Maximal and closed frequent itemsets mining from uncertain database and data stream

Maliha Momtaz*, Abu Ahmed Ferdaus and Chowdhury Farhan Ahmed

Department of Computer Science and Engineering,
University of Dhaka,
Dhaka, 1000, Bangladesh
Email: malihachaity1089@gmail.com
Email: ferdaus1167@gmail.com
Email: farhan@du.ac.bd
*Corresponding author

Mohammad Samiullah

Faculty of Information Technology,
Monash University,
Melbourne, 3800, Australia
Email: samiullah@cse.univdhaka.edu

Abstract: Frequent itemsets (FIs) mining from uncertain database is a very popular research area nowadays. Many algorithms have been proposed to mine FI from uncertain database. But in typical FI mining process, all the FIs have to be mined individually, which needs a huge memory. Four trees are proposed in this paper which are: (i) maximal frequent itemset from uncertain database (MFU) tree which contains only the maximal frequent itemsets generated from uncertain database, (ii) closed frequent itemset from uncertain database (CFU) tree which contains only closed frequent itemsets generated from uncertain database, (iii) maximal frequent itemset from uncertain data stream (MFUS) tree which contains maximal frequent itemsets generated from uncertain data stream and (iv) closed frequent itemset from uncertain data stream (CFUS) tree which contains closed frequent itemsets generated from uncertain data stream. Experimental results are also presented which show that maximal and closed frequent itemsets mining requires less time and memory than typical frequent itemsets mining.

Keywords: FI; frequent itemset; uncertain database; FU; frequent itemset from uncertain database; MFI; maximal frequent itemset; CFI; closed frequent itemset; MFU; maximal frequent itemset from uncertain database; CFU; closed frequent itemset from uncertain database; FUS; frequent itemset from uncertain data stream; MFUS; maximal frequent itemset from uncertain data stream; CFUS; closed frequent itemset from uncertain data stream.

Reference to this paper should be made as follows: Momtaz, M., Ferdaus, A.A., Ahmed, C.F. and Samiullah, M. (2019) 'Maximal and closed frequent itemsets mining from uncertain database and data stream', *Int. J. Data Science*, Vol. 4, No. 3, pp.237–259.

Biographical notes: Maliha Momtaz has received both her Bachelor's and Master's degree from the Department of Computer Science and Engineering, University of Dhaka. She is interested in machine learning, data mining and artificial intelligence.

Abu Ahmed Ferdaus received his Bachelor's degree from the Department of Applied Physics and Electronics, University of Dhaka and Master's degree from the Department of Computer Science and Engineering, University of Dhaka. His research interests are database administration and management, data warehousing, data mining, machine learning, neural networks and artificial intelligence. Currently, he is an Associate Professor in the Department of Computer Science and Engineering, University of Dhaka.

Chowdhury Farhan Ahmed received both his Bachelor's and Master's degree from the Department of Computer Science and Engineering, University of Dhaka. His research interests are data mining and knowledge discovery, machine learning, stream data management, web mining, correlation analysis, database management and information retrieval. Currently, he is a Professor in the Department of Computer Science and Engineering, University of Dhaka.

Mohammad Samiullah received both his Bachelor's and Master's degree from the Department of Computer Science and Engineering, University of Dhaka. His research interests are data mining and knowledge discovery, machine learning, web mining, correlation analysis, database management, information retrieval, bioinformatics, Bayesian networks, Bayesian statistics and artificial intelligence. Currently, he is a PhD student at the Faculty of Information Technology, Monash University, Melbourne, Australia.

1 Introduction

Increasing applications of web searching, telecommunication system, retail system, various sensor networks, geospatial data, market basket analysis, medical diagnosis system etc. are continuously producing a huge streams of uncertain data where data are partial, inaccurate and confusing. So mining quality information as well as sidestepping spurious patterns from the large uncertain databases have been a major concern for decision making in real life. Data mining paves the way for solution to this problem by introducing mining maximal and closed frequent itemsets. While mining FI from uncertain database, the main problem is huge number of itemsets are generated. So huge amount of memory is required. For example, if there is a frequent itemset with size l, then all 2^l nonempty subsets of the itemset have to be generated. For certain database in which items have no probability value, the subsets of a maximal frequent itemset is also frequent.

Let, $I = \{a, b, c, d\}$ is frequent. $S = \{b, c\}$ is a subset of I. Then support of S will be greater than or equal to support of I. That means all non-empty subsets of a frequent pattern are also frequent. This property is known as downward closure property. Expected support(expSup) satisfies this property as well. Conversely, if a pattern is infrequent, then none of its supersets can be frequent. If Y is a subset of X, then

- expSup (X) >= minsup implies expSup (Y) >= minsup
- expSup $(Y) <$ minsup implies expSup $(X) <$ minsup.

All the frequent itemsets can be represented by MFU. It will require less memory space. For a particular database, the number of CFU is greater than the number of MFU but less than the number of FU.

Recent emerging applications such as network traffic analysis, Web click stream mining, power consumption measurement, sensor network data analysis and dynamic tracing of stock exchange data call for efficient techniques to extract useful information from streams of data. Data streams are continuous and unbounded. Stream data management systems and continuous stream query processors are under popular investigation and development. Because users are usually more interested in recent data than older ones.

1.1 Motivating examples

Let us consider a medical diagnosis database where a sample record may look like {fever: 0.7, typhoid: 0.6, jaundice: 0.4}. Frequent itemsets like {fever, typhoid}: 0.42 and {fever, jaundice}: 0.28 can be generated from here. For small dataset, it does not bother that much if such itemsets are generated. But for a huge dataset, if itemsets generation keeps going, then it creates an issue with storage. It will need huge memory spaces to store these itemsets. In such cases, it is necessary to reduce the number of itemsets generated from database. These three different itemsets can be represented by a single itemset {fever, typhoid, jaundice}: 0.168 (Here 0.168 is the expected support of {fever, typhoid, jaundice}. Expected support is explained later). Such itemsets are called maximal frequent itemsets. In this paper, a way to mine maximal frequent itemsets from uncertain database and data stream has been proposed.

Now, it is known that itemsets {fever, typhoid, jaundice}: 0.168 can be used to generate all the frequent itemsets. Unfortunately, only frequent itemsets like {fever, typhoid}, {fever, jaundice} etc can be generated. But the existential support of these itemsets can not be derived. In this case, the frequent itemsets with same existential support are needed to be mined in order to reduce the number of generated frequent itemsets. Such itemsets are called closed frequent itemsets. As it is an uncertain database, the support of an itemset can not be determined in the usual way and no such techniques have yet been generated to solve this problem. Here, a way to generate closed frequent itemsets from uncertain database and data stream has been proposed.

1.2 Contributions

Main contributions of this paper are as follows:

- MFU tree and MFU-growth algorithm which can mine maximal frequent itemsets from uncertain database.
- CFU tree and CFU-growth algorithm which can mine closed frequent itemsets from uncertain database.
- MFUS tree and MFUS-growth algorithm which can mine maximal frequent itemsets from uncertain data stream.

- CFUS tree and CFUS-growth algorithm which can mine closed frequent itemsets from uncertain data stream.

The remainder of this paper contains structures of proposed trees and algorithms, experimental evaluation of these algorithms, conclusion and future works.

2 Related works

As this field is getting more and more attention ,researchers have proposed different algorithms to mine frequent patterns from uncertain databases. At first Apriori (Agrawal and Srikant, 1994) algorithm was proposed to mine association rules. Then Han et al. (1994) proposed the frequent pattern tree (FP tree) which introduces FP Growth algorithm for efficiently deriving the frequent itemsets without candidate itemsets generation. FP-Growth is an improved version of Apriori algorithm. Both of these algorithms work for certain database.

But there are situations in which the presence or absence of an item is uncertain. For example, a doctor is not sure about having cancer of a patient. He can suspect that the patient has 70% probability of having cancer. This probability value is called existential probability. In uncertain transaction database, each item has an existential probability value which expresses the probability of that item to be present in that transaction.

There are many existing algorithms for mining frequent itemsets from uncertain database. At first U-Apriori (Chui et al., 2007) algorithm was proposed. But as it is an Apriori based algorithm, it requires multiple database scans. To remove this problem, UF-growth (Leung et al., 2008) algorithm was proposed later which introduces UF-tree structure. In UF-tree, paths are shared only if tree nodes have same item and same existential probability, which does not produce a compact tree structure. To make compact tree structure, UFP-growth (Aggarwal et al., 2009) algorithm was proposed. But the problem is, this algorithm produces some false positives. Then PUF-growth (Leung and Tanbeer, 2013) algorithm was proposed which introduces PUF-tree (a prefix-capped uncertain frequent pattern tree) structure. PUF-tree not only makes a compact tree structure but also removes all false negatives and false positives.

