Evolution and Maintenance of Frequent Pattern Space when Transactions are Removed

Mengling Feng,[1] Guozhu Dong,[2] Jinyan Li,[3] Yap-Peng Tan,[1] Limsoon Wong[4]

[1]Nanyang Technological University, [2]Wright State University,
[3]Institute for Infocomm Research, & [4]National University of Singapore
[1]{feng0010, eyptan}@ntu.edu.sg, [2]guozhu.dong@wright.edu,
[3]jinyan@i2r.a-star.edu.sg, & [4]wongls@comp.nus.edu.sg

Abstract. This paper addresses the maintenance of discovered frequent patterns when a batch of transactions are removed from the original dataset. We conduct an in-depth investigation on how the frequent pattern space evolves under transaction removal updates using the concept of equivalence classes. Inspired by the evolution analysis, an effective and exact algorithm TRUM is proposed to maintain frequent patterns. TRUM maintains frequent patterns efficiently by updating only the affected equivalence classes. Experimental results demonstrate that our algorithm outperforms representative state-of-the-art algorithms.

1 Introduction

Update is a fundamental data management activity. Data updates allow users to remove expired data, to correct data, and to insert new data. Maintenance of a dynamic dataset and its corresponding discovered knowledge is more complicated compared to the knowledge discovery of a stable dataset. Updates may induce new knowledge and invalidate discovered information. Re-execution of discovery algorithms from scratch every time when a database is updated causes significant computation and I/O overheads. Therefore, effective algorithms to maintain discovered knowledge on the updated database without re-execution of mining algorithms are very desirable.

Databases can be updated in several manners. We focus here on the case when a batch of transactions are removed from the existing database. According to [15], transaction removal is one of the most frequently used operations in DBMS. This operation is very crucial in the applications of sales data, transaction records and clinical data to delete expired data and error records.

In this paper, a novel method is proposed to update and maintain discovered frequent patterns [1], an important pattern type in data mining. One major challenge of maintaining frequent patterns is that the pattern space is often huge. E.g. *mushroom* dataset, which has about 8 thousands transactions, returns over 100K frequent patterns when minimum support is 1K. Furthermore, among the enormous frequent pattern space, we have no prior knowledge about which patterns are affected by the transaction removal update. In addition, updates

cause various degrees of impacts on different patterns, e.g. some patterns may experience just a change of support values, but some may be removed completely.

This paper makes the following contributions: (1) We conduct an in-depth analysis on how the frequent pattern space evolves under transaction removal updates using the concept of equivalence classes. To the best of our knowledge, no previous works have studied this. The evolution analysis inspires us to solve the maintenance problem effectively in a divide-n-conquer manner. (2) An effective and exact algorithm, *Transaction Removal Update Maintainer* (TRUM), is proposed to maintain frequent patterns when transactions are removed. TRUM maintains the pattern space effectively by updating only the affected equivalence classes. Besides equivalence classes, TRUM can be applied to update closed and key patterns, since they are the borders of equivalence classes. It is worth pointing out that the maintenance of key patterns has rarely been studied in prior works. (3) Extensive experimental studies are conducted to evaluate the performance of the proposed algorithm. Experiments show that TRUM has significant performance advantage over some state-of-the-art approaches.

2 Preliminaries and Previous Work

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of distinct literals called "items". An "itemset", or a "pattern", is a set of items. A "transaction" is a non-empty set of items. A "dataset" is a non-empty set of transactions. A pattern P is said to be contained or included in a transaction T if $P \subseteq T$. A pattern P is said to be contained in a dataset $\mathcal{D}$, denoted as $P \in \mathcal{D}$, if there is $T \in \mathcal{D}$ such that $P \subseteq T$. The "support" of a pattern P in a dataset $\mathcal{D}$, denoted $sup(P, \mathcal{D})$, is the number of transactions in $\mathcal{D}$ that contain P. A pattern P is said to be *frequent* in a dataset $\mathcal{D}$ if $sup(P, \mathcal{D})$ is greater than or equal to a pre-specified threshold ms. Given a dataset $\mathcal{D}$ and a support threshold ms, the collection of all frequent itemsets in $\mathcal{D}$ is called the "space of frequent patterns", and is denoted by $\mathcal{F}(ms, \mathcal{D})$.

The "space of frequent patterns" can be large. As a result, maximum patterns [3, 8], closed patterns [7, 9], key patterns [12] (also known as generators), and borders of equivalence classes [11] have been proposed to concisely represent the space of frequent patterns. Borders of equivalence classes are arguably the most flexible succinct lossless representation of the frequent pattern space [11]. Conceptually, it partitions the frequent pattern space into equivalence classes that are convex. Then the entire space is represented by the most general and most specific patterns of these equivalence classes. As it turns out, these most general patterns are precisely the key patterns, and these most specific patterns are precisely the closed patterns.

The task of frequent pattern maintenance is to update the "space of frequent patterns" according to the updates of the dataset.

Incremental maintenance, where new transactions are inserted, has attracted intensive research attention. Current incremental maintenance algorithms can be categorized into two main approaches: *Apriori*-based [5, 6, 2] and sliding window filtering (*SWF*) [4, 10]. The performance of both *Apriori*-

based and SWF algorithms is limited by the candidate-generation-elimination framework, which involves multiple data scans and unnecessary computations on infrequent candidates.

To achieve more efficient updates, algorithms are proposed to incrementally maintain only frequent maximum patterns. ZIGZAG[1] [13] is one effective representative. ZIGZAG is inspired by its related work GenMax [8]. It incrementally maintains maximum patterns by a backtracking search, which is guided by the outcomes of previous maintenance iteration.

Decremental maintenance, where old transactions are removed, on the other hand, has not received as much research attention. Zhang et al. [15] proposed an algorithm, named DUA, to address the decremental maintenance problem. DUA maintains frequent patterns by a pairwise comparison of original frequent patterns and patterns included in the removed transactions. Since the number of frequent patterns is usually enormous, the pairwise comparisons cause heavy computations. In addition, algorithms FUP2H [6], Borders [2], ZIGZAG can also be applied to decremental maintenance with some parameter changes.

It is observed that most previous methods are proposed as an extension of some effective data mining algorithms or data structures. E.g. FUP [5] and Borders [2] are developed based on *Apriori* , and ZIGZAG is inspired by GenMax. Unlike these previous works, our algorithm is proposed based on an in-depth study on the evolution of the frequent pattern space.

3 Evolution of Frequent Pattern Space

3.1 Basic Properties of Frequent Pattern Space

The space of frequent patterns possesses the nice convexity property, which is very helpful when it comes to concise and lossless representation and maintenance of the space.

Definition 1. *A space $\mathcal{S}$ is convex if, for all $X, Y \in \mathcal{S}$ such that $X \subseteq Y$, it is the case that $Z \in S$ whenever $X \subseteq Z \subseteq Y$.*

For a convex space $\mathcal{S}$, we define the collection of all "most general" patterns in $\mathcal{S}$ as a "bound" of $\mathcal{S}$, where a pattern X is most general in $\mathcal{S}$ if there is no proper subset of X in $\mathcal{S}$. Similarly, we define the collection of all "most specific" patterns as another bound of $\mathcal{S}$, where a pattern X is most specific in $\mathcal{S}$ if there is no proper superset of X in $\mathcal{S}$. We call the former bound the "left bound" of $\mathcal{S}$, denoted $\mathcal{L}$; and the latter bound the "right bound" of $\mathcal{S}$, denoted $\mathcal{R}$. We call the pair of left and right bound the "border" of $\mathcal{S}$, which is denoted by $\langle \mathcal{L}, \mathcal{R} \rangle$. A space can be concisely represented by its borders without loss of information.

Fact 2 (Cf. [11]) *$\mathcal{F}(ms, \mathcal{D})$ is convex. Furthermore, it can be structurally decomposed into convex sub-spaces — equivalence classes.*

[1] We thank Adriano Alonoso Veloso, Professor Srinivasan Parthasarathy and Professor Mohammed J. Zaki for providing the ZIGZAG source code.

