An Incremental Technique for Mining Coverage Patterns in Large Databases

by

ralla akhil, P Krishna Reddy, Anirban Mondal

in

IEEE DSAA 2019 The 6th IEEE International Conference on Data Science and Advanced Analytics

Report No: IIIT/TR/2019/-1

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2019

An Incremental Technique for Mining Coverage Patterns in Large Databases

Akhil Ralla
International Institute of Information Technology, Hyderabad, India
ralla.akhil@research.iiit.ac.in

P. Krishna Reddy
International Institute of Information Technology, Hyderabad, India
pkreddy@iiit.ac.in

Anirban Mondal
Ashoka University
Haryana, India
anirban.mondal@ashoka.edu.in

***Abstract*—Pattern mining is an important task of data mining and involves the extraction of interesting associations from large databases. Typically, pattern mining is carried out from huge databases, which tend to get updated several times. Consequently, as a given database is updated, some of the patterns discovered may become *invalid*, while some new patterns may emerge. This has motivated significant research efforts in the area of *Incremental Mining*. The goal of incremental mining is to *efficiently* and *incrementally* mine patterns when a database is updated as opposed to mining all of the patterns from scratch from the complete database. Incidentally, research efforts are being made to develop incremental pattern mining algorithms for extracting different kinds of patterns such as frequent patterns, sequential patterns and utility patterns. However, *none* of the existing works addresses incremental mining in the context of coverage patterns, which has important applications in areas such as banner advertising, search engine advertising and graph mining. In this regard, the main contributions of this work are three-fold. First, we introduce the problem of incremental mining in the context of coverage patterns. Second, we propose the IncCMine algorithm for efficiently extracting the knowledge of coverage patterns when incremental database is added to the existing database. Third, we performed extensive experiments using two real-world click stream datasets and one synthetic dataset. The results of our performance evaluation demonstrate that our proposed IncCMine algorithm indeed improves the performance significantly w.r.t. the existing CMine algorithm.**

***Index Terms*—Data mining, Coverage patterns, Incremental mining, Knowledge discovery**

I. Introduction

Data mining approaches extract knowledge in the form of significant patterns (frequent patterns and association rules), clusters, and classes from large databases. The task of pattern mining involves the extraction of interesting associations from large databases [2], [23]. Pattern mining has several important and diverse applications in areas such as market basket analysis, recommendation systems and internet advertising.

Notably, in pattern mining based applications, database sizes are typically huge and databases tend to get updated over the course of time. Understandably, the knowledge discovered from a given database may become *invalid* when more data is added to the existing database. As the database is updated or incremented, some existing knowledge of patterns may become invalid and some new knowledge of patterns may also emerge. The naïve approach for knowledge extraction when an increment database is added to a given database would be to run a pattern mining algorithm from scratch on the complete database. However, we can intuitively understand that the naïve approach suffers from the drawback of high computational overhead.

As an alternative option, there is an opportunity to devise efficient algorithms by validating the patterns of the existing database and extracting the additional patterns that arise due to the incremental database. In the literature, this area is called *Incremental Mining*. Given the knowledge of patterns from a database DB and an increment database db, the Incremental Mining problem is to develop efficient pattern mining approaches for generating pattern knowledge from the complete database, which comprises both DB and db. Developing efficient incremental pattern mining approaches is an active research area. Hence, several incremental approaches have been proposed to extract frequent patterns [6], [13], [21], sequential patterns [16], [18], [26], utility patterns [4], [19], [27]. Observe that the number of new candidate patterns is likely to be significantly lower than that of the number of candidate patterns, which would have been generated by considering the complete database (i.e., both DB and db). Hence, there is an opportunity to improve the performance of pattern extraction by using an incremental approach.

Another useful type of pattern is the coverage pattern [23], [24], which has important applications in areas such as banner advertising, search engine advertising and graph mining. In the literature, a model to extract the knowledge of coverage patterns (CPs) from transactional databases has been proposed in [23], [24]. Given a transactional database and a set of data items, CP is a set of items covering a certain percentage of transactions by minimizing overlap among the transactions covered by each item of the pattern. A level-wise CP mining algorithm, designated as CMine [23], and a pattern growth approach called CPPG [24] were proposed to extract CPs from large databases. It has been demonstrated that CPs extracted from transactional log data can be employed to improve the performance of display advertising [15], [25] and search engine advertising [7]–[9]. Incidentally, the problem of incremental mining for coverage patterns has not been investigated in the literature. To our knowledge, this is the first work to study incremental mining for coverage patterns.

In this paper, given the CPs of a database DB, we propose IncCMine, which is an incremental variation of the

CMine algorithm [23] for extracting CPs when an incremental database db is added to DB. When db is added to DB, some of the existing CPs of DB may become invalid and some new CPs may emerge. The existing patterns in DB are validated by scanning db. We generate candidate patterns in a level-wise manner by scanning db and validating the new patterns in DB. Let $Flist$ be the list of items sorted based on their frequencies in the transactional database. Under our proposed IncCMine approach, we also need to validate the additional patterns, which would be generated if the $Flist$ of (DB+db) is different from that of DB. Notably, our proposed IncCMine algorithm can potentially improve the performance w.r.t. the CMine algorithm because we are processing only a relatively few candidate patterns that are extracted from db and the relatively low number of additional patterns that are generated due to changes in $Flist$.

The main contributions of this work are three-fold:

1) We introduce the problem of incremental mining in the context of coverage patterns.
2) We propose the IncCMine algorithm for efficiently extracting the knowledge of coverage patterns when incremental database is added to the existing database.
3) We performed extensive experiments using two real-world click stream datasets and one synthetic dataset. The results of our performance evaluation demonstrate that our proposed IncCMine algorithm indeed improves the performance significantly w.r.t. the existing CMine algorithm.

The remainder of this paper is organized as follows. In Section II, we discuss the related work and background information about coverage patterns. In Section III, we discuss the context of the problem. In Section IV, we discuss the proposed approach. In Section V, we report the performance evaluation. Finally, we conclude in Section VI with directions for future work.

II. Related Work and Background

This section discusses existing works as well as background information about coverage patterns.

A. Related work

In the literature, mining of incremental databases was first studied in the context of frequent patterns [13] by means of the FUP algorithm. The work in [12] extended the FUP algorithm to additionally consider the deletion of transactions. Additionally, the *Borders algorithm* [5] is able to generate associations incrementally for dynamic databases. The work in [6] discussed a single-pass tree-based approach for extracting rare association rules from incremental databases. Moreover, in [21], an intermediate structure, designated as the *constructive set* is used for producing closed itemsets. Furthermore, the IMU2P-Miner algorithm [14] is a tree-based incremental algorithm that is used for extracting maximal frequent patterns in incremental databases. IMU2P-Miner extracts patterns by just adding and updating paths in the tree.