To find maximal frequent itemsets from certain database, at first MAXMINER (Bayardo, 1998) algorithm was proposed, which extends Apriori algorithm. Then DEPTHPROJECT (Agarwal et al., 2000) was proposed, which outperformed MAXMINER. By extending the idea of DEPTHPROJECT, an algorithm MAFIA (Burdick et al., 2001) (a maximal frequent itemsets algorithm for transactional databases) was proposed. After that using a novel technique called progressive focusing, GENMAX (Gouda and Zaki, 2001) algorithm was proposed. GENMAX outperforms other existing algorithms on some types of datasets. Later FPMAX (Grahne and Zhu, 2003a, 2003b, 2005) algorithm was introduced, which is an extension of the FP-growth algorithm. FPMAX introduces a tree structure called MFI-tree (Maximal Frequent itemsets tree) which contain only maximal frequent itemsets. FPMAX performs better than MAFIA and GENMAX. Apart from these, there are other works done on mining MFIs. INLA-MFP (improved N-list-based algorithm for mining maximal frequent patterns) (Vo et al., 2017) algorithm was proposed in which an N-list structure was used to compress the dataset for mining maximal frequent patterns. INLA-MFP algorithm showed an improved runtime and memory usage. Then a spark-based approach for mining maximal frequent patterns was proposed (Karim et al., 2018), in which a prime number based data transformation was utilised to handle the problems with null transactions.

For mining closed frequent itemsets from certain database, at first CLOSET (Pei et al., 2000) and then CHARM (Zaki and Hsiao, 2002) algorithm was proposed. After that, CLOSET+ (Wang et al., 2003) algorithm was proposed, which followed FP-tree approach. Later FPCLOSE (Grahne and Zhu, 2003b, 2005) algorithm was proposed, which introduces a tree structure called CFI-tree (Closed Frequent itemset tree). FPCLOSE performed better than CLOSET+. After that, a compact data structure named closed enumeration tree (CET) (Chi et al., 2006) was proposed in which a synopsis data structure was designed to monitor transactions in the sliding window so that the current CFIs can be outputted at any time. Then a new memory efficient dynamic superset bit-vector (DSBV) structure (Hashem et al., 2017) was proposed which is an efficient approach to mine CFIs and their lattice structure.

3 The PUF-tree construction

The PUF-tree node stores the maximum existential probability value of the prefix from that node up to the root (the item cap of the item). Every node of a PUF-tree contains

- an item
- its prefixed item cap.

Distinct frequent items are found and a header table called I-list is constructed to store only frequent items in consistent order with 1st DB scan. With 2nd DB scan the PUF-tree is constructed. From a database consisting of n uncertain transactions, and a predefined minimum support thresholds, the algorithm first calculates the expected support count of each item I (expSupCount(I)) in the uncertain database. Then it checks whether the value of expSupCount(I) for each item I is larger than or equal to the predefined minimum count, n*s. If expSupCount(I) satisfies the above condition, I is an uncertain frequent item and puts I with its expSupCount(I) in the set of large frequent 1-itemsets (L1). Next it sorts the items in L1 according to their transaction frequencies. Then it builds the header table by keeping the 1-itemsets (items) in L1 in the sorted order. It initially sets the root node of the PUF tree as root. Then the tree removes the items which are not in L1, from the transactions and sort the remaining items in each transaction according to the sorted order. Tree construction process is similar to FP-tree. The difference is that when inserting a transaction item, first its item cap is computed and then inserted into the PUF-tree according to I-list order. If that node already exists in the path, its item cap is updated by adding the computed item cap to the existing item cap. Otherwise a new node is created with this item cap value. Finally frequent itemsets are found from the tree in a recursive manner. PUF-tree is shown in Figure 1 for the database presented in Table 1.

4 Proposed algorithms

At first, PUF growth algorithm (PUF tree) is used to generate conditional trees. PUF tree is used for mining frequent itemsets from uncertain database. For mining maximal and close frequent itemsets from uncertain database, different algorithms are revisited among which FPmax (MFI tree) and FPclose (CFI tree) are chosen. DS tree (Leung and Khan, 2006) is used for mining frequent itemsets from data stream.

Figure 1 PUF tree

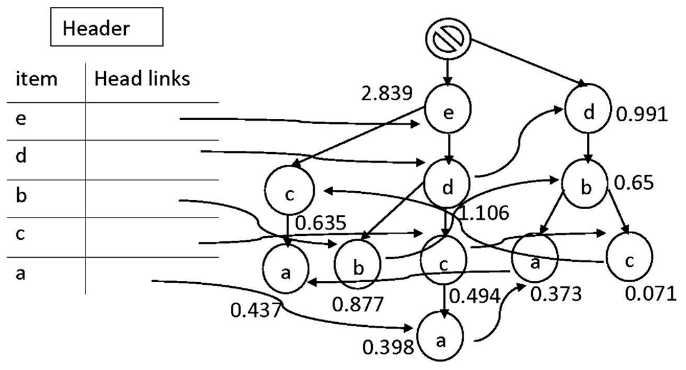


Table 1 Database

TID	Items with existential probability
t1	a : 0.465, c : 0.855, d : 0.578, e : 0.498
t2	b : 0.657, d : 0.488, e : 0.745
t3	a : 0.454, b : 0.821, d : 0.656, f : 0.113
t4	b : 0.432, d : 0.506, e : 0.898
t5	b : 0.332, c : 0.212, d : 0.335
t6	a : 0.48, c : 0.91, e : 0.698

Expected support (expSup)

In uncertain databases, the expected support count of an itemset is used instead of the actual support count of the precise databases. For example, a satellite image processor identifies the presence of any object in the earth or space. However, due to the limitation of bandwidth access, image resolution, noisy electronic media and changing features of spatial area and environment, the presence of the target object can be expressed as expected support or probability.

The expected support count of an itemsets I, in an uncertain database is calculated as,

$$expSup(I) = \sum_{i=1}^{|UDB|} \left(\prod_{x \in I} p(x, t_i) \right)$$

where |UDB| is the number of transactions in uncertain database and $p(x, t_i)$ is the existential probability of an item x in a transaction t_i. An itemsets is considered frequent if its expected support satisfies a predefined threshold.

Prefixed item cap $I^{Cap}(x_r, t_j)$

The prefixed item cap $I^{Cap}(x_r,\ t_j)$ of an item x_r in a transaction $t_j = x_1, \ldots,$ $x_r, \ldots, x_h$, where $1 \le r \le h$, is defined as the product of $P(x_r,\ t_j)$ and the highest existential probability value M of items from x_1 to x_{r1} in t_j (i.e., in the proper prefix of x_r in t_j):

$$I^{Cap}(x_r, t_j) = \begin{cases} P(x_r, t_j) \times M, & \text{if } h > 1 \\ P(x_1, t_j) & h = 1, \end{cases} \text{ where } M = max_{1 \le q \le r-1} P(x_q, t_j)$$

Cap of expected support $expSup^{Cap}$(X)

The cap of expected support $expSup^{Cap}(X)$ of a pattern $X = x_1, \ldots, x_k$ (where $k > 1$) is defined as the sum (over all n transactions in a database) of all item caps of x_k in all the transactions that contain X:

$$expSup^{Cap}(X) = \sum_{j=1}^{n} \left\{ I^{Cap}(x_k, t_j) | X \subseteq t_j \right\}$$

$expSup^{Cap}(X)$ for any k-itemsets $X = x_1, \ldots, x_k$ can be considered as an upper bound to the expected support of X, i.e., expSup(X) $\le expSup^{Cap}(X)$. So, if $expSup^{Cap}(X)$ $< minsup$, then X cannot be frequent. Conversely, if X is a frequent pattern, then $expSup^{Cap}(X)$ must be $\ge minsup$.

In this paper, every itemset is represented in this pattern: {itemset} : $expSup^{Cap}$: expSup.

4.1 Proposed MFU-tree and MFU-growth algorithm

Maximal frequent itemsets from uncertain database are mined using MFU-growth algorithm which introduces Maximal frequent itemsets from uncertain database tree (MFU tree). The idea of mining maximal itemsets is taken from FPmax algorithm. As it is for uncertain database, the technique of mining conditional trees from PUF tree structure has been followed.

Definition 1 (maximal branch)**:** A frequent itemsets is called a maximal branch if it contains all the items of that transaction. Maximal branches form maximal frequent itemsets.

