 1. Every transaction is scanned from left to right for every item. The transaction ID for every item is recorded. Finally, the \( K \) highest support 1-itemsets are selected iteratively as shown in steps 6a and 6c of Algorithm 1. The 2-itemsets are then computed from the \( K \) highest support 1-itemsets, as shown in the table below.

3.4. Candidate Generation Method

The method is used to generate the set of candidate itemset \( C \) for the newly created \( \text{IFI}\textsubscript{n} \). Every item from this set will be used subsequently to extend the current \( \text{IFI}\textsubscript{n} \). For the creation of candidate itemset \( C \), individual items from possible candidate list \( P\textsubscript{c} \) are utilized. An item \( y \) is iteratively selected from \( P\textsubscript{c} \) in step 1, as shown above in Algorithm 3, for the purpose of addition into candidate list \( C \). Before appending an item \( y \) into candidate list \( C \), two types of pruning strategies are applied on it. These strategies are explained in the following.

\textbf{Itemset Based Candidate Pruning (IBCP):} An itemset \( \text{IFI}\textsubscript{n} \cup y \) is not a candidate for IFI if an item in an itemset has support count less than minimum support of IFIs in TK\textsubscript{IFIs}.

\textbf{Two-Itemset Based Candidate Pruning (TIBCP):} An itemset \( \text{IFI}\textsubscript{n} \cup y \) is not a candidate for IFI if any 2-itemset in an itemset has support count less than minimum support of IFIs in TK\textsubscript{IFIs}.

Let \( X \subseteq \text{IFI}\textsubscript{n} \cup y \), where \( X \) is a two-itemset and \( y \in X \) is a one-itemset to extend \( \text{IFI}\textsubscript{n} \). It is clear that the support count of \( y \) or \( X \) is less than minimum support of IFIs in TK\textsubscript{IFIs}, i.e., \( \text{min}_{\text{sprt}}(\text{TK}_{\text{IFIs}}) \). This implies that the \( \text{sprt}(\text{IFI}\textsubscript{n} \cup y) < \text{min}_{\text{sprt}}(\text{TK}_{\text{IFIs}}) \) according to the Apriori property [12]. Hence, an itemset \( \text{IFI}\textsubscript{n} \cup y \) can never be termed as IFI.

The IBCP and TIBCP strategies are applied on line numbers 2 and 4, respectively, in Algorithm 3. If the support of an item \( y \), or two-itemset containing an item \( y \), is less than minimum support of IFIs in TK\textsubscript{IFIs}, the item \( y \) is skipped from frequency test.
Algorithm 3. Candidate_IFIs(IFI_n, P_c, TK_IFIs, 2 − itemsets)

Input
IFI_n : New IFI (new node head portion)
P_c : Possible candidate
TK_IFIs : Current list of Top-K IFIs discovered so far
2-itemsets : set of 2-items formed from K highest support 1-items for pruning purpose

Steps
(1) Foreach item y ∈ P_c
(2) if (sprt(y) < min_sprt(TK_IFIs))
(3) continue; // prune selected item from the list
(4) if (2 − itemset[IFI_n||IFI_n||y] < min_sprt(TK_IFIs) ∨ // last item of IFI with y
(5) 2 − itemset[IFI_n||IFI_n||1], y] < min_sprt(TK_IFIs) ∨
(6) ......................
(7) 2 − itemset[IFI_n||IFI_n||2], y] < min_sprt(TK_IFIs) ∨
(8) 2 − itemset[IFI_n||IFI_n||1], y] < min_sprt(TK_IFIs))
(9) continue; // prune selected item from the list
(10) if (sprt(IFn_∪y) ≥ min_sprt(TK_IFIs))
(11) Append (TK_IFIs, IFn_∪ y, sprt(IFn_∪ y));
(12) C = C ∪ y;
(13) Return C;

The support count of the IFI_n ∪ y is computed in step 10 of Algorithm 3 as shown above. If the support count of the IFI_n ∪ y is found equal or greater than the minimum support of any IFI in a set of IFIs, i.e., TK_IFIs, the resultant IFI is added to TK_IFIs in step 11 of Algorithm 3 above. The currently selected item y is added to the set of candidates C in step 12 of Algorithm 3 above for further depth-first order processing of itemsets. We now give Example 1 to illustrate the Candidate_IFIs method by applying it to the transactional dataset that was given in Table 2.