Incremental mining has also been studied in the context of sequential patterns and utility patterns. The iterative ISE algorithm [18] extracts sequential patterns in incremental databases. Further improvements in efficiency were performed by means of the IncSpan algorithm [11], [20]. The IncSpan algorithm improves efficiency by maintaining almost frequent sequences and also by using optimization techniques. IncUSP-Miner [26] is a utility sequential pattern mining algorithm, which uses a candidate pattern tree data structure. The work in [16] proposed a pattern growth type BSpan algorithm and an apriori type BSPinc algorithm for mining sequential patterns. BSpan aims at minimizing database projections. BSPinc generates candidate sequences by using backward extensions and also by mining patterns recursively.

The work in [4] discussed three types of tree structures, namely the IHUP lexicographic tree, the IHUP transaction frequency tree and the HUP transaction weighted utilization tree. These tree structures are used for mining high utility patterns in incremental databases. Moreover, the one-pass IIHUM algorithm [27] uses efficient list-based data structures, which work without generating candidates. Furthermore, the MEFIM algorithm [19] employs a novel structure, designated as *P-set*, for reducing the number of transaction scans and to speed up the pattern mining process.

Notably, while incremental has been studied in the context of frequent patterns [12], sequential patterns [11], utility patterns [4], rare patterns [6], none of the existing works has investigated incremental mining in the context of coverage patterns.

B. About Coverage Patterns

1) Model of coverage patterns: Let $I = \{i_1, i_2.....,i_n\}$ be the set of items. Database D contains transactions, where each transaction T comprises a set of items, i.e., $T \subseteq I$. $|D|$ represents the total number of transactions in database D. T^{i_p} represents the set of transactions, which contain the item i_p. $|T^{i_p}|$ represents the number of transactions containing i_p.

Coverage patterns incorporate the following notions: relative frequency, coverage set, coverage support and overlap ratio.

Definition 1. Relative frequency of item i_p. *The fraction of transactions containing a item i_p is called the Relative Frequency (RF) of i_p and is computed as RF(i_p) =* $\frac{|T^{i_p}|}{|D|}$.

An item is considered to be frequent if its $RF \geq minRF$, where $minRF$ is a user-specified threshold.

Definition 2. Coverage set CSet(X) of a pattern X. *Given a pattern $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$, CSet(X) is the set of all transactions that contains at least one item of pattern X, i.e., $CSet(X) = T^{i_p} \cup T^{i_q} \cup ...T^{i_r}$.*

Definition 3. Coverage support CS(X) of a pattern X. *Given $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$, CS(X) is the ratio of the size of $CSet(X)$ to $|D|$ i.e., $CS(X) = \frac{|CSet(X)|}{|D|}$*

Definition 4. Overlap ratio OR(X) of a pattern X. *Given $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$ and $|T^{i_p}| \geq ... \geq$*

$|T^{i_q}| \geq |T^{i_r}|$, *OR(X) is the ratio of the number of common transactions between* $CSet(X - i_r)$ *and* T^{i_r} *to the number of transactions having item* i_r*, i.e.,* $OR(X) = \frac{|CSet(X-i_r) \cap T^{i_r}|}{|T^{i_r}|}$

The intuition of overlap ratio is as follows. Suppose we want to increase the coverage by adding a new item i_k to the pattern X. Let the coverage of item i_k be $CSet(i_k)$. Notably, adding i_k to X could result in overlap between $CSet(X)$ and $CSet(i_k)$. If the overlap is high, $CSet(X \cup i_k)$ may not increase significantly over $CSet(X)$. Hence, in such a case, adding i_k to X would not be interesting. In essence, adding a new item i_k to set X would be interesting only if there is a minimal overlap. So, a pattern is interesting if it has high CS and less OR, where high CS indicates more coverage of transactions and less OR indicates less overlap among the transactions covered by the items.

Definition 5. Coverage pattern (CP). *A pattern* $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$ *and* $|T^{i_p}| \geq ... \geq |T^{i_q}| \geq |T^{i_r}|$ *is called coverage pattern if* $OR(X) \leq maxOR$, $CS(X) \geq minCS$ *and* $RF(i_k) \geq minRF$ $\forall i_k \in X$, *where* $minCS$, $maxOR$ *are user-specified minimum coverage support and maximum overlap ratio respectively.*

Given a set of items *I*, transactional database D, $minRF$, $minCS$ and $maxOR$, the problem of mining CPs is to discover the complete set of CPs.

2) Sorted closure property: The overlap ratio satisfies downward closure property if items are ordered in descending order of their frequencies. Such a property is called the sorted closure property [17].

Sorted closure property. Let $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$ be a pattern such that $|T^{i_p}| \geq ... \geq |T^{i_q}| \geq |T^{i_r}|$. If OR(X) $\leq maxOR$, all of the non-empty subsets of X containing i_r and having $size \geq 2$ will also have overlap ratio less than or equal to $maxOR$.

An item a is said to be a non-overlap item w.r.t. a pattern X if $OR(X, a) \leq maxOR$ and $RF(i_k) \geq minRF$ $\forall i_k \in \{X, a\}$. The notion of non-overlap pattern (NOP) is defined as follows.

Definition 6. *Non-overlap pattern (NOP): A pattern* $X = \{i_p, ..., i_q, i_r\}$, $(1 \leq p, q, r \leq n)$ *and* $|T^{i_p}| \geq ... \geq |T^{i_q}| \geq |T^{i_r}|$ *is called non-overlap pattern if* $OR(X) \leq maxOR$ *and* $RF(i_k) \geq minRF$ $\forall i_k \in X$.

3) CMine algorithm: Similar to the apriori algorithm [2], CMine is an iterative multi-pass algorithm [23] for extracting CPs from a given transactional database. Apriori algorithm uses frequent patterns of size k to explore size patterns of size $k+1$. As frequent patterns satisfy downward closure property, if an itemset does not the satisfy minimum support constraint, all of its supersets will be pruned.

In case of CMine also, $NOPs$ of size k are used to explore size $(k + 1)$. As $NOPs$ satisfy sorted closure property, we extract $NOPs$, which satisfy the $maxOR$ constraint. Next, CPs are extracted by applying the $minCS$ constraint. As $NOPs$ satisfy the sorted closure property, if an itemset does not satisfy the $maxOR$ constraint, all of its supersets (where each superset has item frequencies in the corresponding sorted order) would also be pruned (refer to sorted closure property).

The main aspect is to extract $NOPs$. Once we extract $NOPs$ based on the $maxOR$ constraint, extraction of CPs is relatively simple. Hence from this point onwards, we shall present the CMine algorithm and the proposed IncCMine algorithm for the extraction of $NOPs$.

Let C_k, NOP_k, CP_k denote the candidate, non-overlap, coverage patterns of size k respectively. Given $minCS$, $maxOR$, and $minRF$ values, the steps of CMine algorithm to extract CPs from the transactional database D are as follows:

1) First iteration: The frequency of each item is computed by scanning D. After scanning, CP_1 and NOP_1 are computed by checking relative frequency. Item is added to NOP_1 if $RF \geq minRF$, and added to CP_1 if $RF \geq minCS$. The items in NOP_1 are sorted in descending order of their frequencies into a list, which we designate as $Flist$.
2) Second iteration and beyond. Starting from k=2, the following step is repeated until $C_k = \phi$. C_k is generated by computing $NO_{k-1} \bowtie NO_{k-1}$ (self-join). After scanning D, NOP_k and CP_k are computed by checking OR and CS of patterns in C_k accordingly.