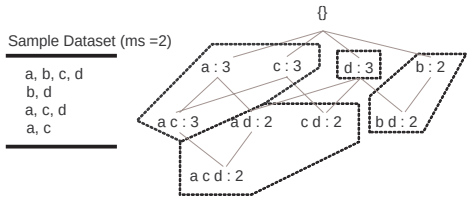

Fig. 1. Demonstration of how a space of frequent patterns, which contains 9 patterns, is decomposed into 5 frequent equivalence classes.

The fact indicates that the space of frequent patterns is a convex space. We found that convex space has an interesting property: it can be decomposed into convex sub-spaces. In the case of frequent pattern space, it can be further decomposed systematically into equivalence classes.

Definition 3. *Let the "filter", $f(P, \mathcal{D})$, of a pattern P in a dataset $\mathcal{D}$ be defined as $f(P, \mathcal{D}) = \{T \in \mathcal{D} \mid P \subseteq T\}$. Then the "equivalence class" $[P]_{\mathcal{D}}$ of P in a dataset $\mathcal{D}$ is the collection of patterns defined as $[P]_{\mathcal{D}} = \{Q \mid f(P, \mathcal{D}) = f(Q, \mathcal{D})$, Q is a pattern in $\mathcal{D}\}$. Note that under this definition, $[Q]_{\mathcal{D}} = \emptyset$ if Q does not appear in $\mathcal{D}$. For convenience in some of our proofs, we also use the traditional notion of an equivalence class, and write it as $[P]^*_{\mathcal{D}} = \{Q \mid f(P, \mathcal{D}) = f(Q, \mathcal{D})\}$.*

In other words, two patterns are "equivalent" in the context of a dataset $\mathcal{D}$ iff they are included in exactly the same transactions in $\mathcal{D}$. Thus the patterns in a given equivalence class have the same support. So we extend the notations and write $sup(C, \mathcal{D})$ to denote the support of an equivalence class and $C \in \mathcal{F}(ms, \mathcal{D})$ to mean the equivalence class is frequent. Figure 1 presents the frequent pattern space for the sample dataset with $ms = 2$. In addition, it graphically demonstrates how the space of frequent patterns can be structurally decomposed into frequent equivalence classes.

Structural decomposition of frequent pattern space inspired us to solve the maintenance problem in a divide-and-conquer manner. Instead of maintaining the pattern space as a whole, which is computationally costly, we attack the problem by maintaining each frequent equivalence class. Compared with the frequent pattern space, an equivalence class is much smaller and easier to update. Moreover, not all the equivalence classes are affected by the updates. If we can efficiently locate only those equivalence classes that are affected by the updates, we can solve the problem effectively by updating only the affected equivalence classes. In addition, a nice property of equivalence classes of patterns is that they are convex and they can be concisely represented by their borders.

Fact 4 (Cf. [11]) *$[P]_{\mathcal{D}}$ is convex, and the right bound of its border is a singleton set.*

Together with equivalence classes, frequent "closed patterns" and frequent "key patterns" (also called "generators") have been widely studied in the data

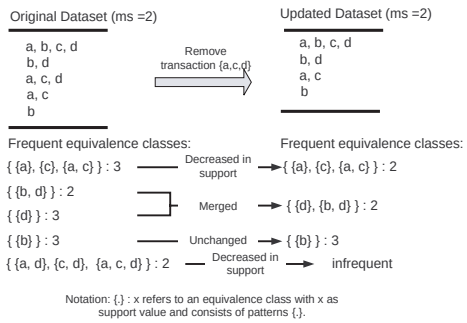

Fig. 2. An example to demonstrate how equivalence classes and the frequent pattern space may evolve when a transaction is removed.

mining field. We discuss next the relationship between equivalence classes and closed and key patterns.

Definition 5. *A pattern P is a "key pattern" in a dataset $\mathcal{D}$ iff for every $P' \subset P$, it is the case that $sup(P', \mathcal{D}) > sup(P, \mathcal{D})$. In contrast, a pattern P is a "closed pattern" in a dataset $\mathcal{D}$ iff for every $P' \supset P$, it is the case that $sup(P', \mathcal{D}) < sup(P, \mathcal{D})$.*

It is discovered in [11] that the right bound of an equivalence class is actually a closed pattern, and the left bound is a group of key patterns. Thus, the corresponding closed and key patterns form the border of and define an equivalence class. Following the definition of borders of convex spaces, a key pattern must be most general in its equivalence class. Similarly, a closed pattern must be most specific in its equivalence class. So we have the following alternative equivalent definitions for key and closed patterns.

Fact 6 *A pattern P is a key pattern in a dataset $\mathcal{D}$ iff P is a most general pattern in $[P]_{\mathcal{D}}$. A pattern P is a closed pattern in a dataset $\mathcal{D}$ iff P is the most specific pattern in $[P]_{\mathcal{D}}$. Therefore, to mine or maintain key and closed patterns, it is sufficient to mine or maintain the borders of equivalence classes, and vice versa.*

3.2 Impacts of Transaction Removal

We investigate in this section how frequent patterns, key patterns, closed patterns, equivalence classes and their support values evolve when multiple transactions are removed from an existing dataset. We use the following notations: $\mathcal{D}_{org}$ is the original dataset, $\mathcal{D}_{dec}$ is the set of old transactions to be removed, and $\mathcal{D}_{upd-} = \mathcal{D}_{org} - \mathcal{D}_{dec}$ is the updated dataset. We assume without loss of generality that $\mathcal{D}_{dec} \subseteq \mathcal{D}_{org}$.

An existing equivalence class can evolve in exactly three ways, as shown in Figure 2. The first way is to remain unchanged without any change in support. The second way is to remain unchanged but with a decreased support. If the support of an existing frequent equivalence class drops below the minimum support threshold, the equivalence class will be removed. The third way is to grow—by merging with other classes, where at most one of the merging classes has the same closed pattern and the same support as the resulting equivalence class and all other merging classes have lower support. In short, after the decremental update, the support of an equivalence class can only decrease and the size of an equivalence class can only grow by merging.

In order to have an in-depth understanding of the three ways that an existing equivalence class may evolve, we now provide the exact conditions for each of these ways to occur. The evolution of frequent pattern space under decremental updates are characterized as follows:

Theorem 1. *For every frequent equivalence class $[P]_{\mathcal{D}_{org}}$ in $\mathcal{D}_{org}$, exactly one of the 6 scenarios below holds:*