Example 3: For the given transactional dataset in Table 2, assuming descending support order < on top-5 IFIs as shown in Table 4, the complete set enumeration tree is given above in Figure 1.
Table 4. Represents 1-itemsets, Top-K 1-itemsets and 2-itemsets.

| Sno | 1-Itemsets = Support |
|-----|----------------------|
| 1   | A = 8                |
| 2   | B = 9                |
| 3   | C = 4                |
| 4   | D = 6                |
| 5   | E = 3                |
| 6   | F = 10               |

| Sno | Top-K 1-Itemsets = Support |
|-----|-----------------------------|
| 1   | F = 10                      |
| 2   | B = 9                       |
| 3   | A = 8                       |
| 4   | D = 6                       |
| 5   | C = 4                       |

| Sno | 2-Itemsets = Support |
|-----|----------------------|
| 1   | FB = 9               |
| 2   | FA = 8               |
| 3   | FD = 6               |
| 4   | FC = 4               |
| 5   | BA = 7               |
| 6   | BD = 5               |
| 7   | BC = 4               |
| 8   | AD = 4               |
| 9   | DC = 3               |
| 10  | AC = 2               |

If we have n items in a set of items I = \{i_1, i_2, i_3, \ldots, i_n\} in a database D, the search space for the FIM problem consists of all possible subsets of I, that is 2^n itemsets. Assuming specific order on the items of set I, the set enumeration tree of I will present all 2^n itemsets. All nodes of the tree except the root node shown in the above Figure 1 are made according to the following two observations.

1. Let $f_p$ be a node label in the SE-tree. Let $C \subseteq I$ be a candidate set of items which is used to extend any node in the tree apart from root node, that is
   $$C = \{ z | z \in I \land \text{supp}(z) \leq \text{supp}(x), \forall x \in f_p \}.$$ 

2. A node label $f_c$ is an extension of a node label $f_p$ if $f_c = f_p \cup X$ where $X \subseteq C$, and $X$ is any possible subset of $C$ such that $X \neq \emptyset$. An itemset $F = f \cup \{ z \}$ is a single item extension or child of $f$ if $z \in C$.

4. Comparative Evaluation

In this section, we will start by presenting Example 2 and considering the transactional dataset given in Table 2.

Example 4: All methods that are included in the frequent itemsets mining algorithm and given in Algorithms 2 and 3 will be applied, in addition to finding all the top-5 IFIs from the given dataset, as illustrated below in Tables 5–9. Moreover, the SE-tree presented in Figure 2 is used to enumerate all of the possible subsets.
Table 5. Candidate and TK\textsubscript{IFIs} List.

| Sno | Candidate List |
|-----|----------------|
| 1   | F = 10         |
| 2   | B = 9          |
| 3   | A = 8          |
| 4   | D = 6          |
| 5   | C = 4          |

| Sno | Top-K IFIs | Support |
|-----|------------|---------|
| 1   | F          | 10      |
| 2   | B          | 9       |
| 3   | A          | 8       |
| 4   | D          | 6       |
| 5   | C          | 4       |

Table 6. Candidate and TK\textsubscript{IFIs} List.

| Sno | Candidate List |
|-----|----------------|
| 1   | FB = 10        |
| 2   | FA = 9         |
| 3   | FD = 8         |
| 4   | FC = 6         |

| Sno | Top-K IFIs | Support |
|-----|------------|---------|
| 1   | F          | 10      |
| 2   | FB, B      | 9       |
| 3   | FA, A      | 8       |
| 4   | FD, D      | 6       |
| 5   | FC, C      | 4       |

Table 7. Candidate and TK\textsubscript{IFIs} List.

| Sno | Candidate List | Pruning Strategies |
|-----|----------------|--------------------|
| 1   | FBA = 7        |                    |
| 2   | FBD = 5, BD = 5| TIBCP              |
| 3   | FBC = 4, C = 4 | IBCP               |

| Sno | Top-K IFIs | Support |
|-----|------------|---------|
| 1   | F          | 10      |
| 2   | FB, B      | 9       |
| 3   | FA, A      | 8       |
| 4   | FBA        | 7       |
| 5   | FD, D      | 6       |

Backtrack on the node with Head Label F to extend with item A from the candidate list \{A, D, C\} \cite{7}\ [13].