Example 1: Let's consider the transaction t = {e: 0.498, d: 0.578, c: 0.855, a: 0.465}. Maximal branch consists of all the items of a transaction and it's expected support cap ($expSup^{Cap}$) value is similar to Item Cap (I^{Cap}) of that item which has the lowest expected support in the database of which it belongs to. Assuming that item a has the lowest expected support in the database, it's I^{Cap} value is I^{Cap} $\{a\}$ = 0.465 * 0.855 = 0.398 in this transaction. So, maximal branch generated from t is $\{e, d, c, a\}$: 0.398 ($expSup^{Cap}$ of $\{e, d, c, a\}$): 0.114 (expSup of $\{e, d, c, a\}$).

Definition 2 (max positive list)**:** Maximal frequent itemsets found from each conditional tree, are kept in max positive list. Itemsets in max positive list have three parts, itemset: $expSup^{Cap}$: expSup.

Steps for mining MFU

The steps to mine MFU are described below:

$Step$ 1: With first database scan expected support for each item is derived. Infrequent itemsets are removed and individual frequent items are found.

$Step$ 2: Only the frequent items of each transaction with their respective item cap values are inserted according to the I-list order and PUF tree is built.

$Step$ 3: Projected databases and conditional trees for each item has to be built from PUF tree. Mining starts from the bottom of the I-list. Actually this maximal frequent itemsets mining process is followed by PUF Growth algorithm. The difference is that, PUF tree mines all the frequent itemsets from each item's conditional tree. But here only the maximal branches are mined.

$Step$ 4: Maximal branches for each item will be generated from their respective conditional trees. As they carry expSupCap (which is upper bound of expSup), some false positives may be generated. To overcome this problem, the actual expSup of these maximal branches are calculated with third database scan and itemsets which don't satisfy minimum support are removed.

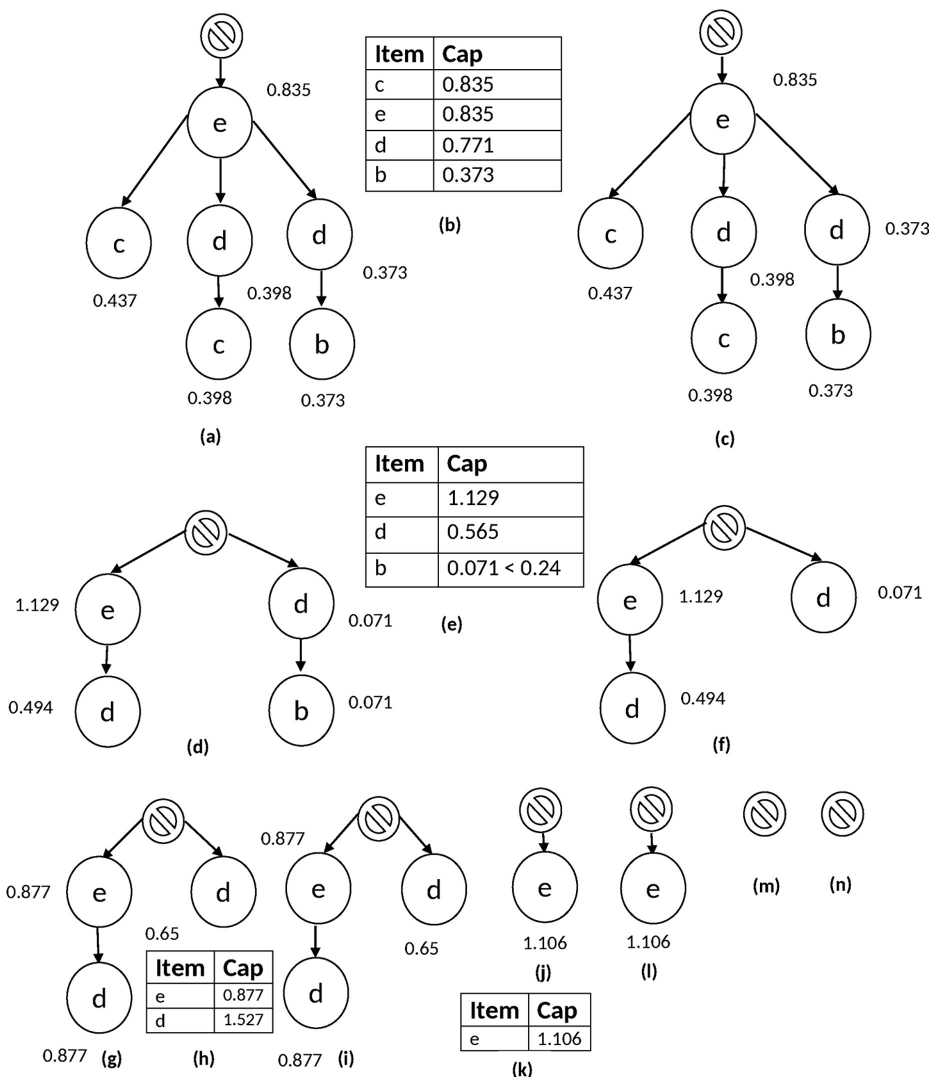
$Step$ 5: Finally MFU tree will be generated from the found maximal branches. The actual MFU will be found from MFU tree. During MFU tree building, highest length itemsets from each conditional tree are to be inserted first. Example, If maximum 4 length itemsets are generated from a conditional tree, then 4-length itemsets are to be taken first. Then 3-length, 2-length, 1-length itemsets are inserted. Before inserting maximal branches into the MFU tree, first it is needed to be checked whether it is a subset of already inserted maximal itemsets into the tree. It is called subset_testing. For subset_testing, each time a new maximal branch is inserted into the MFU tree, a copy of most recent MFU is kept. Any new maximal branch will be compared to the copy first. Only if the new branch is not a subset of existing MFU, it will be inserted. Otherwise not. All sets are ordered according to the header table, so this subset_testing can be done in linear time. Overlapping itemsets are represented by sharing prefixes of the corresponding branches to make it compressed (similar to FP-tree or PUF-tree). In an MFU tree, each node has three fields: item-name, level and node-link. All nodes with same item-name are linked together. An MFU tree also has a header table. It is constructed based on the item order in the I-list of header table of the PUF tree it is associated with. Each entry in the header table consists of two fields: item-name and head of a linked list. The head points to the first node with the same item-name in the MFU tree.

4.1.1 MFU tree building and mining maximal frequent itemsets from uncertain database

a – projected database and a – conditional tree generated from PUF tree is shown in Figure 2(a) and (c). Every item in a – projected database has a cap value greater than minimum support (Figure 2(b)). So, no item is eliminated. $\{e, c, a\}$: 0.437($expSup^{Cap}$) : 0.5(expSup) $>$ 0.24 (minsup) is generated from branch 1 of a – conditional tree. Similarly, $\{e, d, c, a\}$: 0.398 : 0.114 $<$ 0.24 is generated from branch 2 and $\{d, b, a\}$: 0.373 : 0.245 $>$ 0.24 is generated from branch 3. Among them, $\{e, d, c, a\}$ is infrequent and eliminated. $\{e, c, a\}$ and $\{d, b, a\}$ will be inserted into a – max positive list.

In c – projected database (Figure 2(d)), node b has cap value 0.071 which is less than minsup 0.24 (Figure 2(e)). So, b will be eliminated from c – conditional tree (Figure 2(f)). $\{e, d, c\}$: 0.494 : 0.246 $>$ 0.24 is generated from branch 1, $\{d, c\}$: 0.565 : 0.565 $>$ 0.24 is generated from branch 2 and both are frequent. So, c – max positive list contains $\{e, d, c\}$ and $\{d, c\}$.

Figure 2 Conditional trees generated from PUF tree



Item	Cap
c	0.835
e	0.835
d	0.771
b	0.373

Item	Cap
e	1.129
d	0.565
b	0.071 < 0.24

Item	Cap
e	0.877
d	1.527

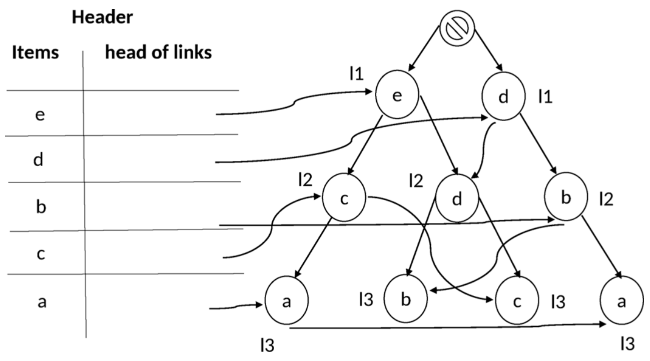
Item	Cap
e	1.106

b – projected database and b – conditional tree generated from PUF tree is shown in Figure 2(g) and (i), respectively. Every item in b - projected database has a cap value greater than minimum support (Figure 2(h)). So, no item is eliminated. $\{e, d, b\}$: 0.877 : 0.435 $>$ 0.24 is generated from branch 1 and $\{d, b\}$: 1.527 : 1.189 $>$ 0.24 is generated from branch 2 of b – conditional tree. None of them are infrequent. So, b – max positive list contains $\{e, d, b\}$: 0.877 : 0.435 and $\{d, b\}$: 1.527 : 1.189.