1. *P is frequent in $\mathcal{D}_{org}$, P is not in $\mathcal{D}_{dec}$, and $f(P, \mathcal{D}_{org}) \neq f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{dec}$, corresponding to the scenario where an equivalence class has remained totally unchanged. In this case, $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}}$, $sup(P, \mathcal{D}_{upd-}) = sup(P, \mathcal{D}_{org})$, $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org})$, and the closed pattern of $[P]_{\mathcal{D}_{upd-}}$ is the same as that of $[P]_{\mathcal{D}_{org}}$. The key patterns of $[P]_{\mathcal{D}_{upd-}}$ are the same as that of $[P]_{\mathcal{D}_{org}}$.*
2. *P is frequent in $\mathcal{D}_{org}$, P is not in $\mathcal{D}_{dec}$, and $f(P, \mathcal{D}_{org}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q occurring in $\mathcal{D}_{dec}$, corresponding to the scenario where the equivalence class of Q has to be merged into the equivalence class of P. In this case, let all such Q's in $\mathcal{D}_{dec}$ be grouped into n distinct equivalence classes $[Q_1]^*_{\mathcal{D}_{dec}}$, ..., $[Q_n]^*_{\mathcal{D}_{dec}}$, having representatives Q_1, ..., Q_n satisfying the condition on Q. Then $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$, $sup(P, \mathcal{D}_{upd-}) = sup(P, \mathcal{D}_{org})$, $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org})$, and the closed pattern of $[P]_{\mathcal{D}_{upd-}}$ is the same as the closed pattern of $[P]_{\mathcal{D}_{org}}$. The key patterns of $[P]_{\mathcal{D}_{upd-}}$ are the most general ones among the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$. Furthermore, $[Q_i]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{upd-}}$ for $1 \leq i \leq n$.*
3. *P is frequent in $\mathcal{D}_{org}$, P is in $\mathcal{D}_{dec}$, and $|f(P, \mathcal{D}_{upd-})| < ms$, corresponding to the scenario where the equivalence class is removed.*
4. *P is frequent in $\mathcal{D}_{org}$, P is in $\mathcal{D}_{dec}$, and $f(Q, \mathcal{D}_{org}) = f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for some Q that is frequent in $\mathcal{D}_{org}$ but not in $\mathcal{D}_{dec}$, corresponding to the scenario where the equivalence class of P has to be merged into the equivalence class of Q. This scenario is complement to Scenario 2. In this case, the equivalence class, support, key, and closed patterns of $[P]_{\mathcal{D}_{upd-}}$ is the same as that of $[Q]_{\mathcal{D}_{upd-}}$, as computed in Scenario 2.*
5. *P is frequent in $\mathcal{D}_{org}$, P is in $\mathcal{D}_{dec}$, $|f(P, \mathcal{D}_{upd-})| > ms$, $f(Q, \mathcal{D}_{org}) \neq f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{org}$ and not in $\mathcal{D}_{dec}$, and $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) \neq f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$, corresponding to the situation where the equivalence class has remained unchanged but has decreased in support. In this case, $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}}$, $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$, $sup(P, \mathcal{D}_{upd-}) = sup(P, \mathcal{D}_{org}) - sup(P, \mathcal{D}_{dec})$, and the closed pattern of $[P]_{\mathcal{D}_{upd-}}$ is the same as that of $[P]_{\mathcal{D}_{org}}$. The key patterns of $[P]_{\mathcal{D}_{upd-}}$ are the same as that of $[P]_{\mathcal{D}_{org}}$.*

6. *P is frequent in $\mathcal{D}_{org}$, P is in $\mathcal{D}_{dec}$, $|f(P, \mathcal{D}_{upd-})| > ms$, $f(Q, \mathcal{D}_{org}) \neq f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{org}$ and not in $\mathcal{D}_{dec}$, and $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$, corresponding to the situation where the equivalence classes of P and Q have to be merged. In this case, let all such Q's in $\mathcal{D}_{dec}$ be grouped into n distinct equivalence classes $[Q_1]^*_{\mathcal{D}_{dec}}$, ..., $[Q_n]^*_{\mathcal{D}_{dec}}$, having representatives Q_1, ..., Q_n satisfying the condition on Q. Then $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$, $sup(P, \mathcal{D}_{upd-}) = sup(P, \mathcal{D}_{org}) - sup(P, \mathcal{D}_{dec})$, and $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$. The closed pattern of $[P]_{\mathcal{D}_{upd-}}$ is the most specific pattern among the closed patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$. The key patterns of $[P]_{\mathcal{D}_{upd-}}$ are the most general ones among the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$. Furthermore, $[Q_i]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{upd-}}$ for $1 \leq i \leq n$.*

Proof. Refer to the Appendix (`http://www.ntu.edu.sg/home5/feng0010/appendix.pdf`) for the detailed proof.

This theorem describes in detail how the space of frequent patterns evolves when a group of transactions are removed. Moreover, it describes how to derive equivalence classes in $\mathcal{D}_{upd-}$ from existing equivalence classes in $\mathcal{D}_{org}$, which is an extremely constructive result for the maintenance of frequent patterns.

4 Proposed Algorithm: TRUM

An algorithm for maintaining the frequent pattern space after some transactions are removed from the original database is proposed in Figure 3. In the proposed algorithm TRUM, we use notations $X.closed$ to mean the closed pattern of an equivalence class, $X.keys$ to mean the set of keys of an equivalence class, and $X.sup$ to denote the support value of an equivalence class. The algorithm addresses the maintenance problem effectively by working on the borders of equivalence classes, instead of the entire pattern space. The proposed algorithm is proved to be correct and complete in the Appendix (`http://www.ntu.edu.sg/home5/feng0010/appendix.pdf`).

According to Figure 3, the maintenance problem mainly consists of two major computational tasks. The first task is to update the support values of each existing frequent equivalence classes. The second task is to merge equivalence classes that are to be joined together after the decremental update.

4.1 Implementation Techniques

TRUM is implemented efficiently with a novel data structure — Tid-tree. The Tid-tree is developed based on the concept of Transaction Identifier List, in short Tid-list. The Tid-list is very popular in the literature of data mining [8, 13]. Tid-lists, serve as the vertical projections of items, greatly facilitate the discovery of frequent itemsets and their support. A new feature of Tid-lists is exploited here. They are utilized as the identifers of equivalence classes. Each frequent equivalence class is associated with a Tid-list, which records all the transactions it appears in. According to the definition of an equivalence class, each equivalence

Input: The set $\mathcal{O} = O_1, ..., O_n$ of frequent equivalence classes in $\mathcal{D}_{org}$,represented by their borders—viz., the corresponding key and closed patterns and supports—and identified by their unique closed patterns, and the set $\mathcal{T} = T_1, ..., T_m$ of transactions in $\mathcal{D}_{dec}$, and the minimum support threshold ms.
Output: The set $O'_1, ..., O'_n$ (if they still exist) of updated frequent equivalence classes in $\mathcal{D}_{upd-}$ represented by their borders and identified by their unique closed patterns.
Method:
{Scenario 1 is default for equivalence classes in $\mathcal{D}_{org}$}
$O'_1 := O_1; ...; O'_n := O_n$;
for all $T \in \mathcal{T}$, $O \in \mathcal{O}$ **do**
  if $O.closed \subseteq T$ **then**
    {Scenario 5 is default for equivalence classes, whose support is decreased due to the decremental update. O' has already been initialized as per Scenario 1.}
    $O'.sup := O'.sup - 1$;
  end if
end for
for all $O'_i \in \{O'_1, ..., O'_n\}$ (if they exist) **do**
  if $O'_i.sup < ms$ **then**
    {Scenario 3. The equivalence class is no longer frequent.}
    Remove O'_i, continue;
  end if
  for all $O'_j \in \{O'_{i+1}, ..., O'_n\}$ (if they exist) **do**
    if $O'_i.sup = O'_j.sup$ & $O'_j.closed \subset O'_i.closed$ **then**
      {Scenario 2 & 6. O'_j merges into O'_i.}
      $O'_i.keys := min\{K | K \in O'_i.keys \ or K \in O'_j.keys\}$
      Remove O'_j;
    end if
    if $O'_i.sup = O'_j.sup$ & $O'_j.closed \supset O'_i.closed$ **then**
      {Scenario 4 & 6. O'_i merges into O'_j.}
      $O'_j.keys := min\{K | K \in O'_i.keys \ or K \in O'_j.keys\}$
      Remove O'_i;
    end if
  end for
end for
return O'_1, ..., O'_n (if they still exist);

Fig. 3. TRUM: a novel algorithm for maintaining frequent patterns after some transactions are removed from the original database.

class has a unique Tid-list, and so can be identified by it. This observation forms the foundation of the proposed implementation technique. To construct a Tid-list, we need to assign a unique Tid to each transaction as shown in Part (a) of Figure 4.

The Tid-tree is a prefix tree of the Tid-lists of equivalence classes. The prefix tree has been used as a concise storage of frequent patterns [9] and closed patterns [7]. In this implementation, Tid-tree serves as a concise storage of Tid-lists of the existing equivalence classes. Figure 4 (b) shows how the Tid-lists of equivalence classes in Figure 4 (a) can be stored in a Tid-tree. Details of the construction of a prefix tree can be referred to [7]. Here, we emphasize two features of the Tid-tree: (1) Each node in the Tid-tree stores a Tid. If the Tid of the node is the last Tid in some equivalence class's Tid-list, the node points to the corresponding equivalence class. Moreover, the depth of the node reflects the support of the corresponding equivalence class. (2) The Tid-tree has a header table, where each slot stores a linked list that connects all the nodes with the same Tid.