Table 8. Candidate and TK\textsubscript{IFIs} List.

| Sno | Candidate List | Pruning Strategies |
|-----|----------------|--------------------|
| 1   | FAD = 5 since AD = 4| TIBCP            |
| 2   | FAC = 4 since C = 4  | IBCP              |

| Sno | Top-K IFIs | Support |
|-----|------------|---------|
| 1   | F          | 10      |
| 2   | FB, B      | 9       |
| 3   | FA, A      | 8       |
| 4   | FBA        | 7       |
| 5   | FD, D      | 6       |

Backtrack on the node with Head Label F to extend with item D from the candidate list \{D, C\}, \{FD\} \cite{13}.
Table 9. Candidate and TK\textsubscript{IFIs} List.

| Sno | Candidate List | Pruning Strategies |
|-----|----------------|--------------------|
| 1   | FDC = 3, C = 4 | IBCP               |

| Sno | Top-K IFIs | Support |
|-----|------------|---------|
| 1   | F          | 10      |
| 2   | FB, B      | 9       |
| 3   | FA, A      | 8       |
| 4   | FBA        | 7       |
| 5   | FD, D      | 6       |

Figure 2. Depth-first-based search space traversal for top-5 IFIs.

Performance Trends

To analyze the performance trends of the TK\textsubscript{IFIs} Miner, it was compared against two recent top-most IFIs mining techniques, top-K Miner [29] and TKFIM [26]. The top-K Miner uses depth-first traversal and is not an Apriori-inspired method, while the TKFIM method uses breadth-first traversal strategy, which is considered an Apriori-inspired approach. All these techniques need one parameter, i.e., K, to find top-most IFIs from the given dataset. For the comparative evaluation, we used six benchmark datasets, as shown in Table 10 below.

Table 10. Datasets used for comparative analysis.

| Sno | Database   | Items | Avg Length | Database Type | Transactions |
|-----|------------|-------|------------|---------------|--------------|
| 1   | Chess      | 75    | 37         | Dense         | 3196         |
| 2   | Connect    | 129   | 43         | Dense         | 67,557       |
| 3   | Retail     | 16469 | 10.3       | Sparse        | 88,162       |
| 4   | T40I10D100K| 1000  | 40         | Dense         | 100,000      |
| 5   | T10I4D100K | 1000  | 10         | Sparse        | 100,000      |

The first two datasets shown in Table 10 above are freely downloadable from the UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/index.php (accessed on 15 September 2022)] [38]. The third, fourth, and fifth datasets are freely downloadable from a frequent itemsets mining datasets repository [http://fimi.uantwerpen.be/data/ (accessed on 22 June 2021)] [39]. The T40I10D100K and T10I4D100K datasets are synthetic in nature, generated on IBM Synthetic Data Generator by IBM Almaden Quest research group. The Chess and Connect datasets are dense in nature. Hence, these datasets will generate result patterns of long and short length if K value is large. The Retail, T40I10D100K,
and T10I4D100K datasets are sparse in nature and the average presence of items in every transaction is low. These datasets will result in patterns of short length even if a large value of the parameter K is supplied.

For experimental evaluation with TKIFIs Miner, we selected three frequent itemsets mining methods: FP-Growth [40], Top-K Miner [29], and TKFIM [26]. The FP-Growth method is used as a benchmark method for experimental evaluation of the many mining methods. Apart from the FP-Growth method, all the other methods apply K as parameter to find topmost IFIs of highest support to the Kth distinct support. The FP-Growth method uses support threshold value for frequent patterns mining. Additionally, FP-Growth, top-K Miner, and TKIFIs Miner use tree data structure and depth-first strategy for mining patterns, whereas the TKFIM approach is an Apriori-inspired algorithm and uses breadth-first strategy to explore the search space. In the experimental work, the support of Kth IFI will be applied to find patterns with FP-Growth. The mapping of K and support threshold parameter that is used to find the same patterns is already established in the work of the Top-K Miner algorithm [29].