In d – projected database (Figure 2(j)), there is only one item e which has cap value greater than minimum support (Figure 2(k)). The only itemset generated from d–conditional tree (Figure 2(l)) is $\{e, d\}$: 1.106 : 1.106 $>$ 0.24 which is frequent and inserted into d – max positive list. e – projected database and e – conditional tree is shown in Figure 2(m) and (n), respectively. No itemset is found from e – max positive list.

To build MFU tree, at first $\{e, c, a\}$ and $\{d, b, a\}$ is taken from a - max positive list and inserted into MFU tree. $\{e, d, c\}$ is taken from c – max positive list and inserted into the tree. $\{d, c\}$ will not be inserted as it has superset $\{e, d, c\}$ which already exists in MFU tree. $\{e, d, b\}$ is taken from b – max positive list and inserted into the tree. $\{d, b\}$ will not be inserted as it has superset $\{e, d, b\}$. $\{e, d\}$ is taken from d - max positive list but will not be inserted into the tree as it has superset $\{e, d, b\}$. So, the final MFU found are $\{e, c, a\}$, $\{d, b, a\}$, $\{e, d, c\}$ and $\{e, d, b\}$. The final MFU tree is shown on Figure 3.

Figure 3 MFU tree



4.2 *Proposed CFU tree and CFU-growth algorithm*

Closed frequent itemsets from uncertain database are mined from closed frequent itemsets from uncertain database tree (CFU tree). The idea of mining closed itemsets is taken from FPclose algorithm, which uses Closed Frequent itemsets tree (CFI tree). As it is for uncertain database, here the technique of mining conditional trees from PUF tree structure has been followed.

Definition 3 (closed positive list)**:** Closed frequent itemsets found from each conditional tree, are kept in closed positive list. Itemsets in closed positive list have three parts, $\{itemset\}$: $expSup^{Cap}$: expSup.

Definition 4 (closed range)**:** If two items' or itemsets' $expSup^{Cap}$ difference is in a fixed range, then the items or itemsets are considered to have similar $expSup^{Cap}$ and the fixed range is called closed range. Closed range is user defined. The absolute value difference of the items' or itemsets' $expSup^{Cap}$ must be taken to calculate closed range. Items which are in closed range, form closed branches or closed itemsets.

Example 2 (closed range)**:** Let's assume, $expSup^{Cap}\{a\}$ = 1.944, $expSup^{Cap}\{a, b\}$ = 2.198, $expSup^{Cap}\{b, c\}$ = 1.464, closed range = 0.40. $|expSup^{Cap}\{a\}$ –

$expSup^{Cap}\{a, b\}|$ = $|1.944 - 2.198|$ = 0.254 $<=$ 0.40. So, $\{a\}$ and $\{a, b\}$ have same $expSup^{Cap}$ and they are in closed range. Again, $|expSup^{Cap}\{b\}$ – $expSup^{Cap}\{b, c\}|$ = $|2.198 - 1.464|$ = 0.734 $>$ 0.40. So, $\{b\}$ and $\{b, c\}$ have different $expSup^{Cap}$ and they are not in closed range.

Definition 5 (closed branch)**:** A frequent itemset generated from a transaction is called a closed branch, if the items of that itemsets have similar $expSup^{Cap}$ (i.e., they are in closed range). Closed branches form closed frequent itemsets.

Example 3 (closed branch)**:** Let's consider the transaction t = $\{e : 0.498, d : 0.578, c : 0.855, a : 0.465\}$ and closed range = 0.40. To find the close branch, at first the I^{Cap} value of each node has to be calculated. The I^{Cap} of each node is similar to the I^{Cap} value of the node which has the lowest expected support in its corresponding database. Assuming that item a has the lowest expected support in its corresponding database. $I^{Cap}\{a\}$ = 0.465 * 0.855 = 0.398. So the transaction can be rewritten as t = $\{e : 0.398, d : 0.398, c : 0.398, a : 0.398\}$. $|e(0.398) - d(0.398)|$ = 0 $<=$ 0.40. So, e and d has the same $expSup^{Cap}$. $|d(0.398) - c(0.398)|$ = 0 $<=$ 0.40. So, d and c has same $expSup^{Cap}$. $|c(0.398) - a(0.398)|$ = 0 $<=$ 0.40. So, c and a has same $expSup^{Cap}$. The closed branch is $\{e, d, c, a\}$: 0.398 : 0.114.

Steps for mining CFU

The steps to mine CFU are described below:

$Step$ 1: With first database scan expected support for each item is derived. Infrequent itemsets are removed and individual frequent items are found.

$Step$ 2: Only the frequent items of each transaction with their respective item cap values are inserted according to the I-list order and PUF tree is built.

$Step$ 3: Projected databases and conditional trees for each item has to be built from PUF tree. Mining starts from the bottom of the I-list. Actually these closed frequent itemsets mining process is followed by PUF tree. The difference is that, PUF tree mines all the frequent itemsets from each item's conditional tree. But here only the closed branches are mined.

$Step$ 4: Closed branches for each item will be generated from their respective conditional trees. As they carry $expSup^{Cap}$ (which is upper bound of expSup), some false positives may be generated. To overcome this problem, the actual expSup of these closed branches are calculated with third database scan and itemsets which don't satisfy minimum support are removed.

$Step$ 5: Finally CFU tree will be generated from the found closed branches. The actual CFU will be found from CFU tree. The CFU tree always stores all already found CFU and their counts. A newly found closed branch only needs to be compared with the previous CFU. If there is no superset of the new closed branch that has same $expSup^{Cap}$, then it is a CFU. In CFU tree, each node has four fields : item-name, expSup, node-link and level. The order of the items in a CFU tree header table is same as the order of items in I-list of header table of its corresponding PUF tree. While inserting a new item or itemsets in CFU tree, the expSup of a node is not incremented. It is always replaced by the maximum expSup up-to-date. During CFU tree building, highest length itemsets from each conditional tree are to be inserted first.

Example, If maximum 4 length itemsets are generated from a conditional tree, then 4-length itemsets are to be taken first. Then 3-length, 2-length, 1-length itemsets are inserted. Each time a new closed branch into the CFU tree is inserted, a copy of this most recent CFU is kept. Any new branch will be compared with the copy first. It is called subset_testing. Only if the new set is not subset of the copy and does not have same $expSup^{Cap}$, the new CFU will be inserted into the CFU tree. All sets are ordered according to the header table, so this subset_testing can be done in linear time. After inserting all itemsets, single items must be checked in the same way and inserted into CFU tree. Single items checking will start from bottom of the I-list of the associated header table of PUF tree.

4.2.1 CFU tree building and mining closed frequent itemsets from uncertain database

Same database of Table 1 is used to show closed frequent itemsets mining and PUF tree is generated (Figure 1). Let's assume, closed range is 0.40. $\{e, c, a\}$: 0.437 : 0.5 $>$ 0.24 is generated from branch 1 of a – conditional tree (Figure 2(c)). $expSup^{Cap}$ difference of e and c is $|e(0.835) - c(0.437)|$ = 0.398 <= 0.40. As, $expSup^{Cap}$ difference of e and c is less than or equal to 0.40, they have similar $expSup^{Cap}$. So, a single itemset $\{e, c, a\}$: 0.437 : 0.5 $>$ 0.24 is generated from branch 1. $\{e, d, c, a\}$ is generated from branch 2. e and d has $expSup^{Cap}$ difference $|e(0.835) - d(0.398)|$ = 0.437 $>$ 0.40, which is greater than closed range. So, $\{e, a\}$: 0.835 : 0.57 $>$ 0.24 is generated. d and c has $expSup^{Cap}$ difference $|d(0.398) - c(0.398)|$ = 0 <= 0.40. So, itemsets $\{e, d, c, a\}$: 0.398 : 0.114 $<$ 0.24 is generated. But $\{e, d, c, a\}$ is infrequent and eliminated. In branch 3, d and b has $expSup^{Cap}$ difference $|d(0.373) - b(0.373)|$ = 0 <= 0.40. So, itemsets $\{d, b, a\}$: 0.373 : 0.245 $>$ 0.24 is generated. a - closed positive list contains $\{e, c, a\}$, $\{e, a\}$ and $\{d, b, a\}$.