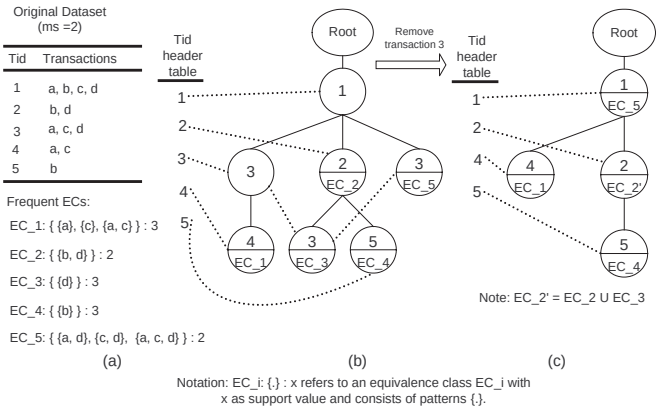

Fig. 4. A running example of TRUM with Tid-tree implementation. (a) Original dataset with Tids and its frequent equivalence classes (ECs). (b)Tid-tree for the original dataset. (c) Tid-tree after removal of transaction 3.

When transactions are removed from the original dataset, the Tid-tree can be updated by removing all the nodes that include the Tids of deleted transactions. This can be accomplished effectively with the help of the Tid header table. As demonstrated in Figure 4, after a node is removed, its children re-link to its parent to maintain the tree structure. If the node points to an equivalence class, the pointer is passed to its parent. When two or more equivalence class pointers collide into one node, they should be merged together. E.g. in Figure 4, equivalence class EC_2 and EC_3 of the original dataset merge into EC_2' after the update.

With the Tid-tree, two major maintenance computational tasks of TRUM are accomplished in one step as we remove Tids from the Tid-tree. Since the Tid linked list can only be removed one by one, the computational complexity of TRUM is $O(|\mathcal{D}_{dec}|)$, where $|\mathcal{D}_{dec}|$ denotes the size of the decremental dataset. TRUM is much more computationally effective, compared to previous works, like [8, 13], whose computational complexity is $O(N_{FP})$, where N_{FP} refers to the number of frequent patterns. This is because $O(|\mathcal{D}_{dec}|) \ll N_{FP}$.

4.2 Limitations and Extensions

TRUM is developed under the setting where minimum support threshold ms is defined in counts. This definition of ms is used in applications like [14]. But in some other applications, ms may be defined in percentages. In such a case, the actual support (in absolute count) of a pattern for satisfying the ms threshold (in percentage) drops after some transactions are removed. Thus new frequent patterns may emerge. So TRUM cannot be directly applied. Nonetheless, this problem can be solved by extending the Tid-tree.

Instead of storing only frequent equivalence classes, whose supports are above the percentage threshold $x\%$, we also include equivalence classes, whose supports are above $(x - \triangle)\%$, in the Tid-tree. In this way, we actually build a "buffer" in the Tid-tree, which contains a group of infrequent equivalence classes that are likely to become frequent after some transactions are removed. This means that, so long as the total number of deletions does not exceed $|\mathcal{D}_{org}| \times \triangle\%$ ($|\mathcal{D}_{org}|$ is the size of $\mathcal{D}_{org}$), all the "new" equivalence classes that may emerge are already kept in the "buffer" of the extended Tid-tree. With the extended Tid-tree, TRUM can now be employed for multiple rounds of decremental maintenance, as long as the accumulated amount of deletion is less than $|\mathcal{D}_{org}| \times \triangle\%$. As the amount of deletion gets close to this limit, we re-execute the discovery algorithm and rebuild the buffer. The size of the $\triangle\%$ buffer can be adjusted based on specific application requirements.

5 Experimental Studies

Extensive experiments were performed to evaluate the proposed algorithm. TRUM was tested using several benchmark datasets from the *FIMI* Repository, `http://fimi.cs.helsinki.fi`. Due to space constraints, only the results of $T10I4D100K$, *mushroom* and *gazelle* are presented in this paper. These datasets form a good representative of both synthetic and real datasets.

We varied two parameters in our experiments: minimum support ms and update interval. For each employed ms, we preformed multiple execution of the algorithm, where each execution employed a different update interval. Moreover, the performance of the algorithm varies slightly when different sets of transactions are removed. To have a stable performance measure, for each update interval, 5 random sets of transactions were employed, and the average performance of the algorithm was recorded. The experiments were run on a PC with 2.8GHz processor and 2GB main memory.

To justified the effectiveness of the proposed algorithm, we compared its performance against some state-of-art frequent pattern discovery and maintenance algorithms. These algorithms includes ZIGZAG [13], FpClose [9] and GC-growth [11]. ZIGZAG is one of the most recently proposed algorithms, which also addresses the maintenance of frequent patterns when transactions are removed. It outperforms most of the previous works. On the other hand, FpClose, according to our knowledge, is the fastest algorithm for closed pattern mining. GC-growth is the only algorithm that generates frequent equivalence classes. Results of the performance comparison is presented in Figure 5.

We observe that TRUM outperforms ZIGZAG by at least an order of magnitude over all update intervals. The advantage of the proposed algorithm is most obvious in *mushroom* dataset. For *mushroom* dataset, TRUM, on average, outperforms ZIGZAG 200 times. It is measured that, for both $T10I4D100K$ and *gazelle*, TRUM achieves around 80 and 20 times average speed-up.

TRUM is also more effective compared to re-discovering all patterns using FpClose and GC-growth. E.g. TRUM is, on average, 30 times faster than FpClose

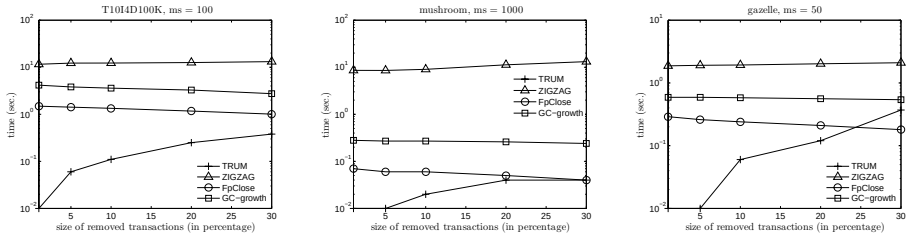

Fig. 5. Average run time comparison of ZIGZAG, FpClose, GC-growth and TRUM.

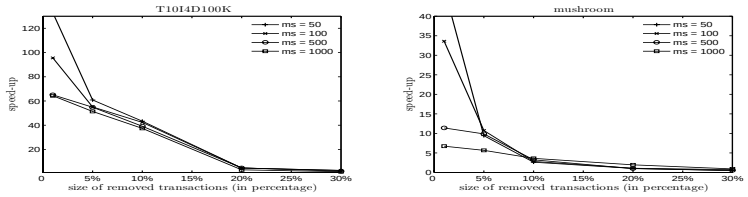

Fig. 6. Speed-up achieved by TRUM against FpClose over various ms thresholds.

and 100 times faster than GC-growth for $T4I10D100K$ dataset. However, we also observe that as the size of the removed transactions increases, the advantage of TRUM diminishes. This is because, corresponding to the complexity analysis, the execution time of TRUM increases as more transactions are removed. In contrast, due to the shrinkage of data size, the execution time of re-discovery approaches drops when more transactions are removed. Combining these two effects, it is logical that the speed-up gained by our maintenance approach diminishes as the size of removed transactions goes up.