On dense datasets such as Chess, Connect, and T40I10D100K, the performance trends of all the methods are almost similar for small values of K from 1 to 20. The top-K Miner, TKFIM, and TKIFIs Miner arrange the search space in descending support order. Therefore, for small values of K these methods compute IFIs of K-highest support in less amount of time. On the other hand, a high support threshold value enables the FP-Growth method to prune the search space and find the same result efficiently. For large values of K, the performance of the TKIFIs Miner is much better due to IBCP and TIBCP pruning strategies. These pruning strategies play a pivotal role enabling the TKIFIs Miner to prune the entire subtree before even finding the support of the candidate patterns. TKFIM and top-K Miner do not apply pruning strategies, hence they perform excessive candidate generation and support computation of patterns which are not even frequent. Therefore, these methods need more time on dense datasets with large values of K than does the TKIFIs Miner. For large values of K, the FP-Growth method uses low support threshold to create lots of conditional pattern trees on dense datasets and for a large number of long and short patterns. Hence, it consumes more time than all the top-K methods. Figures 3–5 present the comparative results from applying the four approaches, FP-Growth, Top-K Miner, TKFIM, and TKIFIs Miner, to the Chess, Connect, and T40I10D100K datasets, respectively.

![Figure 3. The comparative results from applying the four approaches FP-Growth, Top-K Miner, TKFIM, and TKIFIs Miner to Chess dataset.](image-url)
On sparse datasets like Retail and T10I4D100K, the performance of TKIFIs Miner is almost equal for small values of K, as shown in Figures 6 and 7, whereas on large values of K the performance of TKIFIs Miner is better than TKFIM, top-K Miner, and FP-Growth. As already mentioned, the high value of K increases candidate itemsets generation, and the pruning methods start pruning the entire subtrees in TKIFIs Miner, reducing its execution time in comparison to counterparts.
Figure 6. The comparative results from applying the four approaches FP-Growth, Top-K Miner, TKFIM, and TKIFIs Miner to Retail dataset.

Figure 7. The comparative results from applying the four approaches FP-Growth, Top-K Miner, TKFIM, and TKIFIs Miner to T10I4D100K dataset.

5. Conclusions and Future Work

In this article, we presented an efficient TKIFIs Miner algorithm for mining top-K IFIs without using the support threshold parameter from transactional data collected through smart shopping carts. Hence, adjusting this parameter to mine the required number of IFIs is a harder choice for users. Therefore, the TKIFIs Miner algorithm allows users to control the production of the number of IFIs using a parameter K. In TKIFIs Miner, as part of an automatic adjustment of the support threshold, we used new IBCP and TIBCP pruning techniques which were not used earlier in any topmost frequent patterns mining problem. These pruning methods have proven the effectiveness of pruning the entire itemsets’ subtrees from the search space, and hence enabling the depth-first search based on the $Gen_{IFIs}$ method to quickly find those itemsets having a high support value.

TKIFIs Miner outperformed all the topmost patterns mining approaches and the FP-Growth technique in the experimental evaluation. TKIFIs Miner’s computational time is equal to its counterparts on dense and sparse datasets with small values of K. On dense datasets, TKIFIs Miner excels by a bigger margin in terms of computational time for high
values of K. As future work to show the created result-set in a more compact manner, we will integrate maximal or closed patterns mining with the TKNIFIS Miner approach.