TABLE I: Summary of notations

Notation	Definition
DB	Original transactional database
db	Increment transactional database
CDB	Complete transactional database $CDB = DB + db$
RF	Relative frequency
OR	Overlap ratio
CS	Coverage support
$minRF$	Minimum relative frequency
$maxOR$	Maximum overlap ratio
$minCS$	Minimum coverage support
CP	Coverage patterns
NOP	Non-overlap pattern
NOP_D	Set of non-overlap patterns of database *D*
CP_D	Set of coverage patterns of database *D*
C_k	Set of candidate patterns of size k
$NOP_{(D,k)}$	Set of non-overlap patterns of size k in database *D*
$CP_{(D,k)}$	Set of coverage patterns of size k of database *D*
$OR(X_D)$	Overlap ratio of pattern *X* in database *D*
$CS(X_D)$	Coverage support of pattern *X* in database *D*
C_k	Set of size k candidate patterns
APF	Set of additional patterns due to change in $Flist$
Ψ	Percentage of additional patterns due to change in $Flist$

III. Context of the problem

Let $I = \{i_1, i_2....., i_n\}$ be the set of items. Let DB and db be the set of transactions representing the original and increment databases respectively, where each transaction T is a set of items such that $T \subseteq I$. Let the complete database be CDB, i.e., CDB = $DB + db$. Suppose a set of $NOPs$ are extracted from DB at the user-specified threshold values of $minRF$ and $maxOR$. Given a set of $NOPs$ of DB, the problem is to generate $NOPs$ from CDB (Notably, once we

extract $NOPs$, CPs are extracted based on CS threshold). Table I depicts the various notations used in the paper.

Now we explain the properties, which are being identified to develop the proposed IncCMine algorithm.

Lemma 1. *A 1-itemset X, which is NOP in DB, is NOP in CDB, if RF(X) $\geq minRF$ in CDB.*

Proof. According to the definition of RF and NOP.

Notably, RF in CDB is equal to RF in ($DB + db$). As per Lemma 1, we can identify potential $NOPs$ in CDB by measuring RF in db as RF is available from DB.

Lemma 2. *A 1-itemset X, which is not a NOP in DB, is NOP in CDB, if it is NOP in db, i.e., RF(X_{db}) $\geq minRF$.*

Proof. Let $F(X_D)$ represent the frequency of itemset X in D. Since X is not NOP in DB, $F(X_{DB}) \leq minRF * |DB|$. If $F(X_{db}) \leq minRF * |db|$, $F(X_{CDB}) \leq minRF * |CDB|$ since $F(X_{CDB}) = F(X_{DB}) + F(X_{db})$.

As per Lemma 2, if there is a new NOP in CDB, it should exist in db. So, by extracting $NOPs$ from db, it is ensured that all new $NOPs$ in CDB are extracted.

The following lemma is defined in the context of level-wise pruning.

Lemma 3. *If a $(k-1)$ itemset $X = \{i_1, ..., i_{k-1}\}$, is a loser at $(k-1)^{th}$ iteration, any set Y s.t. $X \subseteq Y$ is not an NOP in CDB.*

Proof. According to the sorted closure property of $NOPs$.

Lemma 3 helps to prune the non-potential NOP patterns in a level-wise manner.

Lemma 4. *A k-itemset $X = \{i_1, ...i_{k-1}, i_k\}$, which is not a NOP in DB and not an NOP in db, will not be NOP in CDB, i.e., $DB + db$.*

Proof. Let $n(OR(X_D))$ and $d(OR(X_D))$ represent the numerator and the denominator of the overlap ratio of pattern X in database D, where $n(OR(X_D)) = |CSet(X - i_r) \cap T^{w_r}|$ and $d(OR(X_D)) = |CSet(i_r)|$. It is given that the pattern is not NOP in both DB and db. Hence, we have the following equations:

$$OR(X_{DB}) = \frac{n(OR(X_{DB}))}{d(OR(X_{DB}))} > maxOR \tag{1}$$

$$OR(X_{db}) = \frac{n(OR(X_{db}))}{d(OR(X_{db}))} > maxOR. \tag{2}$$

From Equations 1 and 2, we obtain:

$$\frac{n(OR(X_{DB})) + n(OR(X_{db})}{d(OR(X_{DB})) + d(OR(X_{db}))} > maxOR \tag{3}$$

Equation 3 shows OR of X in CDB is greater than maxOR. Thus, if a pattern is not MOP in DB and db, pattern is not MOP in CDB.

Lemma 4 helps to prune non-potential new $NOPs$ by processing db.

IV. Proposed approach

This section discusses the proposed approach.

A. Basic Idea

CMine is an iterative algorithm for extracting the CPs of a given transactional database. Recall that CMine exploits the sorted closure property of the overlap ratio (OR). In CMine, the non-overlap patterns ($NOPs$) in each iteration are computed and then the CPs are generated based on the coverage support (CS) of $NOPs$.

The basic idea of the proposed IncCMine approach is as follows.

Given $minRF$, $minCS$, and $maxOR$ (see Section 2), suppose NOP_{CDB} is the complete set of $NOPs$ extracted from CDB. For each pattern $P \in NOP_{CDB}$, the following four possibilities exist w.r.t. its validity in DB and db:

1) $P \in NOP_{DB}$ and $P \in NOP_{db}$.
2) $P \in NOP_{DB}$ and $P \notin NOP_{db}$.
3) $P \notin NOP_{DB}$ and $P \in NOP_{db}$.
4) $P \notin NOP_{DB}$ and $P \notin NOP_{db}$.

In Case 1, we extract $NOPs$ in db and add the counts of these $NOPs$ to the corresponding $NOPs$ in DB. In Case 2, only the patterns in DB are validated by adding the counts of corresponding $NOPs$ in db. In Case 3, we extract additional patterns from db. We observe that there can be two kinds of additional patterns, namely (a) additional new patterns that exist in db and (b) additional patterns due to change in the sorting order of items in $Flist$ (recall Section 2). The additional patterns that exist in db are extracted by processing db. We designate the additional patterns due to changes in Flist as APF. The APF are extracted by counting item frequencies of CDB and forming the $Flist_{CDB}$ in the sorted order. If $Flist_{CDB}$ differs from $Flist_{DB}$, the new patterns are generated based on $Flist_{CDB}$. In Case 4, no action needs to be taken since such a possibility does not exist (see Lemma 4).

Considering $APFs$ in Case 3, we compute the percentage Ψ of APF. Let Ψ_{max} be a user-specified threshold. If $\Psi < \Psi_{max}$, we validate the corresponding $APFs$ in $DB + db$. We defer the discussion concerning how to set the value of Ψ_{max} to Section 5 (E).