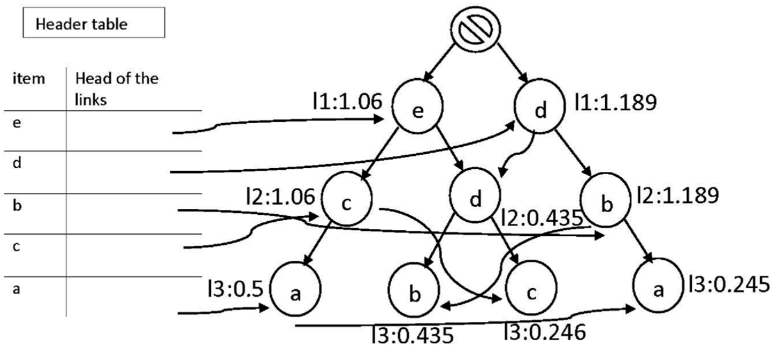
Branch 1 of c - conditional tree (Figure 2(f)) have two nodes: e (cap 1.129) and d (cap 0.494). $expSup^{Cap}$ difference of e and d is $|e(1.129) - d(0.494)|$ = 0.635 $>$ 0.40. So, separate itemsets $\{e, c\}$: 1.129 : 1.06 $>$ 0.24 and $\{e, d, c\}$: 0.494 : 0.246 $>$ 0.24 are generated from branch 1. $\{d, c\}$: 0.565 : 0.565 $>$ 0.24 is generated from branch 2. c - closed positive list contains $\{e, c\}$, $\{e, d, c\}$ and $\{d, c\}$.

Branch 1 of b - conditional tree (Figure 2(i)) have two nodes: e (cap 0.877) and d (cap 0.877). $expSup^{Cap}$ difference of e and d is $|e(0.877) - d(0.877)|$ = 0 <= 0.40. So, single itemset $\{e, d, b\}$: 0.877 : 0.435 $>$ 0.24 is generated from branch 1. $\{d, b\}$: 1.527 : 1.189 $>$ 0.24 is generated from branch 2. b - closed positive list contains $\{e, d, b\}$: 0.877 : 0.435 and $\{d, b\}$: 1.527 : 1.189.

The only branch of d – conditional tree (Figure 2(l)) contains node e (cap 1.106) and only itemset generated from d – conditional tree is $\{e, d\}$: 1.106 : 1.106 $>$ 0.24, which is inserted into d – closed positive list. No itemset is found from e – closed positive list.

After that, CFU tree is built. $\{e, c, a\}$ and $\{d, b, a\}$ are taken from a – closed positive list and inserted into CFU tree. $\{e, a\}$ is not inserted as it has superset $\{e, c, a\}$ with same $expSup^{Cap}$, $|\{e, a\}(0.835) - \{e, c, a\}(0.437)|$ = 0.398 <= 0.4. $\{e, d, c\}$ and $\{e, c\}$ is taken from c – closed positive list and inserted into the tree. As $\{e, c\}$ has greater expSup (1.06) than that of $\{e, c, a\}$ (0.5), the expSup of node e and c will be replaced by 1.06 (previously it was 0.5). $\{d, c\}$ is not inserted as it has superset $\{e, d, c\}$ with same $expSup^{Cap}$, $|\{d, c\}(0.565) - \{e, d, c\}(0.494)|$ = 0.071 <= 0.4. $\{e, d, b\}$ and $\{d, b\}$ is taken from b – closed positive list and inserted into CFU tree. While inserting $\{e, d, b\}$, expSup of node d is replaced by 0.435 (previously it was 0.246). While inserting $\{d, b\}$, expSup of node d and b is replaced by 1.189 (previously it was 0.245). $\{e, d\}$ is taken from d – closed positive list but not inserted into CFU tree as it has superset $\{e, d, b\}$ with same $expSup^{Cap}$, $|\{e, d\}(1.106) - \{e, d, b\}(0.877)| = 0.229 <= 0.4$. So, the final CFU found are $\{e, c, a\}$, $\{d, b, a\}$, $\{e, c\}$, $\{e, d, c\}$, $\{e, d, b\}$ and $\{d, b\}$. The final CFU tree is shown in Figure 4.

Figure 4 CFU tree



4.3 Proposed MFUS tree and MFUS growth algorithm

MFUS tree building is similar to MFU tree. The difference is the maximal itemsets for each data stream have to be mined.

Steps for mining MFUS

The steps to mine MFUS are described below:

Step 1: Similar to MFU tree.

Step 2: Similar to MFU tree.

Step 3: Similar to MFU tree.

Step 4: Similar to MFU tree.

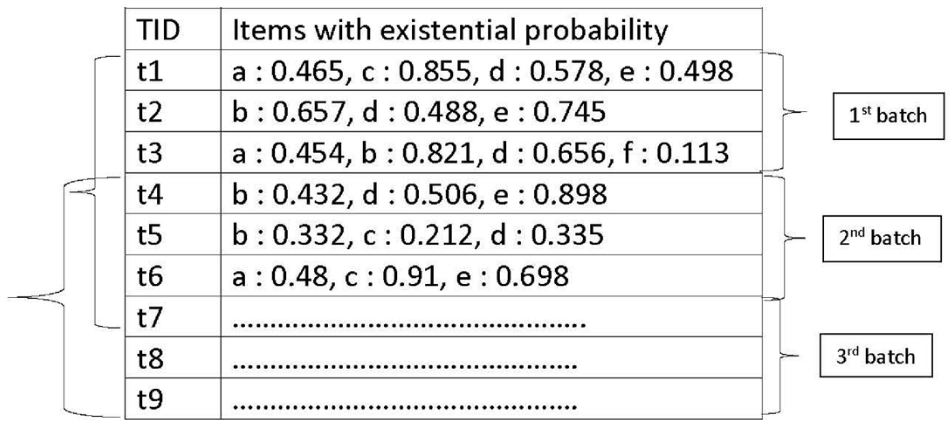
Step 5: Similar to MFU tree.

Step 6: Previous five steps are done for one data stream. As DSTree is used here, three transactions make one batch and two batches together make one data stream. So, one data stream contains six transactions. The whole process will be continued for each data stream.

4.3.1 MFUS tree building and mining maximal frequent itemsets from uncertain data stream

The maximal frequent itemsets from data stream will be mined in the similar way as for the database. Data Stream is shown in Table 2. MFU found for the transactions from t_1 to t_6 are $\{e, c, a\}$, $\{d, b, a\}$, $\{e, d, c\}$ and $\{e, d, b\}$. Then MFU for the transactions from t_4 to t_6 will be derived. In this way, MFU for all the data streams of a dataset will be derived. By mining MFUS from data streams, the important contents of the streams can be captured (e.g., recent data can be captured, because users are usually more interested in recent data than older ones).

Table 2 Data stream


1st batch
2nd batch
3rd batch

TID	Items with existential probability
t1	a : 0.465, c : 0.855, d : 0.578, e : 0.498
t2	b : 0.657, d : 0.488, e : 0.745
t3	a : 0.454, b : 0.821, d : 0.656, f : 0.113
t4	b : 0.432, d : 0.506, e : 0.898
t5	b : 0.332, c : 0.212, d : 0.335
t6	a : 0.48, c : 0.91, e : 0.698
t7	....
t8	....
t9	....

Lemma 1: *Let t_u, t_m and t_l be the transactions containing suffix u,m and l respectively, where $expSup^{Cap}(u) < expSup^{Cap}(m) < expSup^{Cap}(l)$, then t_m can exist as a subset in t_u but not in t_l. In other words, t_m can have supersets in t_u but not in t_l.*

Proof: Assuming $expSup^{Cap}(u) < expSup^{Cap}(m) < expSup^{Cap}(l)$, so $I^{Cap}(t_u) < I^{Cap}(t_m) < I^{Cap}(t_l)$. Therefore, t_m can be a subset of t_u but not of t_l. A subset of an itemsets has greater cap value, if $expSup^{Cap}$ value of the suffix of subset $>$ $expSup^{Cap}$ value of the suffix of superset. □

Example 4: Let be $t_u = \{a : 0.8, b : 0.7, l : 0.6, m : 0.4, u : 0.2\}$; $t_m = \{a : 0.8, b : 0.7, l : 0.6, m : 0.4\}$ and $t_l = \{a : 0.8, b : 0.7, l : 0.6\}$. Here, $expSup^{Cap}(u) < expSup^{Cap}(m) < expSup^{Cap}(l)$. $I^{Cap}(t_u)$ = 0.2 * 0.8 = 0.16, $I^{Cap}(t_m)$ = 0.4 * 0.8 = 0.32 and $I^{Cap}(t_l)$ = 0.6 * 0.8 = 0.48. So, $I^{Cap}(t_u) < I^{Cap}(t_m) < I^{Cap}(t_l)$. This shows that, a subset of an itemset has greater cap value if $expSup^{Cap}$ value of the suffix of subset $>$ $expSup^{Cap}$ value of the suffix of superset.