Author Contributions: Conceptualization, methodology, S.U.R. Investigation, validation, N.A. Formal analysis, J.A. Software, editing, J.I. Supervision, project administration, resources, review, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors of this paper would like to express their thanks for the research facilities provided by the Huaiyin Institute of Technology, Huai’an, China.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ray, P.P. A survey on Internet of Things architectures. J. King Saud Univ.-Comput. Inf. Sci. 2018, 30, 291–319.
2. Lee, E.; Seo, Y.-D.; Oh, S.-R.; Kim, Y.-G. A Survey on Standards for Interoperability and Security in the Internet of Things. IEEE Commun. Surv. Tutor. 2021, 23, 1020–1047.
3. Kumar, P.; Gupta, G.P.; Tripathi, R. A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. J. Ambient Intell. Humaniz. Comput. 2021, 12, 9555–9572.
4. Kumar, P.; Tripathi, R.; Gupta, G.P. P2IDF: A privacy-preserving based intrusion detection framework for software defined Internet of Things-fog (SDIoT-Fog). In Proceedings of the 2021 International Conference on Distributed Computing and Networking, Nara, Japan, 5–8 January 2021; pp. 37–42.
5. Kumar, P.; Gupta, G.P.; Tripathi, R. Design of anomaly-based intrusion detection system using fog computing for IoT network. Autom. Control Comput. Sci. 2021, 55, 137–147.
6. Vilmos, A.; Medaglia, C.; Moroni, A. NFC Technology and its application Scenarios in a future of IOT. Stolpan Proj. 2011, 1–21.
7. Chen, M.; Wan, J.; Li, F. Machine-to-machine communications: Architectures, standards and applications. KSII Trans. Internet Inf. Syst. 2012, 6, 480–497.
8. Shahroz, M.; Mushtaq, M.F.; Ahmad, M.; Ullah, S.; Mehmood, A.; Choi, G.S. IoT-based smart shopping cart using radio frequency identification. IEEE Access 2020, 8, 68426–68438.
9. Wang, Z.; Liang, W.; Zhang, Y.; Wang, J.; Tao, J.; Chen, C.; Yan, H.; Men, T. Data mining in IoT era: A method based on improved frequent items mining algorithm. In Proceedings of the 2019 5th International Conference on Big Data and Information Analytics (BigDIA), Kunming, China, 8–10 July 2019; pp. 120–125.
10. Chen, F.; Deng, P.; Wan, J.; Zhang, D.; Vasilakos, A.V.; Rong, X. Data Mining for the Internet of Things: Literature Review and Challenges. Int. J. Distrib. Sens. Netw. 2015, 11, 431047.
11. Kireev, V.S.; Guseva, A.I.; Bochkaryov, P.V.; Kuznetsov, I.A.; Filippow, S.A. Association rules mining for predictive analytics in IoT cloud system. In Biologically Inspired Cognitive Architectures Meeting; Springer: Berlin/Heidelberg, Germany, 2018; pp. 107–112.
12. Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 25–28 May 1993; pp. 207–216.
13. Weng, C.-H.; Huang, T. Observation of sales trends by mining emerging patterns in dynamic markets. Appl. Intell. 2018, 48, 4515–4529.
14. Seeja, K.R.; Zareapoor, M. FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining. Sci. World J. 2014, 2014, 252797.
15. Li, H.-F.; Lee, S.-Y.; Shan, M.-K. DSM-PLW: Single-pass mining of path traversal patterns over streaming Web click-sequences. Comput. Netw. 2006, 50, 1474–1487. [CrossRef]
16. Chaudhary, A.; Garg, P. Detecting and diagnosing a disease by patient monitoring system. Int. J. Mech. Eng. Inf. Technol. 2014, 2, 493–499.
17. Caruccio, L.; Deufemia, V.; Naumann, F.; Polese, G. Discovering Relaxed Functional Dependencies Based on Multi-Attribute Dominance. IEEE Trans. Knowl. Data Eng. 2020, 33, 3212–3228. [CrossRef]
18. Papenbrock, T.; Naumann, F. A hybrid approach to functional dependency discovery. In Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 821–833.
19. Wei, Z.; Link, S. Discovery and ranking of functional dependencies. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; pp. 1526–1537.
