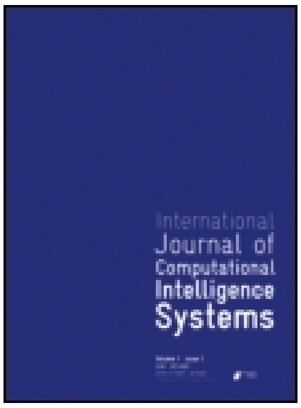

International Journal of Computational Intelligence Systems
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcis20

Progressive CFM-Miner: An Algorithm to Mine CFM - Sequential Patterns from a Progressive Database

Bhawna Mallick [a] , Deepak Garg [a] & P. S. Grover [b]

[a] Department of Computer Science & Engineering, Thapar University, Patiala, India

[b] Department of Computer Science & Engineering, Guru Tegh Bahadur Institute of Technology, Delhi, India

Version of record first published: 22 Jan 2013.

To cite this article: Bhawna Mallick , Deepak Garg & P. S. Grover (2013): Progressive CFM-Miner: An Algorithm to Mine CFM - Sequential Patterns from a Progressive Database, International Journal of Computational Intelligence Systems, 6:2, 209-222

To link to this article: http://dx.doi.org/10.1080/18756891.2013.768432

Progressive CFM-Miner: An Algorithm to Mine CFM – Sequential Patterns from a Progressive Database

Bhawna Mallick
Department of Computer Science & Engineering
Thapar University, Patiala, India
bhawnamallickphd@gmail.com

Deepak Garg
Department of Computer Science & Engineering
Thapar University, Patiala, India

P. S. Grover
Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology, Delhi, India

Received 27 January 2012

Accepted 6 September 2012

Abstract

Sequential pattern mining is a vital data mining task to discover the frequently occurring patterns in sequence databases. As databases develop, the problem of maintaining sequential patterns over an extensively long period of time turn into essential, since a large number of new records may be added to a database. To reflect the current state of the database where previous sequential patterns would become irrelevant and new sequential patterns might appear, there is a need for efficient algorithms to update, maintain and manage the information discovered. Several efficient algorithms for maintaining sequential patterns have been developed. Here, we have presented an efficient algorithm to handle the maintenance problem of CFM-sequential patterns (Compact, Frequent, Monetary-constraints based sequential patterns). In order to efficiently capture the dynamic nature of data addition and deletion into the mining problem, initially, we construct the updated CFM-tree using the CFM patterns obtained from the static database. Then, the database gets updated from the distributed sources that have data which may be static, inserted, or deleted. Whenever the database is updated from the multiple sources, CFM tree is also updated by including the updated sequence. Then, the updated CFM-tree is used to mine the progressive CFM-patterns using the proposed tree pattern mining algorithm. Finally, the experimentation is carried out using the synthetic and real life distributed databases that are given to the progressive CFM-miner. The experimental results and analysis provides better results in terms of the generated number of sequential patterns, execution time and the memory usage over the existing IncSpan algorithm.

Keywords: Sequential pattern mining, CFM-PrefixSpan, Progressive database, updated CFM-tree, progressive CFM patterns, algorithms.

Abbreviations:
CFM: Compactness, Frequency, Monetary; min_sup: minimum support; CFM-tree: Tree with Compact frequent monetary values of sequential pattern; C_T: Compact threshold; T_m: Monetary threshold

1. Introduction

The concern in the discovery of hidden information has improved in the past decade as a result of the fast growth of stored data in digital form. Basket analysis which is based on identifying frequent associations between elements in sets is one approximation to the problem of discovery of hidden information [26, 27]. Application of this approach to the treatment of sequential data results in one important special case. The process of finding all sub-sequences that occur often on a specified sequence database and have minimum support threshold is known as sequential pattern mining [1]. Data is normally assumed to be centralized, memory-resident, and static by conventional methods for sequential mining. Ensuring system scalability and enabling knowledge discovery when data is dynamic and distributed necessitates efficient incorporation of incremental data mining techniques [2].

Incremental mining algorithms efficiently calculate the new set of frequent item sets by fundamentally reusing beforehand mined information and attempting to merge this information with the fresh data. In fact, several application domains incrementally update the contents of databases. For instance, appending of newly bought items for existing customers for their later buying and/or inclusion of new shopping successions for new customers causes the shopping transaction database to grow on a daily basis [3]. This helps to reduce the computational and I/O expenses [4].

Numerous algorithms are available today for incremental mining of sequential patterns [5], [6], [7], [8], and [9]. Generally, incremental growth exists in several real life sequence databases. The information discovered must be updated, maintained and managed by an efficient algorithm to reflect the current state of the database when previous sequential patterns would become irrelevant and new sequential patterns might appear [10].

In this research, we have presented an efficient progressive CFM-miner algorithm to handle the maintenance problem of CFM-sequential patterns, which was introduced in our previous research work (explained in Section 4, Step 1) [11]. In order to efficiently confine the dynamic nature of data addition and deletion into the sequential pattern mining problem, we have constructed the updated CFM-tree using the CFM patterns obtained from the static database. Then, the database gets updated from the distributed sources. Whenever the database is updated from the multiple sources, CFM-tree is also updated by including the updated sequence. Then, the updated CFM-tree is used to mine the progressive CFM-patterns using the proposed tree pattern mining algorithm. The experimentation is carried out with the aid of synthetic and real life datasets that are given to the progressive CFM-miner using thread environment.

The organization of the paper is as follows: The review of related research is given in section II. The problem statement is described in section III and the proposed algorithm for mining of CFM-sequential patterns is given in section IV. The experimental results and its discussion are presented in section V and the conclusions are summed up in section VI.

2. Literature Survey

The literature has presented with a huge number of approaches for constraint-based sequential pattern mining [12], [13], [14], [15] and incremental mining of sequential patterns. In recent times, developing approaches for incremental mining of sequential patterns has gained immense importance in real life applications. A concise review of some recent research work related to the incremental mining of sequential patterns is presented here.

Enhong Chen et al. [16] have proposed a well-organized method to address tough aggregate constraints. First they have shown that two typical types of constraints could be converted into the same form and therefore could be manipulated in a reliable manner by means of a concept of total contribution of sequences based theoretical examination of the hard aggregate constraints. Then, the cost of using tough aggregate constraints has been decreased by proposing an algorithm incorporating two efficient strategies called PTAC (sequential frequent Patterns mining with Tough Aggregate Constraints). One has utilized the potential features shown by some other items and validity of the corresponding prefix to avoid inspecting data items. The other has successfully pruned those unpromising new patterns which may otherwise function as new prefixes to avoid constructing an unnecessary projected database.