The performance of the proposed algorithm was also evaluated under different support thresholds ms. The results are presented in Figure 6. It demonstrates that TRUM remains effective compared to FpClose over a wide range of minimum support thresholds. Nevertheless, the achieved speed-up drops slightly for higher ms thresholds. When ms is high, the frequent pattern space becomes smaller, which makes the discovery process much easier. As a result, the advantage of TRUM becomes less obvious.

6 Closing Remarks

This paper has investigated how the space of frequent patterns, equivalence classes, closed and key patterns will evolve when transactions are removed from a given dataset. It was shown that the equivalence classes can evolve in three ways: (1) remain unchanged with the same support value, (2) remain unchanged with decreased support value, and (3) grow by merging with others. Based the evolution analysis, an effective maintenance algorithm TRUM is proposed. TRUM maintains the frequent pattern space in a divide-and-conquer manner using the

concept of equivalence classes. With the newly proposed data structure — Tid-tree, TRUM addresses the problem efficiently by updating only the affected equivalence classes. The effectiveness of the proposed algorithm is validated by experimental evaluations.

This paper, to our best knowledge, is the first to study the evolution of frequent pattern space under data updates. The proposed algorithm outperforms the state-of-the-art algorithms at least an order of magnitude over a wide range of support thresholds and update sizes.

In the future, it is interesting to exploit the evolution of frequent pattern space under other types of updates, e.g. addition of transaction and items, or removal of items. Solving these maintenance problems with a divide-n-conquer approach could be promising.

References

[1] R. Agrawal, et al. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[2] Y. Aumann, et al. Borders: An efficient algorithm for association generation in dynamic databases. In *JIIS*, (12) page 61-73, 1999.

[3] R. J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD*, pages 85–93, 1998.

[4] C. Chang, et al. Enhancing SWF for incremental association mining by itemset maintenance. In *PAKDD*, pages 301–312, 2003.

[5] D. Cheung, et al. Maintenance of discovered association rules in large databases: an incremental update technieq. In *ICDE*, pages 106–114, 1996.

[6] D. Cheung, et al. A general incremental technique for maintaining discovered association rules. In *Proc. 1996 DASFAA*, pages 185–194, 1997.

[7] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Proc. 1st IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[8] K. Gouda, et al. GenMax: An efficient algorithm for mining maximal frequent itemsets. In *Data Mining and Knowledge Discovery: An International Journal*, 11: 1-20, 2005.

[9] J. Han, et al. Mining frequent patterns without candidates generation. In *SIGMOD*, pages 1–12, 2000.

[10] C. Lee, et al. Sliding window filtering: An efficient method for incremental mining on a time-variant database. *Information Systems*, 30(3):227-244, 2005.

[11] H. Li, et al. Relative risk and odds ratio: A data mining perspective. In *PODS*, pages 368–377, 2005.

[12] N. Pasquier, et al. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[13] A.A. Veloso, et al. Mining frequent itemsets in evolving databases. In *SIAM*, 2002.

[14] C. Wong, et al. Parallel Algorithms for Mining Frequent Structural Motifs in Scientific Data In *ICS*, pages 31-40, 2004.

[15] S. Zhang, et al. A decremental algorithm for maintaining frequent itemsets in dynamic databases. In *DaWak*, pages 305–314, 2005.

Appendix: Proofs of Propositions and Theorems

Evolution of Frequent Pattern Space

To better appreciate Theorem 1, we discuss here the evolution of frequent pattern space when multiple transactions are removed in a greater detail.

The transaction removal operation is very disruptive to the space of frequent patterns. Most of the characteristics of the original frequent pattern space are not preserved. Let us review one of the few characteristics that are preserved by the decremental updates.

Proposition 1. *Suppose P and Q are patterns in both $\mathcal{D}_{org}$ and $\mathcal{D}_{upd-}$. Then $Q \in [P]_{\mathcal{D}_{upd-}}$ if $Q \in [P]_{\mathcal{D}_{org}}$. Consequently, if P occurs in $\mathcal{D}_{org}$ and P does not occur in $\mathcal{D}_{upd-}$, then all patterns in $[P]_{\mathcal{D}_{org}}$ disappear in $\mathcal{D}_{upd-}$.*

This Proposition expresses two observations. First, suppose two patterns are in the same equivalence class in the original dataset. They are still in the same equivalence class in the updated dataset (if they still exist), and vice versa. Secondly, equivalence classes may disappear after a group of transactions are removed.

It is also observed that, after the decremental update, the support of an equivalence class can only decrease and the size of an equivalence class can only grow by merging. The details of this observation are described in the following propositions.

Proposition 2. *Let P be a pattern in $\mathcal{D}_{upd-}$. Then $[P]_{\mathcal{D}_{upd-}} \supseteq [P]_{\mathcal{D}_{org}}$, and $sup(P, \mathcal{D}_{upd-}) \leq sup(P, \mathcal{D}_{org})$.*

Proposition 3. *Let P be a pattern in $\mathcal{D}_{upd-}$, and Q_1, ..., Q_n be patterns in $\mathcal{D}_{org}$ and $[Q_i]_{\mathcal{D}_{org}} \cap [Q_j]_{\mathcal{D}_{org}} = \emptyset$ for $1 \leq i \neq j \leq n$. Then $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$ merge to form $[P]_{\mathcal{D}_{upd-}}$ iff $[P]_{\mathcal{D}_{upd-}} = \bigcup_{1 \leq i \leq n} [Q_i]^*_{\mathcal{D}_{dec}} \cap [Q_i]_{\mathcal{D}_{org}}$, where $Q_i \in [P]_{\mathcal{D}_{upd-}}$ for $1 \leq i \leq n$ and $[Q_i]^*_{\mathcal{D}_{dec}} \cap [Q_j]^*_{\mathcal{D}_{dec}} = \emptyset$ for $1 \leq i < j \leq n$. Furthermore, $[Q_i]_{\mathcal{D}_{org}} = [Q_i]^*_{\mathcal{D}_{dec}} \cap [Q_i]_{\mathcal{D}_{org}}$ for $1 \leq i \leq n$. Moreover, there is at most one Q_i among Q_1, ..., Q_n, such that Q_i does not occur in $\mathcal{D}_{dec}$ and $Q_i \in [C]_{\mathcal{D}_{org}}$, where C is the closed pattern of $[P]_{\mathcal{D}_{upd-}}$. Also, if $P \in \mathcal{F}(ms, \mathcal{D}_{upd-})$, then $Q_i \in \mathcal{F}(ms, \mathcal{D}_{org})$ for $1 \leq i \leq n$.*

Proof of Theorem 1

Proof. Scenario 1 and Scenario 3 is obvious. Scenario 4 follows from Scenario 2.

To prove Scenario 5, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is in $\mathcal{D}_{dec}$, (iii) $|f(P, \mathcal{D}_{upd}-)| > ms$, (iv) $f(Q, \mathcal{D}_{org}) \neq f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{org}$ and not in $\mathcal{D}_{dec}$, and (v) $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) \neq f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$. Point (iv) implies that $[P]_{\mathcal{D}_{upd-}} \neq [Q]_{\mathcal{D}_{upd-}}$ for all Q not in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$. Point (v) implies that $[P]_{\mathcal{D}_{upd-}} \neq [Q]_{\mathcal{D}_{upd-}}$ for all Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$. Thus no pattern Q outside of $[P]_{\mathcal{D}_{org}}$ can become a member of $[P]_{\mathcal{D}_{upd-}}$. Thus $[P]_{\mathcal{D}_{upd-}} \subseteq [P]_{\mathcal{D}_{org}}$. Then $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}}$ by Proposition 2. The rest of the results for this scenario now follow straightforwardly.