20. Asmat, N.; Rehman, S.U.; Ashraf, J.; Habib, A. Maximal frequent itemsets based hierarchical strategy for document clustering. In Proceedings of the International Conference on Computer Science, Data Mining and Mechanical Engineering (ICCDMME’2015), Bangkok, Thailand, 20–21 April 2015; pp. 20–21.
21. Ashraf, J.; Ahmed, S.; Ahsan, M. A review on support threshold free frequent itemsets mining approaches. In Proceedings of the 2016 19th International Multi-Topic Conference (INMIC), Islamabad, Pakistan, 5–6 December 2016; pp. 1–6.
22. Pyun, G.; Yun, U. Mining top-k frequent patterns with combination reducing techniques. *Appl. Intell.* 2014, 41, 76–98. [CrossRef]
23. Chuang, K.-T.; Huang, J.-L.; Chen, M.-S. Mining top-k frequent patterns in the presence of the memory constraint. *VLDB J.* 2008, 17, 1321–1344.
24. Fu, A.W.-C.; Kwong, R.W.-W.; Tang, J. Mining n-most interesting itemsets. In Proceedings of the International Symposium on Methodologies for Intelligent Systems, Charlotte, NC, USA, 11–14 October 2000; Springer: Berlin/Heidelberg, Germany, 2000; pp. 59–67.
25. Han, J.; Wang, J.; Lu, Y.; Tzvetkov, P. Mining top-k frequent closed patterns without minimum support. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 211–218.
26. Iqbal, S.; Shahid, A.; Roman, M.; Khan, Z.; Al-Otaibi, S.; Yu, L. TKFIM: Top-K frequent itemset mining technique based on equivalence classes. *PeerJ Comput. Sci.* 2021, 7, e385. [CrossRef]
27. Ngan, S.C.; Lam, T.; Wong, R.C.-W.; Fu, A.W.C. Mining N-most interesting itemsets without support threshold by the COFI-tree. *Int. J. Bus. Intell. Data Min.* 2005, 1, 88–106. [CrossRef]
28. Quang, T.M.; Oyanagi, S.; Yamazaki, K. ExMiner: An efficient algorithm for mining top-k frequent patterns. In Proceedings of the International Conference on Advanced Data Mining and Applications, Xi’an, China, 14–16 August 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 436–447.
29. Rehman, S.u.; Ashraf, J.; Habib, A.; Salam, A. Top-K Miner: Top-K identical frequent itemsets discovery without user support threshold. *Knowl. Inf. Syst.* 2016, 48, 741–762.
30. Salam, A.; Khayal, M. Mining top-k frequent patterns without minimum support threshold. *Knowl. Inf. Syst.* 2012, 30, 57–86.
31. Shen, L.; Shen, H.; Pritchard, P.; Topor, R.W. Finding the N largest itemsets. In *Transactions on Information and Communications Technologies*; Ebecken, N.F.F., Ed.; WIT Press: Southampton, UK, 1998; Volume 22.
32. Cheung, Y.-L.; Fu, A.W.-C. Mining frequent itemsets without support threshold: With and without item constraints. *IEEE Trans. Knowl. Data Eng.* 2004, 16, 1052–1069.
33. Saxena, A.S.; Dawar, S.; Goyal, V.; Bera, D. Mining Top-k Trajectory-Patterns from Anonymized Data. *arXiv* 2019, arXiv:1912.01861.
34. Zeng, J.; Yan, X.; Han, M.; Tang, B. Fast core-based top-k frequent pattern discovery in knowledge graphs. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 936–947.
35. Schirmer, P.; Papenbrock, T.; Kruse, S.; Naumann, F.; Hempfling, D.; Mayer, T.; Neuschafer-Rube, D. DynFD: Functional Dependency Discovery in Dynamic Datasets. In Proceedings of the 22nd International Conference on Extending Database Technology, Lisbon, Portugal, 26–29 March 2019; pp. 253–264.
36. Tan, Z.; Ran, A.; Ma, S.; Qin, S. Fast incremental discovery of pointwise order dependencies. *Proc. VLDB Endow.* 2020, 13, 1669–1681. [CrossRef]
37. Zaki, M.J.; Gouda, K. Fast vertical mining using diffsets. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 August 2003; pp. 326–335.
38. Asuncion, A.; Newman, D. *UCI Machine Learning Repository;* Irvine, CA, USA, 2007.
39. Goethals, B. Frequent itemset mining dataset repository. In Proceedings of the EEE ICDM Workshop on Frequent Itemset Mining Implementations (FIM’03), Melbourne, FL, USA, 19 November 2003.
40. Han, J.; Pei, J.; Yin, Y. Mining frequent patterns without candidate generation. *ACM Sigmod Rec.* 2000, 29, 1–12. [CrossRef]