Ming-Yen et al. [17] have proposed a method called DELISP (Delimited Sequential Pattern) to provide the amenities existing in the pattern-growth methodology. DELISP has scale down the size of proposed databases utilizing bounded and windowed projection methods. Time-gap valid subsequences have been preserved by bounded projection and redundant subsequences fulfilling the sliding time-window constraint has been saved by windowed projection. In addition, constraint-satisfactory patterns have been directly produced by the subtended growth technique and the pace of the pattern growing process has been increased. The good scalability and superior performance of DELISP over prominent GSP (Generalized Sequential Pattern) algorithm has been proved by the comprehensive experiments.

Jong Bum Lee et al. [18] have proposed an incremental frequent pattern mining algorithm based on Apriori-TFP (Total-from-Partial) for efficiently searching subjected to restriction of memory and the additional categorization work based on those patterns. Particularly, the problem of mining frequent patterns from incrementally increased, large size of data sets has been made possible through the concept of pre-infrequent patterns pruning and utilization of two distinct minimum supports. Ming-Yen Lin et al. [19] have developed a technique for efficient incremental pattern discovery, called backward mining that takes into account the incremental characteristics of sequence-merging. They have used a backward mining strategy in their proposed algorithm, called BSPinc, for incremental mining of sequential patterns. They have detected and removed those stable sequences that have constant support counts in the updated database, from the support counting process. Mining of backward extensions produced candidate sequences could be performed recursively inside the ever-shrinking space of the projected sequences.

Jen-Wei Huang et al. [20] have presented an all-purpose model of sequential pattern mining to secure the dynamic nature of data addition and deletion in a progressive database. In addition, they have progressively identified sequential patterns in specified time interval of interest (POI) by a proposed progressive algorithm called progressive mining of sequential patterns (Pisa). The POI has been a sliding window that constantly progresses with the passage of time. The most recent data sequences have been efficiently maintained, entire collection of up-to-date sequential patterns have been detected and obsolete data and patterns have been removed by Pisa by employing a progressive sequential tree. The memory required by Pisa has been remarkably smaller than that of the optional method, namely, direct appending (DirApp) as height of the sequential pattern tree proposed bounded by the length of POI effectively restricts the memory space needed by Pisa.

L. Vinceslas et al. [21] have proposed an on-line algorithm called SPAMS, to handle the sequential patterns mining problem in data streams. Their algorithm has preserved the set of frequent sequential patterns using an automaton-based structure called SPA (Sequential Pattern Automaton). The problem of combinatorial explosion of sequential patterns has been permitted by the sequential pattern automaton which is smaller than the set of frequent sequential patterns by two or more orders of magnitude.

Yue Chen et al. [22] have demonstrated that all sequential patterns cannot be fully mined by existing incremental mining algorithm based on PrefixSpan, namely IncSpan+ which was proposed in PAKDD'05. Then, using prefix tree, they have proposed an incremental mining algorithm of sequential patterns. Their algorithm has continuously scanned the incremental element set to remove the search space by maintaining the tree structure by means of width pruning and depth pruning after creating a prefix tree to signify the sequential patterns. The algorithm has been proved to achieve good performance by experiment. Lei Chang et al. [23] have analyzed the incremental mining problem of closed sequential patterns. The closed sequential patterns have been retained by a designed compact structure CSTree and an efficient algorithm IMCS (Incremental Mining of Closed Sequential Patterns) has been constructed to maintain the CSTree when the sequence database is updated incrementally.

The major differences of progressive CFM-miner with the existing works are discussed here. In [16, 17], the constraints such as aggregate and time gap are incorporated into the sequential pattern mining. But, in the proposed work, the super market scenario-based three constraints are identified and incorporated into the sequential pattern mining work. In [18] and [19], the incremental sequences are mined through the help of candidate-pruning procedure. But, the proposed progressive CFM-miner considered the projection-based mining procedure so that the efficiency of the algorithm will be improved. In [20, 21], the progressive database [20] and data streams [21] are used for mining the sequence without considering the constraints that is handled in our work based on the real scenarios. In [22], the incremental procedure is applied to projection-based mining and the incremental closed patterns are mined in [23]. Overall, the proposed algorithm considered the progressive database or incremental database along with the real scenarios-based constraints for obtaining the significant and useful sequential pattern.

3. Contribution of the Paper

In this article, we have presented an efficient progressive CFM-miner algorithm to handle the maintenance problem of CFM-sequential patterns. The constraints used are compactness, frequency and monetary constraints that are incorporated in the well known pattern growth algorithm Prefixspan to extract sequential patterns. The paper contributes to the field of sequential pattern mining due to following features:

1. Handle efficiently the dynamic nature of data addition and deletion into the sequential pattern mining problem.
2. The first simple algorithm (to the best extent of our knowledge) that use tree data structure to consider constraint based sequential pattern mining from progressive databases.
3. The algorithm can handle the database that gets updated from the distributed sources using the thread environment.
4. The experimentation is carried out with the synthetic and real life datasets that are given to the progressive CFM-miner. The empirical evaluation performed shows that the proposed algorithm gives good performance as compared to IncSpan algorithm.
5. Various news terms are defined like progressive compact sequence, precious CFM node, incompact node that could used for further research.
6. The algorithm gives good results for execution time, memory usages and number of sequential patterns extracted when compared with different values of minimum support thresholds.

4. Problem Definition

Let I = { i_1, i_2, …….., i_m } be a set of literals, called items. An item set 'X' is a set of items hence, $X \subseteq 1$. A sequence S = (s_1, s_2, ….., s_n) is an ordered set of item sets. Consider two sequences S_1 = (a_1, a_2, ……, a_k) and S_2 = (b_1, b_2, ……, b_l). We say that S_1 contains S_2, or equivalently, S_2 is a subsequence of S_1 if there exist integers j_1, j_2, …..j_l such that $1 < j_1 < j_2 < …..< j_l < k$ and $b_1 \subseteq aj_1$, $b_2 \subseteq aj_2$, …, $b_l \subseteq aj_l$, represented as $S_2 \subseteq S_1$. A sequence S is said to be constraint (in terms of monetary and compactness) if a sequence S should follow the specified constraint, C such that, $C \rightarrow \{(M_1 + M_2 + …. + M_n) / n\} \geq T_m$ and $C \rightarrow t_n - t_1 < C_T$, where, M_n represents the monetary value. For the incremental update problem, we consider that the constraint sequential pattern mining can be executed on a database D to find the constraint sequences. But, the database D is then updated by inserting a set of sequences ΔD. Let us denote the updated database D' such that D' = (D) ʋ ΔD. Here, the incremental update problem is to find all constraint frequent sequences in the database D' for each next time intervals without scanning the whole database D'. Given, a user-specified compact length C_T , monetary value T_m and a user defined minimum support threshold min_sup, the progressive CFM-patterns whose, occurrence frequencies, compact length and monetary value are greater than or equal to the user specified thresholds should be mined from the progressive database, 'D' that gets updated from the distributed sources.