To prove Scenario 2, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is not in $\mathcal{D}_{dec}$, and (iii) $f(P, \mathcal{D}_{org}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q occurring in $\mathcal{D}_{dec}$. Also, let all such Q's in $\mathcal{D}_{dec}$ be grouped into n distinct equivalence classes $[Q_1]_{\mathcal{D}_{dec}}$, ..., $[Q_n]_{\mathcal{D}_{dec}}$, having representatives Q_1, ..., Q_n satisfying the condition on Q. Points (i) and (ii) imply $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org})$ and $sup(P, \mathcal{D}_{upd-}) = sup(P, \mathcal{D}_{org})$. Let C be the closed pattern of $[P]_{\mathcal{D}_{org}}$. By Proposition 3, we conclude that C is the closed pattern of $[P]_{\mathcal{D}_{upd-}}$ and that $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. Points (i), (ii), and (iii) imply that $f(P, \mathcal{D}_{upd-}) = f(P, \mathcal{D}_{org}) = f(Q_i, \mathcal{D}_{org}) - f(Q_i, \mathcal{D}_{dec}) = f(Q_i, \mathcal{D}_{upd-})$ for $1 \leq i \leq n$. Thus $[P]_{\mathcal{D}_{upd-}} = [Q_i]_{\mathcal{D}_{upd-}}$ for $1 \leq i \leq n$.

Next, we show that the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$ are also patterns of $[P]_{\mathcal{D}_{upd-}}$. It basically follows the results of Proposition 1. This also implies that the non-minimal ones cannot be key patterns of $[P]_{\mathcal{D}_{upd-}}$.

Now, to show that the key patterns of $[P]_{\mathcal{D}_{upd-}}$ are the most general ones among the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$, it is sufficient to show instead that the key patterns of $[P]_{\mathcal{D}_{upd-}}$ are also the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$. Suppose K is a key pattern of $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. Then K is a pattern in $[P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. Without loss of generality, suppose K is a pattern in $[Q_i]_{\mathcal{D}_{org}}$. If K is not a key pattern of $[Q_i]_{\mathcal{D}_{org}}$, this would imply there is a pattern $K' \in [Q_i]_{\mathcal{D}_{org}}$ that is more general than K. However, such a K' would also be a pattern in $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. This would contradict that assumption that K is a key pattern of $[P]_{\mathcal{D}_{upd-}}$. Hence, such a K' could not have existed, and thus K is a key pattern of $[Q_i]_{\mathcal{D}_{org}}$. Thus, every key pattern of $[P]_{\mathcal{D}_{upd-}}$ is a key pattern among the key patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$. This completes the proof for Scenario 2.

As for Scenario 6, it can be proved in the same way as Scenario 2, with a small modification to handle the inference of the closed pattern. In particular, we assume that $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q occurring in $\mathcal{D}_{dec}$. Then let all such Q's in $\mathcal{D}_{dec}$ be grouped into n distinct equivalence classes $[Q_1]_{\mathcal{D}_{dec}}$, ..., $[Q_n]_{\mathcal{D}_{dec}}$, having representatives Q_1, ..., Q_n satisfying the condition on Q. By Proposition 3, $[P]_{\mathcal{D}_{upd-}} = [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. Let C be the closed pattern of $[P]_{\mathcal{D}_{upd-}}$. Then $C \in [P]_{\mathcal{D}_{org}} \cup \bigcup_i [Q_i]_{\mathcal{D}_{org}}$. Since no pattern in $[P]_{\mathcal{D}_{upd-}}$ is more specific than C, we conclude that C is the most specific closed pattern among the closed patterns of $[P]_{\mathcal{D}_{org}}$, $[Q_1]_{\mathcal{D}_{org}}$, ..., $[Q_n]_{\mathcal{D}_{org}}$, as desired for Scenario 6. The rest of the proof for this scenario is exactly the same as that for Scenario 2.

Completeness and Correctness of Proposed Algorithm

Closed patterns enjoy the "anti-monotonicity" property, in the sense that a closed pattern is a subset of another closed pattern whenever the filter of the latter is a subset of the former. Similarly, equivalence classes of patterns enjoy the "anti-monotonicity" property.

Fact 7 *1. Let P and Q be closed patterns in $\mathcal{D}$. Then $P \subseteq Q$ iff $f(Q, \mathcal{D}) \subseteq f(P, \mathcal{D})$.*
2. Suppose $\mathcal{D}' \subseteq \mathcal{D}$. Then $[P]_{\mathcal{D}} \subseteq [P]_{\mathcal{D}'}$ iff $f(P, \mathcal{D}') \subseteq f(P, \mathcal{D})$.

Following the Fact 7, we have Lemma 1 and Corollary 1.

Lemma 1. *Let $[P]_{\mathcal{D}_{org}}$ be an equivalence class in $\mathcal{D}_{org}$. Then P is in $\mathcal{D}_{dec} \subseteq \mathcal{D}_{org}$ iff $[P]_{\mathcal{D}_{org}} \subseteq [P]_{\mathcal{D}_{dec}}$. Furthermore, since equivalence classes partition the*

patterns in a dataset, it follows that $[Q]_{\mathcal{D}_{dec}} = [P]_{\mathcal{D}_{dec}}$ *whenever* $P' \in [Q]_{\mathcal{D}_{dec}}$ *for any* $P' \in [P]_{\mathcal{D}_{org}}$.

Corollary 1. *Let* $[P]_{\mathcal{D}_{org}}$ *be an equivalence class in* $\mathcal{D}_{org}$ *having* C *as its closed pattern. Let* $[Q]_{\mathcal{D}_{dec}}$ *be an equivalence class in* $\mathcal{D}_{dec}$. *Suppose* $C \in [Q]_{\mathcal{D}_{dec}}$. *Then for every pattern* $P' \in [P]_{\mathcal{D}_{org}}$, *it is the case that (i)* $P' \in [Q]_{\mathcal{D}_{dec}}$, *(ii)* $f(P', \mathcal{D}_{dec}) = f(Q, \mathcal{D}_{dec})$, *(iii)* $f(P', \mathcal{D}_{upd-}) = f(P', \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$, *and (iv)* $sup(P', \mathcal{D}_{upd-}) = sup(P', \mathcal{D}_{org}) - sup(Q, \mathcal{D}_{dec})$.

With the above results, we are now ready to prove the correctness and completeness of the proposed algorithm.

Theorem 2. *The algorithm given in Figure 3 correctly maintains the key patterns, closed patterns, and supports of equivalence classes after multiple transactions are removed from the original database.*

Proof. According to Theorem 1, for any equivalence class $[P]_{\mathcal{D}_{org}}$ in $\mathcal{D}_{org}$, there are only 6 scenarios. We prove the correctness of the proposed algorithm according to these 6 scenarios.

For Scenario 1, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is not in $\mathcal{D}_{dec}$, and (iii) $f(P, \mathcal{D}_{org}) \neq f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{dec}$. Suppose $P \in O_i$. Points (i) and (ii) and Lemma 1 imply $O_i.closed$ is not in any transaction of $\mathcal{T}$. Thus Line 4 cannot hold on O_i. So $O'_i = O_i$, as initialized in Line 2 when we reach Line 8. Points (i) and (ii) imply $O'_i.sup = O_i.sup > ms$. Thus Line 10 cannot hold. Next, we consider those O'_j where $O'_j \neq O'_i$. Point (iii) implies that, for all O'_j, $O'_j.sup \neq O'_i.sup$. Thus Line 15 and Line 20 cannot hold for all O'_j. This proves Scenario 1.

For Scenario 2, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is not in $\mathcal{D}_{dec}$, and (iii) $f(P, \mathcal{D}_{org}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q in $\mathcal{D}_{dec}$. Let $O_i \in \mathcal{O}$ be the unique one equivalence class that contains P. Since $\mathcal{D}_{dec} \subset \mathcal{D}_{org}$, by Lemma 1, Points (i) and (ii) imply that $O_i.closed$ is not in $\mathcal{D}_{dec}$. Thus O_i does not satisfy Line 4 for all $T \in \mathcal{T}$. As a result, $O'_i.sup = O_i.sup$ and $O'_i.closed = O_i.closed$, as correctly initialized in Line 2, By an argument similar to Scenario 1, Points (i) and (ii) imply Line 10 cannot hold. Next we show that Lines 15 and 20 can hold. First observe that Points (ii) and (iii) imply that $O_i.sup = O'_i.sup = O'_j.sup$ for some $O_j \neq O_i$.