Lemma 2: *If a maximal frequent itemset Y is generated from MFU/MFUS tree, there is no existence of itemset X among the generated maximal itemsets, such that $Y \subset X$.*

Proof: Let t_u, t_m and t_l be the transactions containing suffix u, m and l, respectively, where $expSup^{Cap}(u) < expSup^{Cap}(m) < expSup^{Cap}(l)$. Itemset Y is generated from t_m. According to lemma 1, X cannot exist in any t_l. X can be generated from any t_m. If length$(Y) = n$, which is the highest length among all t_m, then lengths of all $t_m \leq n$, where $t_m \neq Y$. So, there is no existence of X in any t_m. If length $(Y) < n$, then Y is not the longest t_m, it can or can not have a superset X among all t_m. If X exists in this case, Y will be eliminated by subset checking process and X will be identified as Y. So, there is no X. X can be generated from any t_u. In this case, Y is eliminated by subset_testing process and X is identified as Y. So, there is no existence of X. □

4.4 Proposed CFUS tree and CFUS growth algorithm

CFUS tree building is similar to CFU tree. The difference is, the closed itemsets for each data stream have to be mined.

Steps for Mining CFUS

The steps to mine CFUS are described below:

$Step$ 1: Similar to CFU tree.

$Step$ 2: Similar to CFU tree.

$Step$ 3: Similar to CFU tree.

$Step$ 4: Similar to CFU tree.

$Step$ 5: Similar to CFU tree.

$Step$ 6: Previous five steps are done for one data stream. As DSTree is used here, three transactions make one batch and two batches together make one data stream. So, one data stream contains six transactions. The whole process will be continued for each data stream.

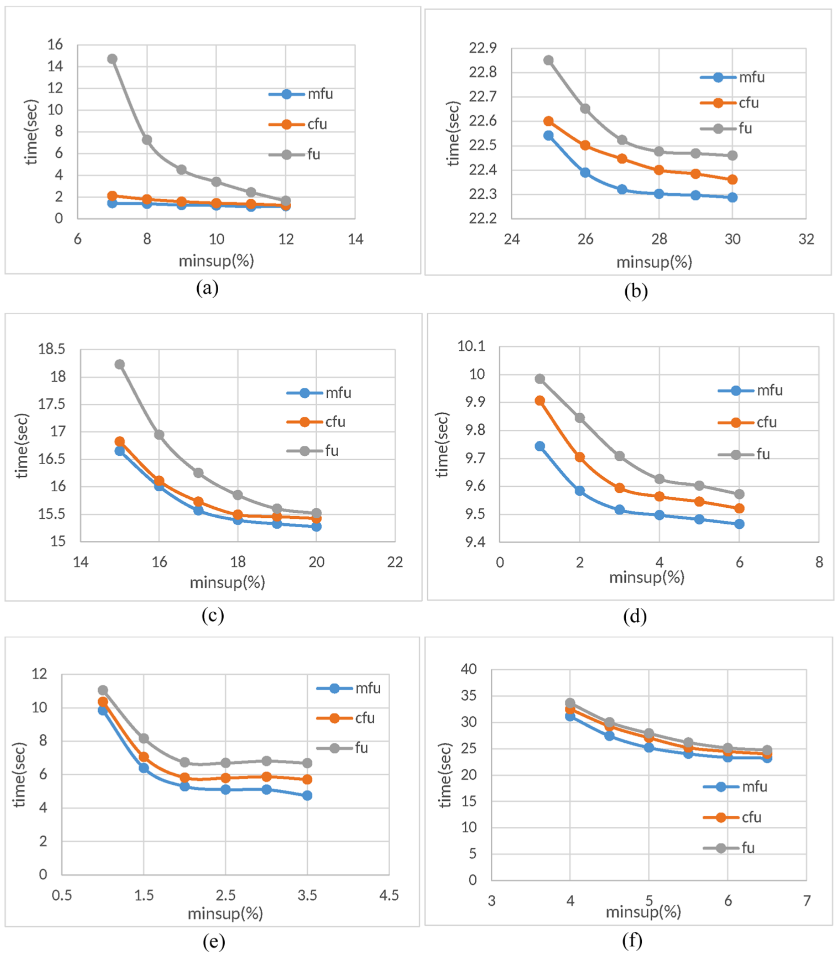
4.4.1 CFUS tree building and mining closed frequent itemsets from uncertain data stream

The closed frequent itemsets from data stream will be mined in the similar way as for the database. Data Stream is shown in Table 2. CFU found for the transactions from t_1 to t_6 are $\{e, c, a\}$, $\{d, b, a\}$, $\{e, c\}$, $\{e, d, c\}$, $\{e, d, b\}$ and $\{d, b\}$. Then CFU for the transactions from t_4 to t_9 will be derived. In this way, CFU for all the data streams of a dataset will be derived. By mining CFUS from data streams, the important contents of the streams can be captured.

Lemma 3: *If a closed frequent itemset Y is generated from CFU/CFUS tree, there is not existence of itemset X among the generated itemsets, such that $Y \subset X$, where X and Y both have similar $expSup^{Cap}$.*

Proof: Let t_u, t_m and t_l be the transactions containing suffix u, m and l respectively, where $expSup^{Cap}(u) < expSup^{Cap}(m) < expSup^{Cap}(l)$. Itemsets Y is generated from t_m. According to Lemma 1, X cannot exist in any t_l. X can be generated from any t_m. If length $(Y) = n$ which is the highest length among all t_m, then the lengths of all $t_m \leq n$, where $t_m \neq Y$. So, there is no existence of X in any t_m. If length $(Y) < n$, Y is not the longest t_m, then it can or can not have a superset X among all t_m. If X exists in this case, Y will be eliminated by closed range checking process and X will be identified as Y. So, there is no X. X can be generated from any t_u. In this case, Y is eliminated by subset_testing as well as closed range checking process and X is identified as Y. So, there is no existence of X. $\square$

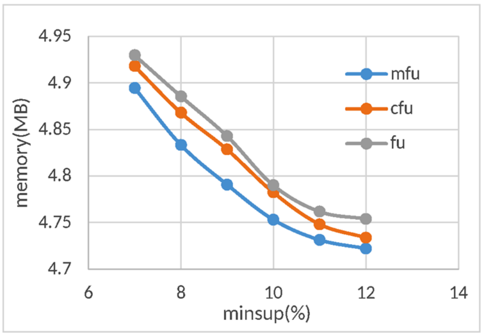
Figure 5 minsup vs. time graphs for FU, CFU and MFU: (a) mushroom, $t = 5418$, $cr = 0.8$, (b) pumsb, $t = 32699$, $cr = 0.6$, (c) pumsb_star, $t = 32699$, $cr = 0.7$, (d) kosarak, $t = 99075$, $cr = 0.5$, (e) T_{10}, $t = 66667$, $cr = 0.9$ and (f) T_{40}, $t = 66667$, $cr = 0.6$ (see online version for colours)



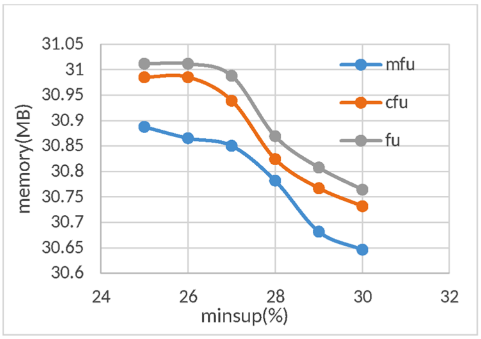
5 Experimental evaluation

The performance evaluation of the proposed algorithm is presented. At first, the algorithms are tested using six different datasets. Two different parameters are taken and the curves for each parameter of each dataset are derived. All the curves of same parameter represent a similar nature. Time and memory decreases with the increment of minimum support threshold. Because when the minimum support threshold is high, more items are eliminated at the beginning and less items remain for further processing. Number of frequent, maximal frequent and closed frequent itemsets generated from each algorithm are shown.