Based on this problem statement, we define the important terms used in the proposed approach to mine the progressive CFM patterns.

Definition 1 (CFM-tree): For a sequence database D, we can construct a CFM-tree after mining the CFM patterns from it. Here, every node 'n' in the CFM-tree contains items and its relevant information, represented as, n= [(p {t_1, t_n}), (M, F)], where, 'p' is the item, t1 is the starting time interval, tn is the ending time interval, M is monetary and F is frequency. Here, the depth of the CFM-tree,'d' is equivalent to the larger length of the CFM-sequential patterns.

Definition 2 (Empty node): A node in the CFM-tree is called as empty node only if (i) t_l and t_n is filled with zero, (ii) 'p' should contain the item information and (iii) M and F have the zero value. This node is necessary for building the CFM-tree after mining the sequences from the static database because the CFM-miner does not satisfy the downward closure property. So, some of the sequential patterns are frequent but, their subsets may not be frequent. These types of subsets are stored in the empty nodes, but their supersets are stored in the precious CFM-node that is frequent.

Definition3 (Precious CFM-node): A node in the CFM-tree is called as precious CFM-node only if (i) t_l and t_n contains the information of time occurrence, (ii) 'p' should contain the item and, (iii) M and F have the valuable information about its monetary and frequency.

Definition4 (updated CFM tree): After inserting some nodes in CFM-tree on behalf of updated database, then it is called as, updated CFM-tree, in which some nodal information may be updated or some new nodes may be included.

Definition5 (Incompact node): A node in the updated CFM-tree is known as, incompact node only if, tn value is less than T, where, T is the user specified threshold.

Definition6 (Non-zero infrequent node): A node in an updated CFM Tree is said to be a non-zero infrequent node if the following conditions are satisfied: (i) the frequent value, F should be less than the 'min_sup', ii) should not be updated and (iii) the monetary constraint should be satisfied.

Definition7 (Progressive compact sequence): Let S = $\{(p_1, t_1, M_1), (p_2, t_2, M_2),\ldots\ldots, (p_n, t_n, M_n)\}$ be a data sequence of database D and a sequence Su = $\{(q_m, t_{n+1}, M_m)\}$ be an updated sequence, where p_j is an item, m_j is a purchasing money and t_j signifies the time at which p_j occurs, $1 \leq j \leq n$ and $t_{j-1} \leq t_j$ for $2 \leq j \leq n$. 'P' denotes a set of items in the database D. A sequence $S_s = \{(q_1,t_1,M_1),(q_2,t_2,M_2),\ldots.,(q_m,t_m,M_m)\}$ is said to be a progressive compact sequence only if, (a) item set S_s is a subsequence of $S||S_u$, (b) S_s should have item, q_m, and (c) the compactness constraint is satisfied, i.e. $t_m - t_1 \leq C_T$.

Definition8 (Progressive compact monetary sequence): Let S = $\{(p_1, t_1, M_1), (p_2, t_2, M_2), \ldots, (p_n, t_n, M_n)\}$ be a data sequence of database D and a sequence $S_u = \{(q_m,t_{n+1}, M_m)\}$ be an updated sequence, where p_j is an item, m_j is a purchasing money and t_j signifies the time at which p_j occurs, $1 \leq j \leq n$ and $t_{j-1} \leq t_j$ for $2 \leq j \leq n$. 'P' denotes a set of items in the database D. A sequence $S_s = \{(q_1,t_1,M_1),(q_2,t_2,M_2),\ldots.,(q_m,t_m,M_m)\}$ is said to be a progressive compact monetary sequence only if, (a) item set S_s is a subsequence of $S||S_u$, (b) S_s should have item q_m, (c) the compactness constraint is satisfied, i.e. $t_m - t_1 \leq C_T$ and (d) the monetary constraint is satisfied, i.e. $\{(M_1 + M_2 + \ldots.+ M_m) / m\} \geq T_m$.

5. An Algorithm to Mine CFM –Sequential Patterns from a Progressive Database

In reality, sequence databases are updated incrementally. The changes on the database may invalidate some existing sequential patterns and introduce new ones. Instead of re-computing the database each time, the incremental mining algorithms target efficiently maintaining the sequential patterns in the dynamically changing database. With the evolution of databases, some existing sequential patterns would be invalid and some new sequential patterns might be introduced. Thus, maintaining sequential patterns (over a significantly long period) becomes essential for sequential pattern mining. Generally, the change on a sequential database can be categorized as (a) deleting records, (b) inserting new records and (c) appending new items on the existing records. By handling these issues, the proposed algorithm was designed with the aid of five major steps.

1. Mining of CFM sequential patterns from the static database
2. Building up the CFM-tree from the CFM patterns
3. Handling the update operation
4. Handling the node deletion operation in the updated CFM-tree
5. Mining of progressive CFM patterns from the progressive database

Step 1: Mining of CFM Sequential Patterns from the Static Database

In this, CFM patterns from the static database are efficiently mined using the CFM algorithm proposed in our previous work [11]. The relevant part of the CFM algorithm that is based on PrefixSpan [24] is presented here for the completeness of this article. We have used two concepts namely, monetary and compactness that are derived from the aggregate and duration constraints which are presented in the available literature. To begin with, the proposed algorithm discovered the 1-length compact frequent patterns (1-CF) by considering the compactness threshold (C_T) and support threshold (min_sup). Then, we filtered the 1-length compact frequent monetary sequential patterns (1-CFM) from the mined 1-CF patterns by inputting the monetary constraint (T_m). Subsequently, we built the projected database corresponding to the mined 1-CF patterns and 2- CF patterns that are mined from the projected database. Again, we found the 2-CFM sequential patterns from it and the process was applied recursively until all length CFM sequential patterns were mined.

Example: The sample database is given in Table 1 in which the timestamps T1 to T5 are static set of data, whereas the timestamps T6 to T7 are the updated set of data. The corresponding monetary values of all the items are given in Table 2 and the mined CFM-sequential patterns using our previous algorithm for the input thresholds, (min_sup >=2, C_T <= 3, T_m >=10) are shown in Table 3.