In the next subsection, we discuss the estimation of Ψ. Next, we present the proposed IncCMine algorithm followed by an illustrative example.

B. Estimation of Ψ

Now we shall present an approach to estimate the value of the upper-bound of the number of $APFs$. We compute Ψ by normalization i.e., $\frac{|APF|}{max(|APF|)} * 100$, where $max(|APF|)$ is the maximum value of $|APF|$. The range of Ψ is 0 to 100. We shall shortly see how to determine $max(|APF|)$.

Now we shall define the notion of *prefix set*, which is used in the computation of $|APF|$.

Definition 7. ***Prefix set**: The prefix set $P_D(i_p)$ of an item i_p in database D is the set of items, whose frequency is greater*

than that of its own frequency i.e., the set of items that come before i_p in the Flist.

In computing the value of $|APF|$, we take the $Flist$ of DB and $Flist$ of CDB as inputs. Let $Flist_{DB}$ ($Flist$ of DB) and $Flist_{CDB}$ ($Flist$ of CDB) be $\{i'_1, i'_2, .., i'_m\}$ and $\{i_1, i_2, ..., i_n\}$ respectively. Consider a pattern X in which i_p is the least frequent item. We define $commset(i_p)$ as the set of items, which are present in both $P_{CDB}(i_p)$ and $P_{DB}(i_p)$ i.e., $P_{CDB}(i_p) \cap P_{DB}(i_p)$. The patterns having items in $commset$ along with i_p are not additional patterns as those have already been checked in extracting $NOPs$ of DB. The number of additional patterns having i_p as the least frequent item is $(2^{|P_{CDB}(i_p)|} - 2^{|commset|})$. The value of $|APF|$ is more influenced by the least frequent item as the size of the prefix set of the least frequent item is high. The value of $|APF|$ is zero when both the $Flists$ of DB and CDB are the same. Notably, $|APF|$ does not always attain the upper-bound in the iterative approach. This occurs because some additional patterns of size k are not generated as its subsets of size $(k-1)$ are not $NOPs$. Thus, we estimate this value of $|APF|$ by multiplying it with $maxOR$. The estimated value of $|APF|$ is computed using Equation 4 as follows:

$$|APF| = maxOR * \Big(\sum_{i_p \in Flist_{CDB}} 2^{|P_{CDB}(i_p)|} - 2^{|commset(i_p)|} \Big) \tag{4}$$

$|APF|$ is maximum when $maxOR$ is 1 and the $Flists$ of DB and CDB are disjoint sets. The value of $max(|APF|)$ is equal to $2^N - N - 1 \approx 2^N$, where N is the size of $Flist_{CDB}$. The value of Ψ is computed using Equation 5 as follows:

$$\Psi = \frac{|APF|}{max(|APF|)} \approx \Big(\frac{|APF|}{2^N} \Big) * 100 \tag{5}$$

C. IncCMine Algorithm

Similar to the CMine algorithm, our proposed IncCMine algorithm is also an iterative algorithm. At the k^{th} iteration of the IncCMine algorithm, we compute $NOPs$ of size k by performing a single scan of DB and *db*. The inputs to the algorithm consist of DB, *db*, $minRF$, $minCS$, $maxOR$ and NOP_{DB}. We assume that the parameters $maxOR$, $minRF$ and $minCS$ remain unchanged with the increment in database. Now we shall explain the IncCMine algorithm.

(i) Iteration 1: Generation of non-overlap and coverage patterns of size one

The steps for extracting $NOP_{(CDB,1)}$ and $CP_{(CDB,1)}$ are as follows:

1) The relative frequencies of patterns in $NOP_{(DB,1)}$ are computed by performing one scan of the *db*. After scanning, the patterns having RF greater than $minRF$ (see Lemma 1) are added to $NOP_{(CDB,1)}$.
2) In the same scan of *db*, RF of candidate items (not in $NOP_{(DB,1)}$) are computed in *db*. The candidate items, whose RF is less than $minRF$, are pruned as they cannot be in $NOP_{(CDB,1)}$, according to Lemma 2.
3) One scan of DB is performed to compute the RF of potential candidate itemsets of size one. After scanning DB, the patterns having RF greater than $minRF$ are added to $NOP_{(CDB,1)}$.
4) After computing $NOP_{(CDB,1)}$, we compute $Flist_{CDB}$ by sorting the items in descending order of their frequencies. Next, we compute Ψ using the $Flists$ of DB and CDB.

(ii) Second iteration and beyond: Generation of non-overlap and coverage patterns of size k

Starting from $k = 2$, the following steps are repeated until $NOP_{(CDB,k)} = \phi$. The steps for extracting $NOP_{(CDB,k)}$ and $CP_{(CDB,k)}$ are as follows:

1) The OR of the patterns in $NOP_{(DB,k)}$ are computed by performing one scan of *db*. Before scanning *db*, the patterns in $NOP_{(DB,k)}$ can be pruned as follows. Based on Lemma 3, we compute the set *pp* of pruned patterns in the $(k-1)^{th}$ iteration by subtracting $NOP_{(CDB,k-1)}$ from $NOP_{(DB,k-1)}$. Then an itemset $X \in NOP_{(DB,k)}$ is pruned if $Y \in pp$, where $Y \subseteq X$. Additionally, if there is a change in the $Flist$, the patterns in $NOP_{(DB,k)}$, which do not follow the order of items in $Flist_{CDB}$, will be pruned. After computing OR, the patterns satisfying the $maxOR$ constraint are added to the set $NOP_{(CDB,k)}$.
2) The candidate patterns of size k are extracted from *db*. To extract new possible knowledge of patterns from *db*, the candidate patterns of size 2 are generated. The candidate set contains patterns generated from $NOP_{(CDB,k-1)}$ and not in $NOP_{(DB,k)}$. The OR values of these candidate patterns in *db* are computed while scanning *db*. After scanning *db*, the patterns that fail to satisfy the $maxOR$ constraint will be pruned, according to Lemma 4. Notably, if $\Psi > 0$, the candidate patterns of *db*, which belong to APF and which fail to satisfy $maxOR$ constraint in *db*, cannot be pruned because they do not follow the order of the items in $Flist$ of DB.
3) We compute OR of the remaining candidate patterns by scanning DB. All candidate patterns having value of OR less than that of $maxOR$ are added to the set $NOP_{(CDB,k)}$.

In the IncCMine algorithm, the number of candidate patterns validated by scanning DB is significantly lower w.r.t. the number of candidate patterns in the CMine algorithm. A much smaller candidate set gives the IncCMine algorithm an improvement in performance when compared with the CMine algorithm.