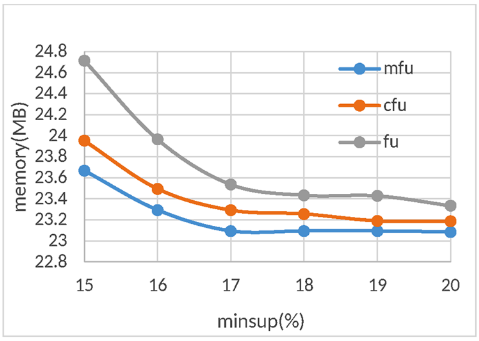
Figure 6 minsup vs. memory graphs for FU, CFU and MFU: (a) mushroom, $t = 5418$, $cr = 0.8$, (b) pumsb, $t = 32699$, $cr = 0.6$, (c) pumsb_star, $t = 32699$, $cr = 0.7$, (d) kosarak, $t = 99075$, $cr = 0.5$, (e) T_{10}, $t = 66667$, $cr = 0.9$ and (f) T_{40}, $t = 66667$, $cr = 0.6$ (see online version for colours)



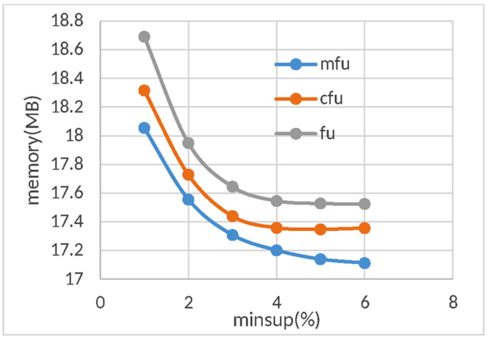
(a)



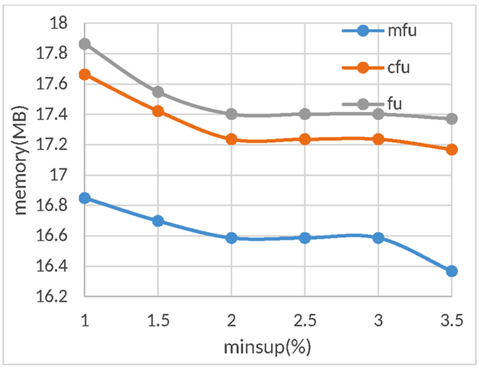
(b)



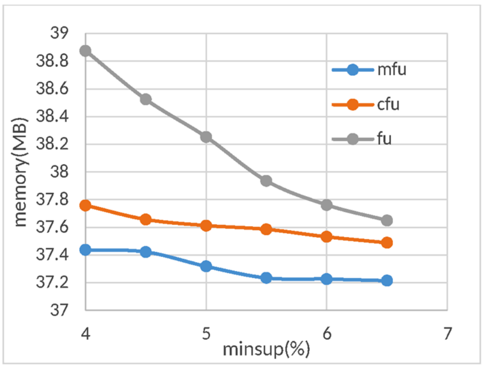
(c)



(d)



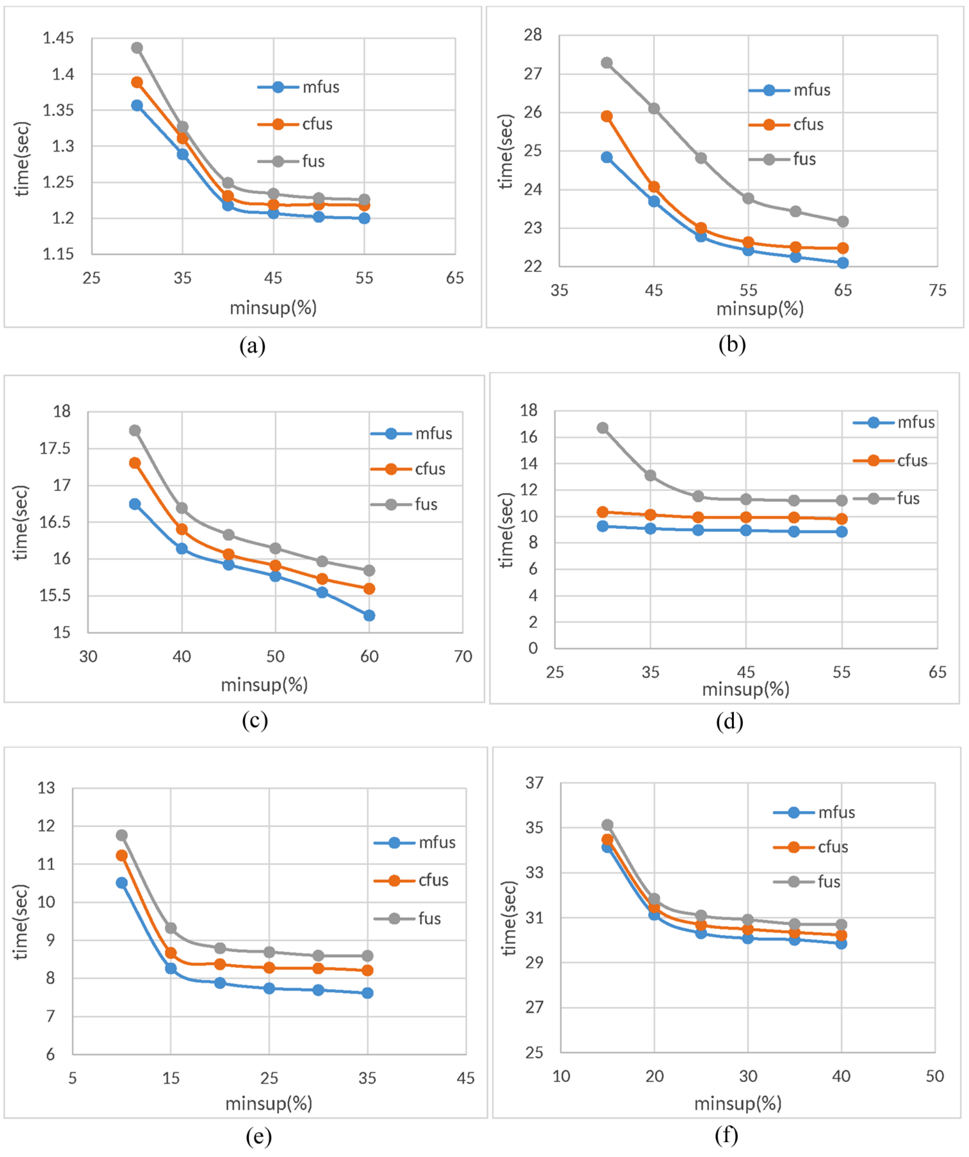
(e)



(f)

To test the algorithms, large datasets from well known data repository are required to be used. Unfortunately, large data repository for uncertain data is not available online. For this reason, uncertain datasets from certain datasets have been generated. Probability values are generated using normal or Gaussian distribution from each item of a transaction. The reason behind using normal distribution is to produce real life dataset.

Figure 7 minsup vs. time graphs for FUS, CFUS and MFUS: (a) mushroom, t = 5418, cr = 0.8, (b) pumsb, t = 30000, cr = 0.6, (c) pumsb_star, t = 30000, cr = 0.7, (d) kosarak, t = 95000, cr = 0.5, (e) T_{10}, t = 66667, cr = 0.9 and (f) T_{40}, t = 66000, cr = 0.6 (see online version for colours)



Experiments are done on mushroom, pumsb, pumsb_star, kosarak, T_{10} and T_{40}. Among them mushroom, pumsb and pumsb_star are dense datasets. Kosarak, T_{10} and T_{40} are sparse datasets. Experimental results contain two parts:

i Graphical representation of each algorithm. Here, evaluation is shown for two comparison measurement scales:

1 Minsup threshold vs. time

2 Minsup threshold vs. memory.

ii Number of frequent, maximal frequent and closed frequent itemsets generated from each algorithm. While applying normal distribution, mean = 0, standard deviation = 1 have been assumed.