Table 1. Sample Database

Seq. Id	T1	T2	T3	T4	T5	T6	T7
01	a	abc	ac	d	cf		
02	ad	c	bc	ae		h	
03	f	ab	c	df	cb		
04		g	af	c	b		
05						cb	
06							ab

Table 2. Monetary Table

Item	Monetary value
a	2
b	10
c	20
d	20
e	5
f	15
g	25
h	2

Table 3. Mined CFM-Sequential Patterns

CFM-Sequential Patterns	
a	<ac>,<acb>,<acc>
b	<b>,<bc>,<bcc>,<bcd>,<bcdc>
c	<c>,<ca>,<cc>
(ab)	<(ab)>,<(ab)c>,<(ab)cc>,<(ab)cd>,<(ab)cdc>
(bc)	<(bc)>,<(bc)a>
d	<d>
f	<f>,<fc>

Step 2: Building up the CFM-tree from the CFM Patterns

After the mining process of the CFM-sequential patterns, we have built the CFM- tree from the mined CFM-sequential patterns. The process of building up the CFM- tree is explained as follows. The monetary value and the frequency value of each of the patterns should be maintained properly. The CFM tree that contains all the sequential patterns are building up, by which mines the progressive CFM-patterns without candidate generation, requires the less database scans to achieve a highly compact frequency and the monetary tree structure. According to the frequency and monetary list, it produces a CFM-pattern tree, which can store compact information on transactions involving sequential patterns. At first, transactions are inserted into the CFM-tree according to a predefined order one by one. The order of all the patterns of a CFM-tree is maintained by a list, which maintains the current frequency value and the monetary value with the timestamp of each item. Here, each level refers to the length of sequential patterns so the depth of the CFM-tree is identical to the maximum length of the sequential patterns.

Example: The first insertion phase begins with the root node by taking all the mined patterns. By taking the patterns that has prefix 'a', the obtained sequential patterns from the CFM-mining algorithm are <ac>, <acb>, <acc>. Initially, the empty node 'a' is appended with the root node of the tree by giving the corresponding monetary value and the frequency value. When we take the obtained sequential pattern <ac>, here the precious CFM-node 'c' is added to the node 'a' to achieve the building process of <ac>. The monetary value of 'ac' is considered as 11, which is found to compute the average of their monetary values. Likewise, all the remaining patterns are utilized to build the CFM-pattern tree. The final CFM-tree for the static database is shown in Fig. 1.

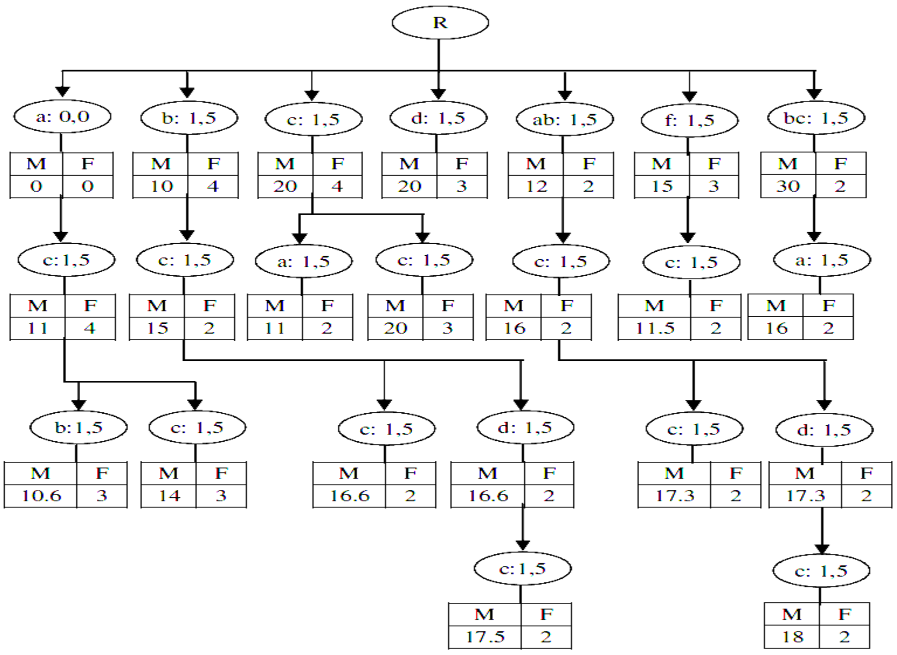

Fig.1. CFM-Pattern Tree for the Static Database

Step 3: Handling the Update Operation

After building up the CFM-tree from the static database, we have to build the tree structure of the updated sequences. After inserting some of the transactions, if the items order of the list deviates from the current frequency and monetary to a specified degree, the CFM-tree is dynamically restructured by the current frequency and monetary and the list updates the pattern order with the current list. The sequential patterns obtained from the updated sequences are incremented based on the timestamps, monetary value and the frequency of each patterns. While updating the tree structure, CFM-tree constantly maintains the initial sort order of the sequential patterns with their information. Thus, it adds the new frequent items at the end of a list and it constructs to maintain the frequency of each item and in the tree structure as new nodes. The information about the frequency and the monetary value should be updated in a timely manner. The timestamp of the sequence in the child node should be updated as the new one. This is reasonable because for every element between the old timestamp and the new one, they are already appended to this node as a candidate sequential pattern with the old timestamp. Thus, the sequential patterns between the old timestamp and the new one can be found. Additionally, for the elements after the new timestamp, appending them to the node having the sequence with the new timestamp is the only way to find up-to-date sequential patterns beginning at the new timestamp.

Example: By considering the updating nodes of the CFM-tree, the newly inserted items are arrived in a periodic manner. Here, in timestamp T6, the items '<h>' and '<cb>' are the new set of items. We have to update these into the existing CFM-tree dynamically. While updating '<h>', the progressive compact sequence obtained are <ch>, <(bc)h>, <c(bc)h>, <(bc)(ae)h>, <cbh>, <cch>, <ccah>, <cceh>. These patterns are updated sequentially into the CFM-pattern tree along with the information about the frequency and the monetary value of the updated nodes. Similarly, the other updated sequence, '<cb>' is also updated in the CFM-pattern tree and form the updated CFM-tree. The updated CFM-tree with timestamp T6 is given in Fig. 2. In the CFM-tree, the newly updated node to the root node is marked as in dotted line, whereas the update process is done in the existing nodes is indicated as a thin line and the dark line represents the nodes in which there is no update is carried out. For mining the progressive CFM patterns, we have used the user specified thresholds, (min_sup >= 1, C_T <= 4, T_m>=10).

Step 4: Handling the Node Deletion Operation in the Updated CFM-tree

On mining progressive CFM sequential patterns, the newly arrived patterns may not be identified as frequent one if the static database is a larger one. It is noted that users are usually more interested in the recent data than the old ones.

So, the deletion of an item from the CFM-tree is carried out utilizing the time information stored in every node. Thus, the incompact nodes and the non-zero infrequent nodes should be deleted from the final updated CFM-tree.

Example: While deleting the obsolete sequences, the timestamp of the nodes which couldn't satisfy the time sequences as the updating process is carried out. We have deleted the incompact nodes, which don't satisfy the user specified threshold, where there is no update process are carried out. As well, we have removed the non-zero infrequent nodes in which the frequent value is less than the threshold. The CFM-tree with no incompact nodes is shown in Fig. 3. The final CFM tree without non-zero infrequent nodes is shown in Fig. 4.