Algorithms 1 and 2 depict the first and k^{th} iterations of our proposed IncCMine algorithm respectively. The inputs to Algorithm 1 are $NOP_{(DB,1)}$, DB, *db*, $minRF$, $minCS$ and the outputs are $NOP_{(CDB,1)}$, $CP_{(CDB,1)}$ and $Flist_{CDB}$. First, we compute RF of items of size one in $NOP_{(DB,1)}$ by performing one scan of *db* (Line 1). In the same scan of *db*, we compute RF of C_1 in *db* (Line 2). $NOP_{(CDB,1)}$ and $CP_{(CDB,1)}$ are updated by checking RF of patterns in $NOP_{(DB,1)}$ according to Lemma 1 (Line 3). Some of the candidate patterns are pruned before scanning DB by using

Algorithm 1: First iteration of IncCMine algorithm

Input : $NOP_{(DB,1)}$: set of $NOPs$ of size one in DB; DB: original transactional database; db: increment transactional database; $minRF$; $minCS$

Output: $NOP_{(CDB,1)}$ = set of $NOPs$ of size one in CDB; $CP_{(CDB,1)}$ = set of CPs of size one in CDB; $Flist_{CDB}$

1 Compute RF of items in $NOP_{(DB,1)}$ in CDB by scanning db
2 In the same scan of db, RF of items in C_1 in db is computed
3 Items in NOP_1 are added to $NOP_{(CDB,1)}$, $CP_{(CDB,1)}$ by checking RF accordingly
4 Prune candidate patterns
5 Compute RF of C_1 in CDB by scanning DB
6 Items in C_1 are added to $NOP_{(CDB,1)}$, $CP_{(CDB,1)}$ accordingly
7 Compute $Flist_{CDB}$ by sorting items of $NOP_{CDB,1}$ in descending order of their frequencies

Algorithm 2: k^{th} iteration of IncCMine algorithm

Input : $NOP_{(CDB,(k-1))}$: set of $NOPs$ of size $(k-1)$ in CDB; DB: original database; $NOP_{(DB,k)}$: set of $NOPs$ of size k of DB; db: increment database; $minCS$; $maxOR$; $Flist_{CDB}$; $Flist_{DB}$

Output: $NOP_{(CDB,k)}$ = set of non-overlap patterns of size k in CDB; $CP_{(CDB,k)}$ = set of coverage patterns of size k in CDB

1 Prune patterns in $NOP_{(DB,k)}$
2 **if** $\Psi > 0$ **then**
3 Prune patterns in $NOP_{(DB,k)}$ if items do not follow $Flist_{CDB}$
4 C_k = $(NOP_{(CDB,(k-1))} \bowtie NOP_{(CDB,(k-1))})$-$NOP_{(DB,k)}$
5 Compute OR and CS of $NOP_{(DB,k)}$ in CDB by scanning db
6 In the same scan of db, CS and OR of C_k in db is computed
7 Items in $NOP_{(DB,k)}$ are added to $NOP_{(CDB,k)}$, $CP_{(CDB,k)}$ accordingly
8 **if** $\Psi = 0$ **then**
9 Pattern is pruned if $OR > maxOR$
10 **else if** $\Psi > 0$ **then**
11 Pattern is pruned if $OR > maxOR$ (Lemma 4) and follows the order of item in $Flist_{DB}$
12 Compute OR and CS of C_k in CDB by scanning DB
13 Items in C_k are added to $NOP_{(CDB,k)}$, $CP_{(CDB,k)}$ accordingly

Lemma 2 (Line 4). After pruning is done, RF of patterns in C_1 is computed by scanning DB. Now, $NOP_{(CDB,1)}$ and $CP_{(CDB,1)}$ are updated by checking RF of patterns in C_1 (Lines 5, 6). Finally, after computing $NOP_{(CDB,1)}$, $Flist$ is computed by sorting items in descending order of their frequencies (Line 7).

The pseudo-code for the k^{th} iteration of our proposed IncCMine algorithm is explained in Algorithm 2. The inputs to the Algorithm 2 are $NOP_{(CDB,(k-1))}$, $NOP_{(DB,k)}$, DB, db, $minCS$, $maxOR$, $Flist_{CDB}$ and $Flist_{DB}$. The outputs are $NOP_{(CDB,k)}$ and $CP_{(CDB,k)}$. First, the pruning of patterns in $NOP_{(DB,2)}$ is performed using Lemma 3 (Lines 1-3). Candidate patterns are computed using $NOP_{(CDB,(k-1))}$ (Line 4). Then, we compute OR and CS of $NOP_{(DB,k)}$ and C_k by performing one scan of db (Line 5). After computing OR and CS of $NOP_{(DB,k)}$, we update $NOP_{(CDB,k)}$ and $CP_{(CDB,k)}$. Some of the candidate patterns in C_k are pruned using Lemma 4. If $\Psi > 0$, we cannot prune a given pattern if it does not follow the order of the items in $Flist_{DB}$ (Lines 8-11). Then, we compute OR and CS of pattern in C_k by doing one scan of DB (Line 12). After computing counts of C_k, we update $NOP_{(CDB,k)}$ and $CP_{(CDB,k)}$.

D. Illustrative example

TABLE II: DB

TID	Items
1	{a,d}
2	{a,b,c}
3	{a,d}
4	{b,e}
5	{c}
6	{a,b,e}

TABLE III: db

TID	Items
7	{c,e}
8	{a,b,c}
9	{c,d,e}
10	{a,f}

TABLE IV: NOP_{DB}

Pattern	Pattern
{a}	{b,d}
{b}	{c,d}
{c}	{c,e}
{d}	{d,e}
{e}	{a,c,e}
{a,c}	{b,c,d}
{a,e}	{c,d,e}
{b,c}	

We explain the working of our proposed IncCMine algorithm using an example. Consider DB and db shown in Table II and Table III respectively. Let the values of $minRF$, $minCS$ and $maxOR$ used for the extraction of coverage patterns be 0.3, 0.8 and 0.5 respectively. The input to the algorithm also consists of iteration-wise $NOPs$ of DB, as shown in Table IV.

In the first iteration, $NOP_{(CDB,1)}$ and $CP_{(CDB,1)}$ are extracted. The items of size one, having frequency greater than ($minRF$ * size of the given database) are called $NOPs$. The $NOPs$ of size one in DB along with their frequencies are {a:4, b:3, c:2, d:2, e:2T10I10D100K}. The frequencies of these items are updated by performing one scan of db. After updating, the frequencies of the items in $NOP_{(DB,1)}$ are {a:5, b:4, c:5, d:3, e:4}. The candidate set consists of item {f:1}. These candidate items can be pruned because $F(f_{db})(1) < 0.3 * 4$ (Lemma 2). As the candidate set is empty, we need not scan DB. Finally, $NOP_{(CDB,1)}$ = {a:5, c:5, b:4, e:4, d:3} and $CP_{(CDB,1)}$ = {}. The set of CPs is empty as none of the items has frequency greater than 8 ($minCS * |CDB|$). The $Flists$ of DB and CDB are {a,b,c,d,e} and {a,c,b,e,d} respectively. The number of additional patterns is computed for each item. For item b, the number of additional patterns is $2^{|\{a,c\}|} - 2^{|\{a\}|}$ = 2. In this example, Ψ is 19.2.