The situation where $j < i$, we should encounter O'_j before we encounter O'_i in our iteration. So $O'_j.closed = O_j.closed$ at the point of encounter, as initialized in Line 2. Then $O'_j.closed = O_j.closed \subset O_i.closed = O'_i.closed$ by Fact 7. Then Line 20 holds—note that as $j < i$, the role of "i" and "j" in Line 20 is reversed. Since we iterate all $O'_j \in \{O'_1, ..., O'_{i-1}\}$ if they exist, we find all O'_j, that are numbered in front of O'_i, that satisfy Line 20. Let $O'_{j_1}, ..., O'_{j_h}$ be such O'_js. Then Line 22—note that as $j < i$, the role of "i" and "j" in Line 22 is reversed—iteratively updates $[P]_{\mathcal{D}_{upd-}}.keys$ to $O_i^{\prime(j_1)}.keys$, ..., $O_i^{\prime(j_h)}.keys$, where $O_i^{\prime(j_q)}.keys = min\{k_{j_q} \mid k_{j_q} \in O_i^{\prime(j_{q-1})}.keys$ or $k_{j_q} \in O'_{j_q}.keys\}$, for $1 \leq q \leq h$. Also, $O_i^{\prime(j_0)} = [P]_{\mathcal{D}_{org}}$ as initialized in Line 2.

For the situation where $j > i$, we have $O'_j.closed = O_j.closed$ at this point, as initialized in Line 2. Then $O'_j.closed = O_j.closed \subset O_i.closed = O'_i.closed$ by Fact 7. Thus Line 15 holds. Since we iterate all $O'_j \in \{O'_{i+1}, ..., O'_n\}$ if they exist, we find all O'_j, that are numbered behind O'_i, that satisfy Line 15. Let $O'_{j_{h+1}}, ..., O'_{j_{h+g}}$ be such O'_js. Then Line 17 iteratively updates $[P]_{\mathcal{D}_{upd-}}.keys$ to $O_i^{\prime(j_{h+1})}.keys$, ..., $O_i^{\prime(j_{h+g})}.keys$,

where $O_i'^{(j_q)}.keys = min\{k_{j_q} \mid k_{j_q} \in O_i'^{(j_{q-1})}.keys$ or $k_{j_q} \in O'_{j_q}.keys\}$, for $h+1 \leq q \leq h+g$. Also, $O_i'^{(j_h)}$ is as last updated in Line 22, as described in the previous paragraph.

After unfolding this chain, it is clear that $[P]_{\mathcal{D}_{upd-}}.keys$ has been updated to $min\{k|k \in O'_i$ or $k \in O'_{j_1}$ or ... or $k \in O'_{j_{h+g}}\}$, which correctly corresponds to Scenario 2.

For Scenario 3, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is in $\mathcal{D}_{dec}$, and (iii) $|f(P, \mathcal{D}_{upd}-)| < ms$. Let C be the closed pattern of $[P]_{\mathcal{D}_{org}}$. According to Point (ii) and and Corollary 1, there are transactions $T \in \mathcal{T}$, such that $C \subset T$. Thus, Line 4 is satisfied, and the support of $[P]_{\mathcal{D}_{upd-}}$ is iteratively updated by Line 6. Point (iii) implies that $[P]_{\mathcal{D}_{upd-}}.sup < ms$. Then Line 10 succeeds and $[P]_{\mathcal{D}_{org}}$ is removed, correctly corresponding to Scenario 3, where the equivalence class disappears completely.

Scenario 4 is complementary to Scenario 2. So we do not repeat the proof here.

For Scenario 5, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is in $\mathcal{D}_{dec}$, (iii) $|f(P, \mathcal{D}_{upd}-)| > ms$, (iv) $f(Q, \mathcal{D}_{org}) \neq f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{org}$ and not in $\mathcal{D}_{dec}$, and (v) $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) \neq f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$. Let $O_i \in \mathcal{O}$ be the unique equivalence class that contains P. Similar to Scenario 3, Point (ii) and Corollary 1 imply there are transactions T from $\mathcal{T}$, such that $O_i.closed \subset T$. Thus, Line 4 is satisfied. Then $[P]_{\mathcal{D}_{upd-}}.sup$ is iteratively updated by Line 6. Eventually, $[P]_{\mathcal{D}_{upd-}}.sup =, [P]_{\mathcal{D}_{org}}.sup - [P]_{\mathcal{D}_{dec}}.sup$. Point (iii) then implies $[P]_{\mathcal{D}_{upd-}}.sup > ms$. So Line 10 cannot hold.

Let $O'_i = [P]_{\mathcal{D}_{upd-}}$. We now show that Lines 15 and 20 also cannot hold. Let $O_j \in \mathcal{O}$ be an equivalence class that contains a pattern Q, where (a) Q is in $\mathcal{D}_{org}$ and (b) Q is not in $\mathcal{D}_{dec}$. Points (a) and (b) and and Corollary 1 imply $O_j.closed$ is not in any transaction of $\mathcal{D}_{dec}$. Thus Line 4 cannot hold on O_j. So $O'_j = O_j$, as initialized in Line 2. Point (iv) implies that, for all Q satisfies Points (a) and (b), $f(Q, \mathcal{D}_{org}) = f(Q, \mathcal{D}_{upd-}) \neq f(P, \mathcal{D}_{upd-})$. Thus $P \notin O'_j$ and $O'_i \cap O'_j = \emptyset$. We then have two cases. In the first case, we have $O'_i.sup \neq O'_j.sup$. For this case, it is clear that Line 15 and 20 cannot hold. In the second case, we have $O'_i.sup = O'_j.sup$. In this case, $O'_i.sup = O'_j.sup$, $O'_i \cap O'_j = \emptyset$ and FACT 7 imply that $O'_j.closed \not\subset O'_i.closed$ and $O'_i.closed \not\subset O'_j.closed$. Line 15 and 20 cannot hold in this case neither.

Now, let $O_j \in \mathcal{O}$ be an equivalence class that contains a pattern Q', where (a) Q' in $\mathcal{D}_{org}$, (b) Q' in $\mathcal{D}_{dec}$, and (c) $Q' \notin [P]_{\mathcal{D}_{org}}$. Points (a) and (b) and Corollary 1 imply that there is $T \in \mathcal{T}$ such that $O_j.closed \subset T$. Thus Line 4 is satisfied. Let $O'_j = [Q']_{\mathcal{D}_{upd-}}$. Then $O'_j.sup$ is iteratively updated in Line 6. Point (v) implies that, for all Q' satisfies Points (a) (b) and (c), $f(P, \mathcal{D}_{upd-}) \neq f(Q', \mathcal{D}_{upd-})$. Thus $P \notin O'_j$ and $O'_i \cap O'_j = \emptyset$. With a similar argument in the previous paragraph, we can reach the conclusion that Line 15 and 20 cannot hold on such O'_js neither.

Moreover, it is clear that $O.closed$ either satisfies the definition of Q or the definition of Q', for any $O \in \mathcal{O}$ where $O \neq O_i$. Therefore, combining above two conclusions, Line 15 and 20 cannot hold, and the main loop is basically skipped. As a result, $[P]_{\mathcal{D}_{upd-}}$ remains the same as $[P]_{\mathcal{D}_{org}}$ except for a decrease in support, which correctly corresponds to Scenario 5.