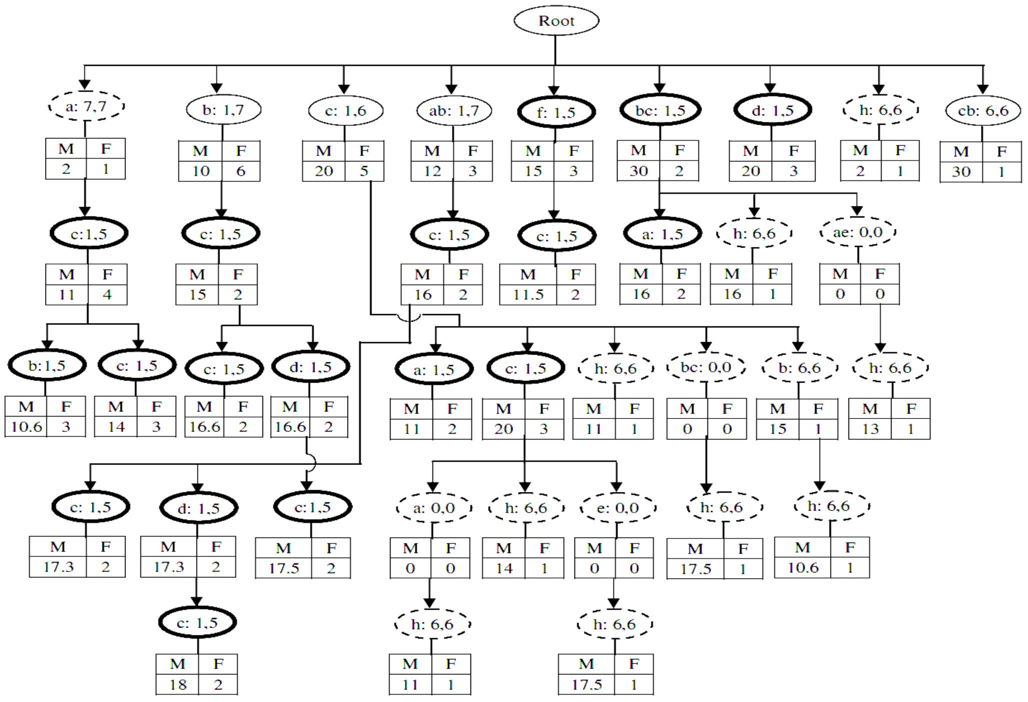

Fig. 2. Updated CFM-tree

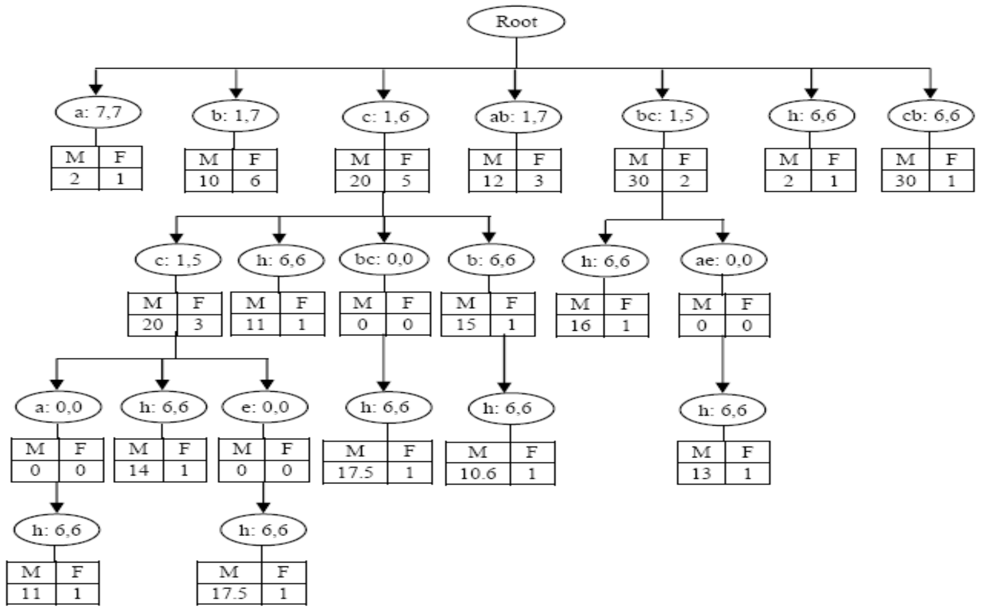

Fig. 3. Updated CFM -tree with no Incompact Nodes

Step 5: Mining of Progressive CFM Patterns from the Progressive Database using proposed tree Pattern Mining Algorithm

After the construction of updated CFM-tree, the progressive CFM patterns are mined from it based on the user specified thresholds. Here, we have designed tree pattern mining algorithm that uses the top-down process to mine the CFM-patterns. We start with the mining process from the top nodes of the CFM-tree and their corresponding paths are extracted from it. Then, by combining the nodes of each level, the progressive CFM patterns are obtained.

Example: From the final updated CFM-Tree shown in Fig.4, one of the top node <bc> and its corresponding paths are extracted. From the paths, we have combined each level of nodes so that the progressive CFM patterns, {<(bc)>, <(bc)h> <(bc)(ae)h} are obtained. Fig. 5 (a)

Shows the extracted path for the node <bc> and we combine each level with the previous level so that the CFM patterns can be achieved, shown in Fig. 5 (b), 5 (c) and 5 (d). The mined sequential CFM-patterns for all the top nodes are given in the below Table 4.

Table 4. Final progressive CFM-patterns

CFM-patterns	
<b>	<b>
<c>	<c>, <cc>, <ch> , <cb>, <c(bc)h>, <cbh>, <ccah>, <cch>, <cceh>
<(ab)>	<(ab)>
<(bc)>	<(bc)>, <(bc)h> <(bc)(ae)h>
<(cb)>	<(cb)>