The $NOPs$ of size 2 of DB are {{a,c}, {a,e}, {b,c}, {b,d}, {c,d}, {c,e}, and {d,e}}. The counts of these patterns are updated by scanning db. Before scanning db, some candidate patterns are generated and some patterns in $NOP_{(DB,2)}$ are pruned. The patterns that are pruned are {b,c},{d,e} as they do not follow the order of items in $Flist_{CDB}$. Candidate patterns generated are {{a,b}, {a,d}, {b,e}, {c,b}, {e,d}}. The OR and CS of these patterns were updated during the scan of db. Consider a candidate pattern X = {a,b}, after scanning db OR(X_{db}) = 1. This is not an NOP and can be pruned using Lemma 4. The patterns {c,b} and {e,d} cannot be pruned as they do not follow the order of the items in $Flist_{DB}$.

TABLE V: NOP_{CDB}

{a}	{a,e}	{b,d}
{c}	{c,b}	{e,d}
{b}	{c,e}	{c,b,d}
{d}	{c,d}	{c,e,d}
{e}	{b,e}	{b,e,d}
{a,c}		

TABLE VI: CP_{CDB}

Pattern	CS	OR
{a,c}	0.9	0.4
{a,e}	0.9	0.25
{c,b,d}	0.9	0.33
{c,e,d}	0.9	0.33
{b,e,d}	0.8	0.33

The counts of these candidate patterns {a,d}, {b,e}, {c,b} and {e,d} are updated scanning DB. Patterns having OR less than $maxOR$ are added to the set $NOP_{(CDB,2)}$ and patterns having OR less than $maxOR$ and CS greater than $minCS$ are added to the set $CP_{(CDB,2)}$. We follow a similar procedure for iteration 3 as well. The final sets of $NOPs$ and CPs are shown in Table V and Table VI respectively.

V. Performance evaluation

We have conducted experiments by implementing our proposed IncCMine algorithm as well as the reference CMine algorithm in *Python 2.7*. Our experiments were performed using a computer having a fifth-generation Intel Core-i5 2.7 GHz processor with 8 GB RAM.

The experiments were conducted on three datasets. The first dataset is BMS-POS dataset, which is a click-stream dataset [1] of an e-commerce company; the dataset has 515,596 transactions and 1656 distinct items. The second dataset is CABS120k08 dataset, which is a click stream dataset [22], is a collection of search queries, documents clicked, time-stamps and user id. The transactions from this dataset are extracted using the 30-minutes rule [10]. The cabs120K08 dataset has 402,776 transactions and 572 distinct items. The third dataset is the T40I10D100K, which is a synthetic dataset [3] generated by a dataset generator. This synthetic dataset had 100,000 transactions and 870 distinct items. We shall henceforth refer to this dataset as *Synthetic* dataset.

The experiments for performance evaluation were conducted by changing increment size (Δ), $maxOR$ and $minRF$. For experiments varying Δ, in BMS-POS dataset, among 515,596 transactions, we have taken first 265,596 transactions as DB and Δ is varied from 25,000 to 250,000 transactions with the step-size of 25,000 transactions. For Cabs120K08 dataset, among 402,776 transactions, we have taken 202,776 transactions as DB and Δ is varied from 20,000 to 200,000 transactions with the step-size of 20,000 transactions. For Synthetic, among 100,000 transactions, we have taken first 50,000 transactions as DB and Δ is varied from 5,000 to 50,000 transactions with the step-size of 5,000 transactions.

Table VII summarizes the parameters used in our experiments. We have selected appropriate values of $maxOR$, $minRF$ and $minCS$ (shown in Table VII) for each of the datasets under consideration based on the results of our preliminary experiments.

As reference, we used the CMine algorithm [23] discussed in Section 2. Notably, CMine is the closest to our proposed approach because no other work exists for the extraction of coverage patterns. Performance metrics are as follows: (a) Execution time (ET): the processing time of CPU in seconds to extract CPs (b) Number of candidates (NC): the number of candidate patterns scanned in the original database.

TABLE VII: Parameters used in our experiments

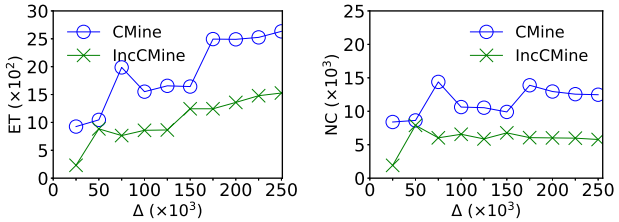
Dataset	Parameter	Default value	Variations
BMS-POS	Δ	100K	[25K,250K] step-size 25K
	$\mid DB \mid$	265,596	
	$minRF$	0.065	[0.065, 0.095] step-size 0.005
	$minCS$	0.5	
	$maxOR$	0.6	[0.1, 0.9] step-size 0.1
CABS120k08	Δ	60K	[20K,200K] step-size 20K
	$\mid DB \mid$	202,776	
	$minRF$	0.035	[0.035, 0.06] step-size 0.005
	$minCS$	0.5	
	$maxOR$	0.4	[0.05, 0.5] step-size 0.05
Synthetic	Δ	25K	[5K,50K] step-size 5K
	$\mid DB \mid$	50K	
	$minRF$	0.045	[0.045, 0.06] step-size 0.0025
	$minCS$	0.3	
	$maxOR$	0.4	[0.05, 0.5] step-size 0.05

A. Effect of variations in Δ

Figure 1 depicts the effect of variations in Δ. For BMS-POS, the changes in ET and NC are shown in Figure 1(a) and Figure 1(b) respectively. The fluctuations in ET and NC observed indicates that the $Flist$ is changing continuously. It can be observed that IncCMine is up to 5 times faster than CMine as NC for IncCMine is significantly lower. The significant lower value of NC when compared to CMine occurs because our proposed IncCMine algorithm does not scan DB for the existing knowledge of patterns (NOP_{DB}). IncCMine is two times faster when Δ is equal to the size of DB. Similar trends are observed for ET and NC in CABS120k08 dataset, as shown in Figures 1(c) and 1(d) respectively. For Synthetic dataset, the fluctuations are not observed, representing no significant changes in the $Flist$ for Synthetic dataset. The changes in ET and NC for Synthetic dataset are depicted in Figures 1(e) and 1(f) respectively.