For Scenario 6, suppose (i) P is frequent in $\mathcal{D}_{org}$, (ii) P is in $\mathcal{D}_{dec}$, (iii) $|f(P, \mathcal{D}_{upd}-)| > ms$, (iv) $f(Q, \mathcal{D}_{org}) \neq f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec})$ for all Q in $\mathcal{D}_{org}$ and not in $\mathcal{D}_{dec}$, and (v) $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$ for some Q in $\mathcal{D}_{dec}$ and $Q \notin [P]_{\mathcal{D}_{org}}$. Let $O_i \in \mathcal{O}$ be the unique one equivalence class containing P. Then, O'_i is then initialized in Line 2 as $O'_i = O_i$. Similar to Scenario 3, Point (ii) and Corollary 1 imply that $O'_i.sup$ is iteratively updated by Line 6. Point (iii) then implies that $O'_i.sup > ms$. Thus Line 10 cannot hold. As proven in Scenario 5, Point (iv)

implies that Line 15 and 20 cannot hold for any $O_j \in \mathcal{O}$ and $O'_j = O_j$ that contains Q, where Q is in $\mathcal{D}_{org}$ but not in $\mathcal{D}_{dec}$.

Now let $O_j \in \mathcal{O}$ be an equivalence class that contains a pattern Q', where (a) Q' in $\mathcal{D}_{org}$, (b) Q' in $\mathcal{D}_{dec}$, (c) $Q' \notin [P]_{\mathcal{D}_{org}}$, and (d) $f(P, \mathcal{D}_{org}) - f(P, \mathcal{D}_{dec}) = f(Q, \mathcal{D}_{org}) - f(Q, \mathcal{D}_{dec})$. Points (a) and (b) and Corollary 1 imply there is $T \in \mathcal{T}$ such that $O_j.closed \subset T$. Thus Line 4 is satisfied. Let $O'_j = [Q']_{\mathcal{D}_{upd-}}$. Then $O'_j.sup$ is iteratively updated in Line 6. Point (c) implies $O_j \neq O_i$. Point (d) implies O_j contains a Q' that witnesses Point (v). Specifically, we conclude (e) $O'_i.sup = O'_j.sup$ and (f) either $O'_i.closed \subset O'_j.closed$ or $O'_j.closed \subset O'_i.closed$, for $Q' \in [P]_{\mathcal{D}_{upd-}}$.

Assume O'_{j_1}, ..., O'_{j_g}, ..., $O'_{j_{g+h}}$, where $j_g < i < j_{g+1}$, are all the equivalence classes in $\mathcal{O}$ that contain some Q's as described in Points (a) to (d) above. Now we proceed by a case analysis. Due to the constrains introduced by per Conclusion (f), we have four cases based on the relationship between $O'_i.closed$ and $O'_{j_q}.closed$, where $1 \leq q \leq g+h$.

In the first case, we have $O'_i.closed \subset O'_{j_q}.closed$, for some O'_{j_q}, where $1 \leq q \leq g$. In this case, $j_q < i$. Thus O'_{j_q} is encountered before O'_i in our iteration. Moreover, as per Conclusion (e), $O'_{j_q}.sup = O'_i.sup$ and $O'_i.closed \subset O'_{j_q}.closed$. Then Line 15 holds—note that as $j_q < i$, the role of "i" and "j" in Line 15 are reversed. Then, O'_i is removed by Line 18—note that as $j_q < i$, the role of "i" and "j" are reversed. This means we will not encounter O'_i later in our iteration. Therefore, for all encountered O'_i, this case is definitely invalid.

In the second case, we have $O'_i.closed \supset O'_{j_q}.closed$, for all O'_{j_q}, where $1 \leq q \leq g$, and $O'_i.closed \subset O'_{j_{g+1}}.closed$,. For all O'_{j_q}, where $1 \leq q \leq g$, since $j_q < i$, O'_{j_q} is encountered before O'_i in our iteration. Moreover, as per Conclusion (e), $O'_{j_q}.sup = O'_i.sup$ and $O'_i.closed \supset O'_{j_q}.closed$. Thus Line 20 holds — note that as $j_q < i$, the roles of "i" and "j" in Line 20 are reversed. Since we iterate all $O'_j \in \{O'_1, ..., O'_{i-1}\}$ if they exist, we find all O'_{j_q}, $1 \leq q \leq g$. Then, $[P]_{\mathcal{D}_{upd-}}.keys$ is iteratively updated to $[P]_{\mathcal{D}_{upd-}}.keys^{(i)}$, where $[P]_{\mathcal{D}_{upd-}}.keys^{(i)} = min\{k | k \in O'_i$ or $k \in O'_{j_1}$ or ... or $k \in O'_{j_g}\}$, as described in the $j < i$ case of Scenario 2. The proof will not be repeated here. For $O'_{j_{g+1}}$, Conclusion (e) and $O'_i.closed \subset O'_{j_{g+1}}.closed$ imply that Line 20 holds for O'_i. Thus Line 22 further updates $[P]_{\mathcal{D}_{upd-}}.keys$ to $[P]_{\mathcal{D}_{upd-}}.keys^{(j_{g+1})}$, where $[P]_{\mathcal{D}_{upd-}}.keys^{(j_{g+1})} = min\{k | k \in [P]_{\mathcal{D}_{upd-}}.keys^{(i)}$ *or* $k \in O'_{j_{g+1}}.keys\}$. Then, O'_i is removed by Line 23. $[P]_{\mathcal{D}_{upd-}}.keys$ continues to be iteratively updated, in a similar manner as Scenario 2. Eventually, $[P]_{\mathcal{D}_{upd-}}.keys = min\{k | k \in O'_i$ or $k \in O'_{j_1}$ or ... or $k \in O'_{j_{g+h}}\}$, which correctly corresponds to Scenario 6.

In the third case, we have $O'_i.closed \supset O'_{j_q}.closed$, for for all O'_{j_q}, where $1 \leq q \leq v$ & $g \leq v \leq g+h$, and $O'_i.closed \subset O'_{j_{v+1}}.closed$. For all O'_{j_q}, where $1 \leq q \leq g$, $[P]_{\mathcal{D}_{upd-}}.keys$ is updated to $[P]_{\mathcal{D}_{upd-}}.keys^{(i)}$ in the same manner as discussed in the previous case. For those O'_{j_q}, where $g+1 \leq q \leq v$ and $g \leq v \leq g+h$, $O'_i.closed \supset O'_{j_q}.closed$ and Conclusion (e) imply that Line 15 holds. $[P]_{\mathcal{D}_{upd-}}.keys$ is further updated to $[P]_{\mathcal{D}_{upd-}}.keys^{(j_v)}$ in the similar manner described in the $j > i$ case of Scenario 2. Here, $[P]_{\mathcal{D}_{upd-}}.keys^{(j_v)} = min\{k | k \in O'_i$ or $k \in O'_{j_1}$ or ... or $k \in O'_{j_v}\}$. We then iterate to $O'_{j_{v+1}}$. Conclusion (e) and $O'_i.closed \subset O'_{j_{g+1}}.closed$ imply that Line 20 holds for O'_i. Thus, Line 22 further update $[P]_{\mathcal{D}_{upd-}}.keys$ to $[P]_{\mathcal{D}_{upd-}}.keys^{(j_{v+1})}$, where $[P]_{\mathcal{D}_{upd-}}.keys^{(j_{v+1})} = min\{k | k \in [P]_{\mathcal{D}_{upd-}}.keys^{(j_v)}$ *or* $k \in O'_{j_{v+1}}.keys\}$. Then, O'_i is removed by Line 23. $[P]_{\mathcal{D}_{upd-}}.keys$ continues to be iteratively updated, in a similar manner as Scenario 2. Eventually, $[P]_{\mathcal{D}_{upd-}}.keys = min\{k | k \in O'_i$ or $k \in O'_{j_1}$ or ... or $k \in O'_{j_{g+h}}\}$, which correctly corresponds to Scenario 6.

In the last case, we have $O'_i.closed \supset O'_{j_q}.closed$, where $1 \leq q \leq g + h$. This case corresponds to the situation where $[P]_{\mathcal{D}_{upd-}}.closed = O'_i.closed$. In this case, $[P]_{\mathcal{D}_{upd-}}.keys$ is updated in exactly the same manner as in Scenario 2. This completes the proof.