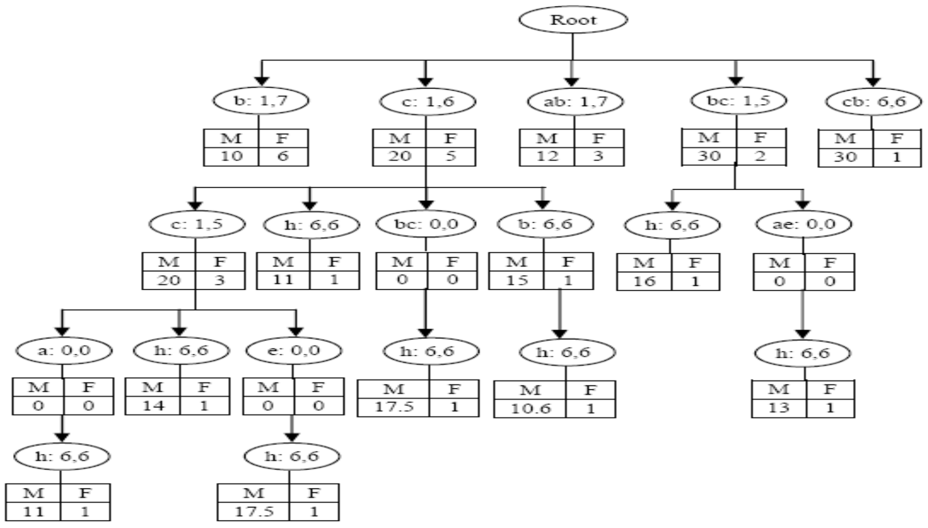

Fig.4. Final updated CFM-tree

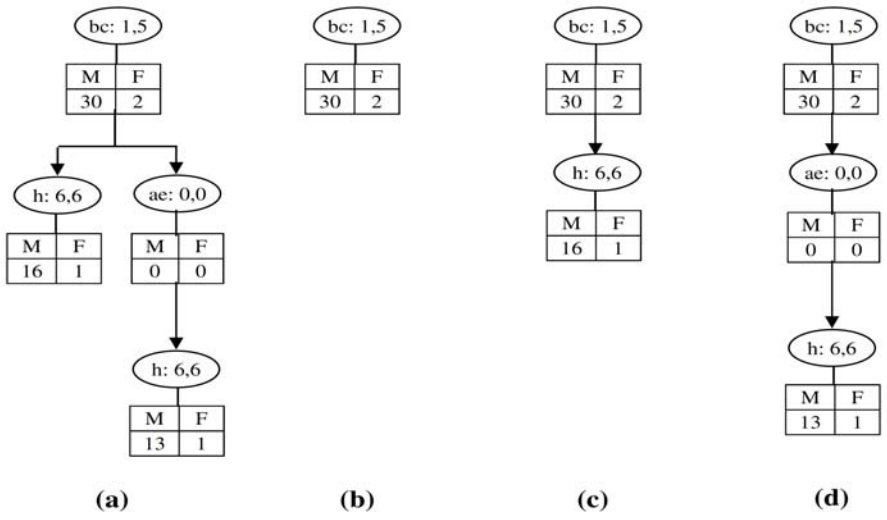

Fig.5. Mining of progressive CFM Patterns from the Updated CFM-tree

The pseudo code for the proposed procedure for mining the progressive CFM-patterns is given as follows.

Pseudo Code

Input: CFM-tree, min_sup, Tm
Output: A complete set of Progressive CFM patterns
Assumptions
m → Number of nodes (next to the root node) in the constructed CFM-tree
min_sup → Minimum support threshold
S_pat→ Sequential pattern
PCFM_pat→ Progressive CFM-patterns
k→ Number of distinct paths
D→ Depth of the path
pi→ Item information in the node

```
Begin
for each node m in CFM tree
    for ( $j = 1$ ; $j < k$ ; $j++$ )
        $d[j] =$ distinct path.CFM – tree
do_miner ( $d[j]$ )
if (support ($S\_pat[l]$) $\geq$ min_sup & $T_m$)
PCFM_pat << S_pat
end if
        end for
    end for
end
```

```
sub routine: do_miner ($d$[j])
Begin
    $p.1 = top\ node.d\ [j]$
    S_pat << p.1
    for ( $i = 1$ ; $i < D$ ; $i++$)
        $p.(i+1) = p.i \,||\, p.(i+1)$
        S_pat << $p.(i+1)$
    end for
end
```

6. Empirical Evaluation

The experimental results of the proposed algorithm for mining of progressive CFM-sequential patterns from a progressive database are described in this section. The experimental results and analysis of the CFM-sequential patterns are done with the aid of the well-known incremental IncSpan Algorithm [25].

6.1 Experimental Design

We used synthetic dataset and real life dataset to evaluate the performance of our proposed algorithm. The progressive CFM-Miner and IncSpan algorithms were implemented using Java language (jdk 1.6). The experimentation has been carried out on a 2.9 GHz, dual core PC machine with 1 GB main memory running a 32-bit version of Windows XP. The proposed algorithm execute in a distributed environment that means the updation of data records can be done from the multiple sources. So, we run the algorithm in thread environment, in which the updation of data records is done in various threads. Synthetic data set: We have generated a set of synthetic data sequence by a data generator similar in spirit to the IBM data generator designed for testing sequential pattern mining algorithms. Each data sequence contains a sequence of item sets. However, we assign different time values to the items in different item sets but the same time values to those in the same item sets. Real life datasets: We make use of the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets). This data describes the page visits of users who visited msnbc.com on September 28, 1999. Visits are recorded at the level of URL category ("frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports") and are recorded in time order.

6.2 Comparison of Incspan and Progressive CFM – Miner in Mining Meaningful Rules

Table 5. Comparison of algorithms in mining of meaningful sequences

CFM-patterns	
<b>	<b>
<c>	<c>, <cc>, <ch> , <cb>, <c(bc)h>, <cbh>, <ccah>, <cch>, <cceh>
<(ab)>	<(ab)>
c)>	<(bc)>, <(bc)h> <(bc)(ae)h>
<(cb)>	<(cb)>

Incspan	
<a>	(ab), ab, ac, (ac), ad, af, (ab)c, (ab)d, (ab)f, aba, aca, acc, adc,
<b>	(ba), ba, (bc), bc, bd, bf, (ba)c, (ba)d, (ba)f, (bc)a, bcd, bcc, bcf, bdc.
<c>	ca, cb, cd, cf, cdc.
<d>	db, dc.
<f>	fc, fb, fcb.

The above table shows the patterns mined by the proposed algorithm and the Incspan algorithm. The Incspan algorithm produced the rules based on the frequency of the items. But, the proposed algorithm mines the most utility sequences compared with the previous algorithm. From the table, we can identify that most of the sequences containing the 1-length patterns such as <b> and <c> are mined from the database using proposed algorithm. But, the previous algorithm mined the sequences of having the patterns of <a>, <d>, <g> and <e>. When analyzing these patterns, we can prove that <h> and <a> have less monetary value. On the other hand, <d>, <g> and <e> are not recently frequent items.

6.3 The Comparison between Algorithms Incspan and Progressive CFM – Miner

The performance of the proposed CFM-Miner algorithm for sequential pattern mining from the progressive database is evaluated by three standard evaluation measures. They are (a) Number of sequential patterns, i.e., the significant number of sequential patterns generated based upon the given minimum support threshold, (b) Execution time, i.e., the time taken to execute the computer program and it characteristically depends with the input size and the (c) Memory usage, i.e., the memory utilized by the current jobs present in the particular system. We have analyzed our proposed algorithm with the well known incremental algorithm, IncSpan using both the synthetic and the real life datasets.