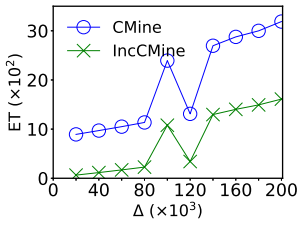
B. Effect of variations in maxOR

Figure 2 depicts the effect of variations in $maxOR$. For BMS-POS, the changes in ET and NC are depicted in Figures 2(a) and 2(b) respectively. ET and NC of CMine and IncCMine are increasing continuously with the increase in the value of $maxOR$ representing the more number of CPs, which are extracted with increase in $maxOR$. Exponential increase in ET and NC is observed for CMine for the three datasets representing more number of $NOPs$ that are extracted with small changes in $maxOR$. In contrast, for IncCMine, the increase in ET and NC is not exponential, because for the patterns present in NOP_{DB} were not scanned

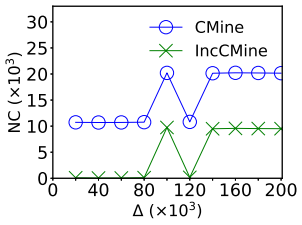


(a) ET (BMS-POS)

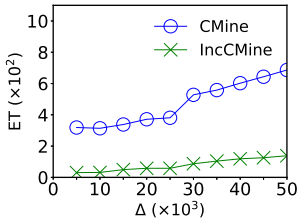
(b) NC (BMS-POS)



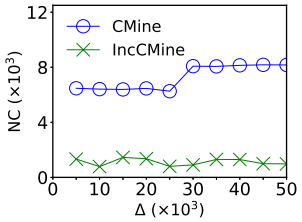
(c) ET (CABS120k08)



(d) NC (CABS120k08)

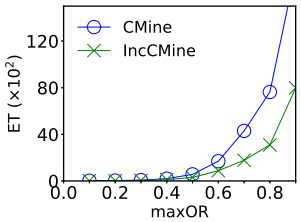


(e) ET (Synthetic)

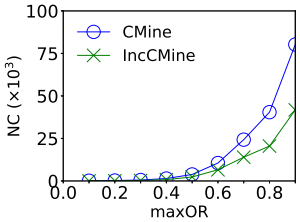


(f) NC (Synthetic)

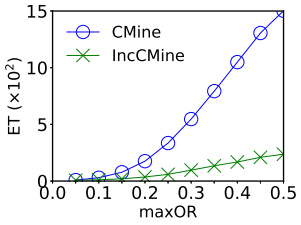
Fig. 1: Effect of variations in Δ



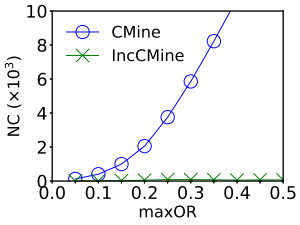
(a) ET (BMS-POS)



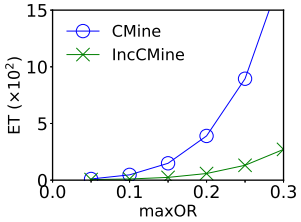
(b) NC (BMS-POS)



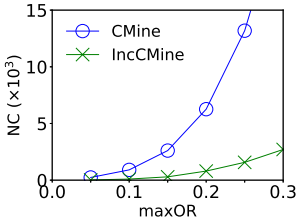
(c) ET (CABS120k08)



(d) NC (CABS120k08)



(e) ET (Synthetic)



(f) NC (Synthetic)

Fig. 2: Effect of variations in $maxOR$

again in DB. IncCMine is 7 times and 13 times faster than CMine when $maxOR$ = 0.05 for CABS12k08 and Synthetic datasets respectively. The reason is that, for the patterns present in NOP_{DB}, we are not scanning DB.

C. Effect of variations in minRF

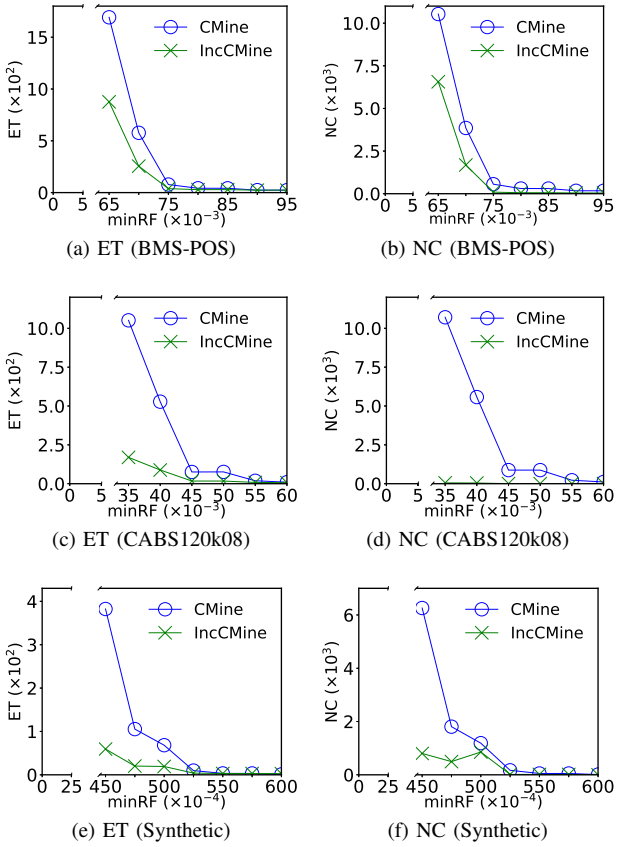
Figure 3 depicts the effect of variations in $minRF$. For BMS-POS, the changes in ET and NC are shown in Figures 3(a) and 3(b) respectively. The decrease in ET and NC for CMine and ICMine with the increase in $minRF$ represents the decrease in the number of $NOPs$ extracted. For BMS-POS and CABS120k08 dataset, the gradual decrease in ET and NC show that small changes in $Flist$ with increase in $minRF$. However, for Synthetic dataset, there is a sudden fall in ET and NC, which shows that most of the items are having comparable frequencies. For BMS-POS, IncCMine is 2 times faster than CMine when $minRF$ is 0.065. For CABS120k08, IncCMine is 4 times faster than CMine when $minRF$ is 0.035. For Synthetic, IncCMine is 8 times faster than CMine when $minRF$ is 0.045. The improvement in execution times is due to significantly lower values of NC, which is because for patterns present in NOP_{DB}, we are not scanning DB.

D. Changes in Ψ due to variations in Δ

We study the changes in Ψ by varying Δ. The results on three datasets are depicted in Figure 4. For BMS-POS dataset, as we vary Δ, the value of Ψ increases slowly with fluctuations. It reaches 21 percent when Δ = 250K, i.e., the overall size of the incremental database is the same as the original database. The result shows there is no significant increase in Ψ even though the database size is doubled. Notably, the maximum Ψ for CABS120k08 (Figure 4(b)) and Synthetic (Figure 4(c)) is 0.4 percent and 6 percent respectively. The results for three datasets show that even though database size is doubled the value of Ψ is not significantly high. The results indicate that the proposed approach is potentially applicable to most of the common applications.

E. Discussion: Setting the threshold for Ψ

Recall that Ψ_{max} is the threshold value of Ψ. We shall now explain about fixing the value of Ψ_{max}. Notably, the value of Ψ is just an estimate of the percentage of the additional patterns, if $Flist_{CDB}$ differs from $Flist_{DB}$. The value of Ψ is 0, when the $Flist_{CDB}$ does not differ from $Flist_{DB}$. The value of Ψ is 100, when $Flist_{CDB}$ and $Flist_{DB}$ are disjoint sets and $maxOR$ is 1. When Ψ is 100, the performance of CMine algorithm is better than the proposed IncCMine algorithm because every pattern is additional pattern and IncCMine performs additional computation for extracting CPs. The value of Ψ_{max} is fixed based on the number of APF which fit in the main memory.