1) Synthetic Dataset

With the help of the synthetic dataset, we have analyzed the mined CFM-sequential patterns with the IncSpan algorithm using three evaluation measures with diverse support values. We have done the analysis and plotted as a graph by computing the generated number of sequences, execution time and the memory usage with different minimum support threshold. By analyzing the plotted graphs of Fig. 6, 7 and 8 using the synthetic datasets, we have found that the proposed progressive CFM-Miner algorithm efficiently mined the sequential patterns than the incremental IncSpan algorithm. Here, the input sequences have been varied in certain time intervals. The generated number of sequences shows better results in our proposed approach is given in Fig. 6. But in Fig 7, the corresponding execution time of the CFM-Miner gets slightly slipped down in some cases than the IncSpan algorithm. The effective usage of the memory in the proposed algorithm is shown in Fig. 8.

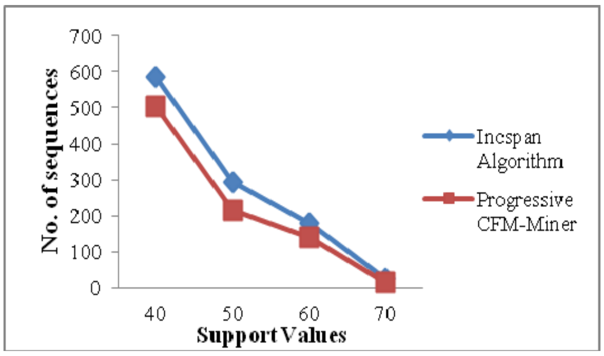

Fig.6. Generated Sequential Patterns with Different Support Values

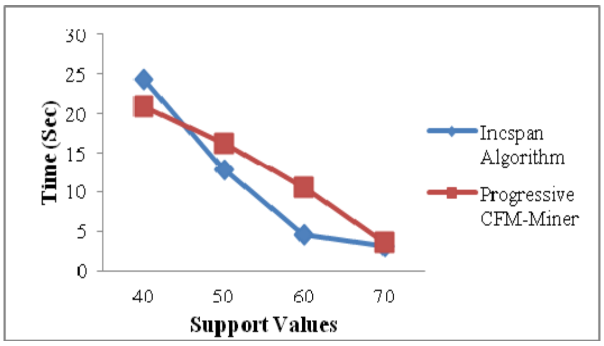

Fig.7. Execution Time Required With Different Support Values

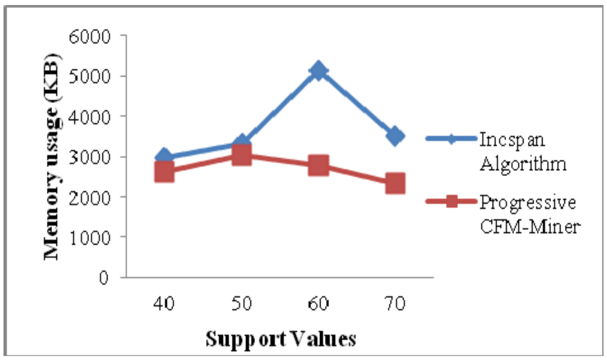

Fig.8. Memory Usages with Different Support Values

2) Real life dataset

With the aid of the Real life dataset, we have analyzed the mined CFM-sequential patterns with the IncSpan algorithm by three ways of evaluation measures with diverse support values. We have done the analysis and plotted as a graph by computing the generated number of sequences, execution time and the memory usage with different minimum support threshold. By analyzing the plotted graphs of Fig. 9, 10 and 11 using the real life datasets, we have find that the proposed progressive CFM-Miner algorithm efficiently mined the sequential patterns than the incremental IncSpan algorithm. Here, the input sequences have been varied in certain time intervals. The generated number of sequences shows better results in our proposed approach is given in Fig. 9. In Fig. 10, the corresponding execution time of the CFM-Miner shows better results than the IncSpan algorithm. The effective usage of the memory in the proposed algorithm is shown in Fig. 11.

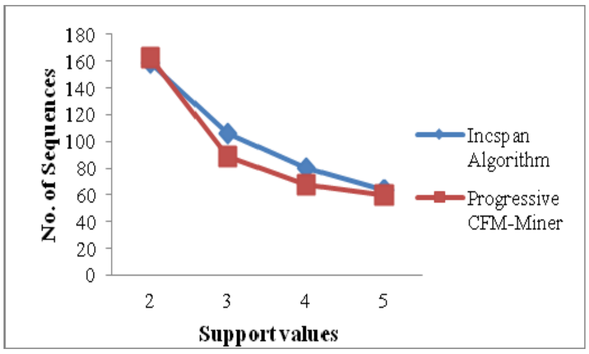

Fig.9. Generated Sequential Patterns with Different Support Values

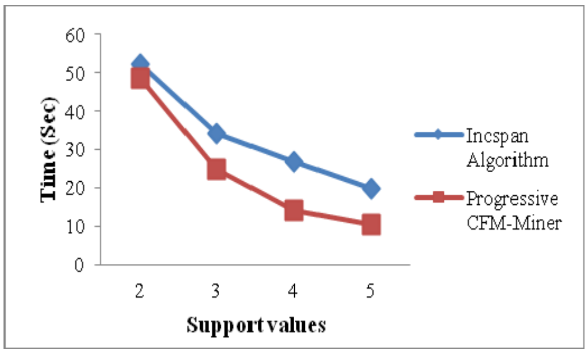

Fig.10. Execution Time Required With Different Support Values

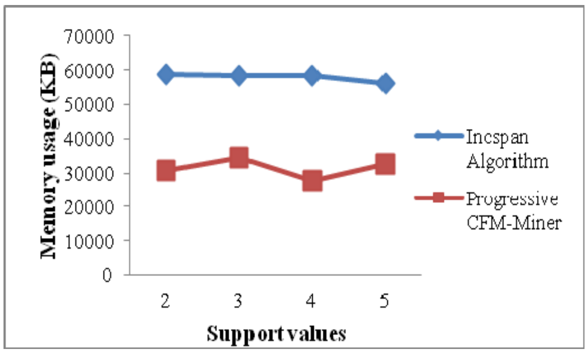

Fig.11. Memory Usages with Different Support Values

7. Conclusion

We have presented an efficient progressive CFM-miner algorithm to handle the maintenance problem of CFM-sequential patterns. We have built an updated CFM-tree using the CFM- sequential patterns obtained from the static database to control the dynamic nature of data updating process and deletion process into the sequential pattern mining problem. Subsequently, the database gets updated from the distributed database that may be static, inserted, or deleted. Whenever the database is updated from the multiple sources, CFM tree is also updated by including the updated sequence. Then, the updated CFM-tree is used to mine the progressive CFM-patterns using the proposed tree pattern mining algorithm. Eventually, the experimentation is carried out using the synthetic and real life datasets that are given to the progressive CFM-miner using thread environment. The experimental results and analysis provides better results in terms of the evaluation measures over IncSpan algorithm. The paper can be extended in two directions such as, 1) a method for solving the accessing of database to find the progressive compact sequence, 2) the extensive analysis with the different kind of datasets.

Acknowledgement

The authors would like to thank the Editor and anonymous referees for providing valuable comments and constructive suggestions. We express our gratitude towards the Galgotias College of Engineering and Technology, Greater Noida for the support being provided to complete this research work.

References

1. Cláudia Antunes, Arlindo L. Oliveira, "Generalization of Pattern-growth Methods for Sequential Pattern Mining with Gap Constraints", *Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science*, Vol: 2734, pp: 239-251, 2003.
2. M. J. Zaki, "Efficient enumeration of frequent sequences," *In Proceedings of the 7th International Conference on Information and Knowledge Management, Washington*, USA, pp. 68-75, 1998.
3. Jianyong Wang, Yuzhou Zhang, Lizhu Zhou, George Karypis, Charu C. Aggarwal, "Discriminating Subsequence Discovery for Sequence Clustering.", *Proceedings of the Seventh SIAM International Conference on Data Mining*, April 26-28, 2007, Minneapolis, Minnesota, USA 2007.
4. Jinlin Chen ,Terry Cook , "Mining contiguous sequential patterns from web logs", *In Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada*, pp: 1177 - 1178 , 2007.
5. Florent Masseglia, Pascal Poncelet and Maguelonne Teisseire, "Incremental mining of sequential patterns in large databases", *Data & Knowledge Engineering*, Vol. 46, No.1, pp. 97-121, 2003.
6. Ming-Yen Lin and Suh-Yin Lee, "Interactive Sequence Discovery by Incremental Mining", An International Journal of Information Sciences-Informatics and Computer Science, vol. 165, No. 3-4 , pp.187 - 205, October 2004.
7. Jian Pei, Jiawei Han and Wei Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", *Journal of Intelligent Information Systems*, Vol: 28, No: 2, pp: 133-160, 2007.
8. Rong She, Fei Chen, Ke Wang ,Martin Ester, Jennifer L. Gardy, Fiona S. L. Brinkman, "Frequent-subsequence-based prediction of outer membrane proteins", *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp: 436 - 445, 2003.
9. Bhawna Mallick, Deepak Garg and P. S. Grover, "Incremental Mining of Sequential Patterns- Progress and Challenges", *Intelligent Data Analysis – An International Journal*, Vol. 17, No. 3, 2013, accepted for publication.

10. David Lo and Siau-Cheng Khoo., "SMArTIC: towards building an accurate, robust and scalable specification miner.", *In Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2005, Portland, Oregon, USA, November 5-11, 2006.
11. Bhawna Mallick, Deepak Garg and P.S. Grover, "CFM-PrefixSpan: A pattern growth algorithm incorporating compactness and monetary", *International Journal of Innovative Computing, Information and Control*, ISSN 1349-4198, Volume 8, Number 7(A), July 2012, pp 4509 – 4520.
12. Ming-Yen, Lin ,Suh-Yin Lee, "Efficient mining of sequential patterns with time constraints by delimited pattern growth", *Knowledge and Information Systems*, Vol.7 , No. 4, pp.499 - 514 , 2005.
13. Jian Pei, Jiawei Han and Wei Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", *Journal of Intelligent Information Systems*, Vol. 28, No: 2, pp: 133-160, 2007.
14. Tarek Sobh, "Innovations and Advanced Techniques in Computer and Information Sciences", *Springer*, 2007, ISBN 978-1-4020-6268-1.
15. Jigyasa Bisaria, Namita Srivastav, Kamal Raj Pardasani, "A Rough Set Model for Sequential Pattern Mining with Constraints", *In Proceedings of the (IJCNS) International Journal of Computer and Network Security*, Vol. 1, No. 2, November 2009.
16. Enhong Chen, Huanhuan Cao , Qing Li , and Tieyun Qian, "Efficient strategies for tough aggregate constraint-based sequential pattern mining", *Information Sciences*, Vol. 178, No.6, pp.1498-1518, 15 March 2008.
17. Ming-Yen, Lin, Suh-Yin Lee, "Efficient mining of sequential patterns with time constraints by delimited pattern growth", *Knowledge and Information Systems*, Vol.7, No. 4, pp.499 - 514, 2005.
18. Jong Bum Lee, Minghao Piao, Jin-ho Shin, Hi-Seok Kim; Keun Ho Ryu, "ITFP: Incremental TFP for mining frequent patterns from large data sets", *In proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET)*, Chengdu, pp: V2-181 - V2-185, 2010 .
19. Ming-Yen Lin, Sue-Chen Hsueh, Chih-Chen Chan, "Incremental Discovery of Sequential Patterns Using a Backward Mining Approach", *In proceedings of the International Conference on Computational Science and Engineering, Vancouver*, BC, pp: 64 - 70, 2009.
20. Jen-Wei Huang, Chi-Yao Tseng, Jian-Chih Ou, Ming-Syan Chen, "A General Model for Sequential Pattern Mining with a Progressive Database", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 20, No: 9, pp: 1153 - 1167, 2008.
21. Lionel Vinceslas, Jean-Emile Symphor, Alban Mancheron and Pascal Poncelet, "SPAMS: a novel Incremental Approach for Sequential Pattern Mining in Data Streams", *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Vol: 292, pp: 201-216, 2010.
22. Yue Chen, Jiankui Guo, Yaqin Wang, Yun Xiong and Yangyong Zhu, "Incremental Mining of Sequential Patterns Using Prefix Tree", *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, Vol: 4426 pp: 433-440, 2007.
23. Lei Chang, Dongqing Yang, Tengjiao Wang and Shiwei Tang, "IMCS: Incremental Mining of Closed Sequential Patterns", *Advances in Data and Web Management*, Lecture Notes in Computer Science, Vol: 4505, pp: 50-61, 2007.
24. Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal and Mei-Chun Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", *IEEE transactions on Knowledge and Data Engineering*, Vol. 16, No. 10, October 2004.
25. Hong Cheng, Xifeng Yan, Jiawei Han, "IncSpan: incremental mining of sequential patterns in large database", *In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.
26. Maragatham G & Lakshmi M, (2011), "A weighted Particle Swarm Optimization Technique for optimizing association rules ", *4th International Conference on Recent trends in Computing, communication and information technologies*, Dec 9 -11, 2011. Proceedings published by Springer (LNCS) - Communications in Computer and Information Science (CCIS**)** Series part II, pp: 675-684.
27. George Aloysius, D. Binu, "An approach to products placement in supermarkets using PrefixSpan algorithm", *Journal of King Saud University - Computer and Information Sciences*, in press, July 2012.