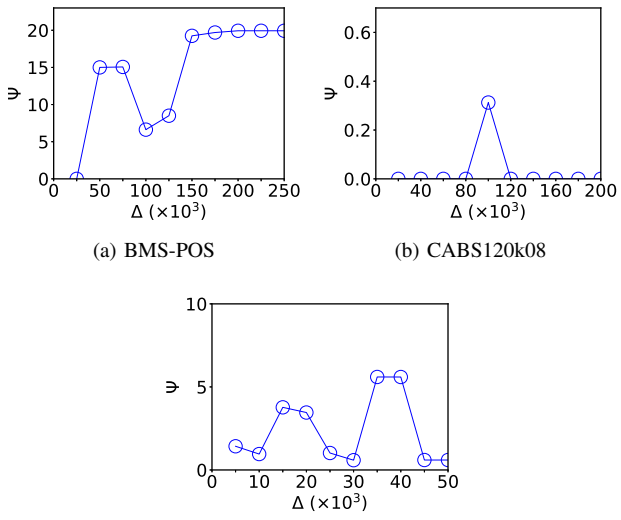


(a) ET (BMS-POS) (b) NC (BMS-POS)

(c) ET (CABS120k08) (d) NC (CABS120k08)

(e) ET (Synthetic) (f) NC (Synthetic)

Fig. 3: Effect of variations in minRF



(a) BMS-POS (b) CABS120k08

(c) Synthetic

Fig. 4: Changes in Ψ due to variations in Δ

VI. Conclusion

In practice, pattern mining is performed on huge databases, which tend to get updated over the course of time. Hence, incremental mining aims at *efficiently* and *incrementally* mining patterns when a database is updated. While incremental mining algorithms have been proposed for extracting frequent patterns, sequential patterns and utility patterns, existing works have so far not investigated incremental mining in the context of coverage patterns. In this paper, we have addressed the problem of incremental mining in the context of coverage patterns and proposed the IncCMine algorithm for efficiently extracting the knowledge of coverage patterns when incremental database is added to the existing database. The results of our extensive performance evaluation using both real and synthetic datasets demonstrate that our proposed IncCMine algorithm significantly outperforms the existing CMine algorithm. As part of future work, we plan to develop incremental algorithms for pattern growth approach towards extracting coverage patterns. We also plan to develop incremental algorithms when deletion and modification of transactions are also considered.

ACKNOWLEDGEMENTS

This work is supported by Kohli Center on Intelligent Systems, IIIT-Hyderabad. The authors are thankful to the anonymous referees for their useful comments.

References

[1] Dataset repository: http://fimi.cs.helsinki.fi
[2] Agarwal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. VLDB. pp. 487–499 (1994)
[3] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. ACM SIGMOD. vol. 22, pp. 207–216 (1993)
[4] Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE TKDE **21**(12), 1708–1721 (2009)
[5] Aumann, Y., Feldman, R., Lipshtat, O., Manilla, H.: Borders: An efficient algorithm for association generation in dynamic databases. JIIS **12**(1), 61–73 (1999)
[6] Borah, A., Nath, B.: Rare association rule mining from incremental databases. Pattern Analysis and Applications pp. 1–22 (2018)
[7] Budhiraja, A., Ralla, A., Reddy, P.K.: Coverage pattern based framework to improve search engine advertising. International Journal of Data Science and Analytics pp. 1–13 (2018)
[8] Budhiraja, A., Reddy, P.K.: An approach to cover more advertisers in adwords. In: Proc. DSAA. pp. 1–10 (2015)
[9] Budhiraja, A., Reddy, P.K.: An improved approach for long tail advertising in sponsored search. In: Proc. DASFAA. pp. 169–184 (2017)
[10] Chau, M., Fang, X., Liu Sheng, O.R.: Analysis of the query logs of a web site search engine. JASIST **56**(13), 1363–1376 (2005)
[11] Cheng, H., Yan, X., Han, J.: Incspan: incremental mining of sequential patterns in large database. In: Proc. ACM SIGKDD. pp. 527–532 (2004)
[12] Cheung, D.W., Lee, S.D., Kao, B.: A general incremental technique for maintaining discovered association rules. In: Proc. DASFAA, pp. 185–194 (1997)
[13] Cheung, D.W., Wong, C., Han, J., Ng, V.T.: Maintenance of discovered association rules in large databases: An incremental updating techniques. In: Proc. ICDE. pp. 106–114 (1996)
[14] Fasihy, H., Shahraki, M.H.N.: Incremental mining maximal frequent patterns from univariate uncertain data. Knowledge-Based Systems **152**(C), 40–50 (2018)
[15] Kavya, V.N.S., Reddy, P.K.: Coverage patterns-based approach to allocate advertisement slots for display advertising. In: Proc. ICWE. pp. 152–169 (2016)
[16] Lin, M.Y., Hsueh, S.C., Chan, C.C.: Mining and maintenance of sequential patterns using a backward generation framework. Journal of Information Science & Engineering **34**(5), 1329–1349 (2018)
[17] Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proc. ACM SIGKDD. pp. 337–341 (1999)
[18] Masseglia, F., Poncelet, P., Teisseire, M.: Incremental mining of sequential patterns in large databases. DKE **46**(1), 97–121 (2003)

[19] Nguyen, L.T., Nguyen, P., Nguyen, T.D., Vo, B., Fournier-Viger, P., Tseng, V.S.: Mining high-utility itemsets in dynamic profit databases. Knowledge-Based Systems **175**, 130–144 (2019)
[20] Nguyen, S.N., Sun, X., Orlowska, M.E.: Improvements of incspan: Incremental mining of sequential patterns in large database. In: Proc. PAKDD. pp. 442–451 (2005)
[21] Nguyen, T.T.: Mining incrementally closed item sets with constructive pattern set. Expert Systems with Applications **100**, 41–67 (2018)
[22] Noll, M.G., Meinel, C.: The metadata triumvirate: Social annotations, anchor texts and search queries. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 640–647 (2008)
[23] Srinivas, P.G., Reddy, P.K., Bhargav, S., Kiran, R.U., Kumar, D.S.: Discovering coverage patterns for banner advertisement placement. In: Proc. PAKDD. pp. 133–144 (2012)
[24] Srinivas, P.G., Reddy, P.K., Trinath, A.V., Bhargav, S., Kiran, R.U.: Mining coverage patterns from transactional databases. JIIS **45**(3), 423–439 (2015)
[25] Trinath, A., Srinivas, P.G., Reddy, P.K.: Content specific coverage patterns for banner advertisement placement. In: Proc. DSAA. pp. 263–269 (2014)
[26] Wang, J.Z., Huang, J.L.: On incremental high utility sequential pattern mining. ACM TIST **9**(5), 55:1–55:26 (2018)
[27] Yun, U., Nam, H., Lee, G., Yoon, E.: Efficient approach for incremental high utility pattern mining with indexed list structure. Future Generation Computer Systems **95**, 221–239 (2019)