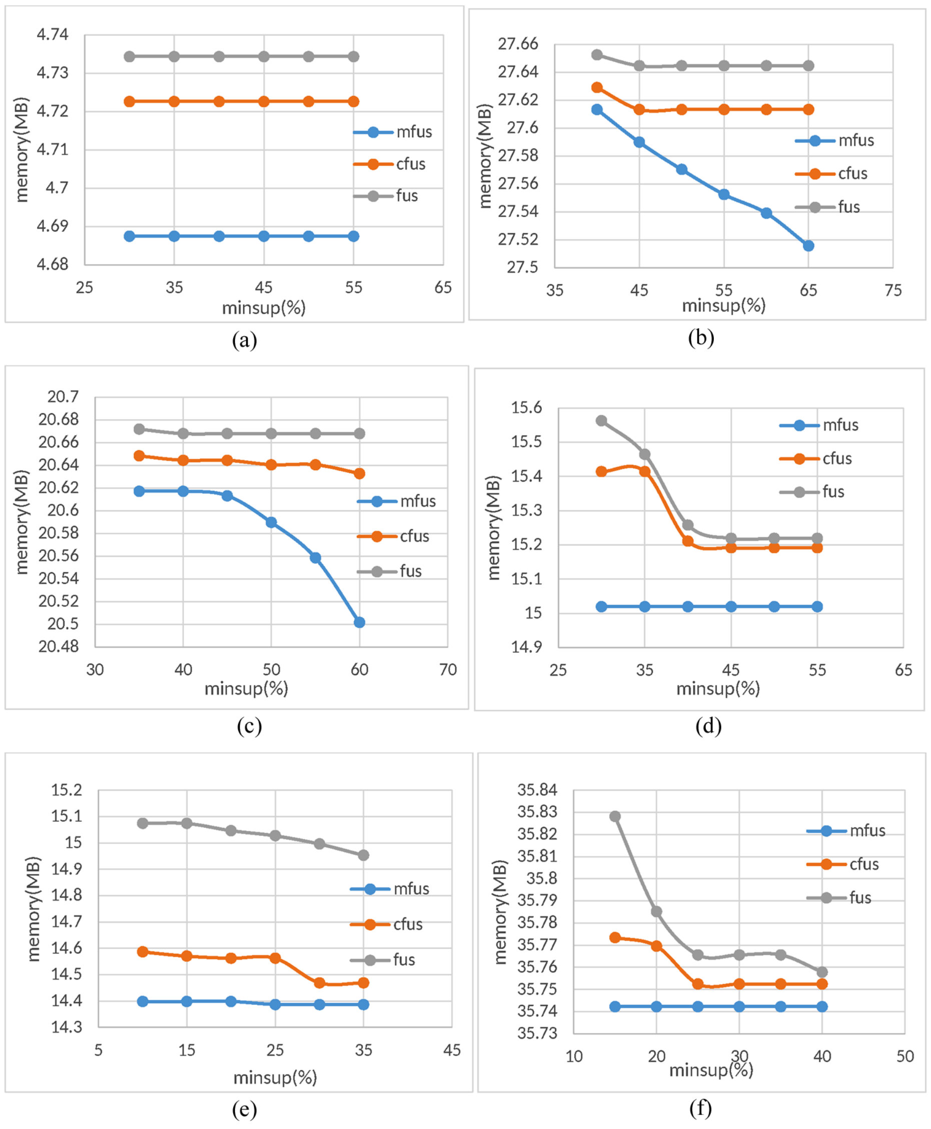


Figure 8 minsup vs. memory graphs for FUS, CFUS and MFUS: (a) mushroom, t = 5418, cr = 0.8, (b) pumsb, t = 30000, cr = 0.6, (c) pumsb_star, t = 30000, cr = 0.7, (d) kosarak, t = 95000, cr = 0.5, (e) T_{10}, t = 66667, cr = 0.9 and (f) T_{40}, t = 66667, cr = 0.6 (see online version for colours)

In the minsup vs. time graphs, x-axis represents the minimum support threshold and y-axis represents the time needed for particular threshold. Here it is noticeable that the amount of time decreases with the increment of minimum support threshold. Because when minimum support threshold increases, number of itemsets satisfying the minimum threshold decreases. So, the calculation becomes smaller. Figure 5 shows minsup vs. time graphs for FU, CFU and MFU. Figure 7 shows minsup vs. time graphs for FUS, CFUS and MFUS.

In the minsup vs. memory graphs, x-axis represents the minimum support threshold and y-axis represents the memory needed for particular threshold. Here it is noticeable that the amount of memory decreases with the increment of minimum support threshold. Because when minimum support threshold increases, number of itemsets satisfying the minimum threshold decreases. So, the calculation becomes smaller. Figure 6 shows minsup vs. memory graphs for FU, CFU and MFU. Figure 8 shows minsup vs. memory graphs for FUS, CFUS and MFUS.

Table 3 shows the number of itemsets generated from FU, CFU and MFU. Here it can be seen for each of the datasets, number of FU $>$ number of CFU $>$ number of MFU. Table 4 shows the number of itemsets generated from FUS, CFUS and MFUS. Here for each of the datasets, number of FUS $>$ number of CFUS $>$ number of MFUS.

Table 3 Number of itemsets generated from FU, CFU and MFU: (a) pumsb, t = 32699, cr = 0.6, (b) mushroom, t = 5418, cr = 0.8, (c) kosarak, t = 99075, cr = 0.5, (d) pumsb_star, t = 32699, cr = 0.7, (e) T_{40}, t = 66667, cr = 0.6 and (f) T_{10}, t = 66667, cr = 0.9

minsup(%)	FU	CFU	MFU
25	16	16	0
26	9	9	0
27	1	1	0
28	1	1	0
29	1	1	0
30	1	1	0

(a)

minsup(%)	FU	CFU	MFU
7	143	35	1
8	100	30	1
9	72	25	1
10	54	24	1
11	39	22	0
12	32	19	0

(b)

minsup(%)	FU	CFU	MFU
1	28	24	2
2	10	10	2
3	6	6	1
4	6	6	0
5	4	4	0
6	3	3	0

(c)

minsup(%)	FU	CFU	MFU
15	24	24	0
16	16	16	0
17	11	11	0
18	9	9	0
19	2	2	0
20	1	1	0

(d)

minsup(%)	FU	CFU	MFU
4	14	14	0
4.5	9	9	0
5	6	6	0
5.5	5	5	0
6	3	3	0
6.5	3	3	0

(e)

minsup(%)	FU	CFU	MFU
1	30	30	0
1.5	13	13	0
2	6	6	0
2.5	3	3	0
3	1	1	0
3.5	0	0	0

(f)

Table 4 Number of itemsets generated from FUS, CFUS and MFUS: (a) pumsb, t = 30000, cr = 0.6, (b) mushroom, t = 5418, cr = 0.8, (c) pumsb_star, t = 30000, cr = 0.7, (d) kosarak, t = 95000, cr = 0.5, (e) T_{40}, t = 66000, cr = 0.6 and (f) T_{10}, t = 66667, cr = 0.9

minsup(%)	FUS	CFUS	MFUS
40	5000	5000	0
45	5000	5000	0
50	4923	4923	0
55	3028	3028	0
60	361	361	0
65	2	2	0

(a)

minsup(%)	FUS	CFUS	MFUS
30	1747	1543	114
35	915	912	3
40	902	902	0
45	881	881	0
50	681	681	0
55	249	249	0

(b)

minsup(%)	FUS	CFUS	MFUS
35	5035	5030	5
40	4916	4916	0
45	4251	4251	0
50	2506	2506	0
55	843	843	0
60	87	87	0

(c)

minsup(%)	FUS	CFUS	MFUS
30	8133	7668	1
35	4659	4313	1
40	2190	1929	1
45	907	707	1
50	346	189	1
55	175	48	1

(d)

minsup(%)	FUS	CFUS	MFUS
15	33323	32700	483
20	15875	15869	6
25	9135	9135	0
30	3624	3624	0
35	1025	1025	0
40	223	223	0

(e)

minsup(%)	FUS	CFUS	MFUS
10	49588	47012	539
15	12251	11323	2
20	3979	3279	1
25	1018	465	1
30	534	84	1
35	398	25	1

(f)

6 Conclusion and future works

The mining process in an uncertain data stream is more challenging and complicated than in precise database. The MFU and CFU algorithms are new algorithms to mine maximal and closed frequent itemsets from uncertain data stream. The algorithm used a very efficient data structure which provide great output regarding the runtime and memory consumption. The main goal of the paper is to target fast and memory efficient computation of the maximal and closed frequent itemsets mining in an appropriate approach. Thus a new area of data mining studies has been explored with newer possibilities. However, some points are to be focused for future improvements over the algorithms.

First, the expSup from original database has to be calculated each time for the generated MFU and CFU whose $expSup^{Cap}$ satisfies minimum support threshold. If calculating expSup for each and every item can be ignored, the algorithm will be more time efficient.

Second, $expSup^{Cap}$ difference is used as closed range. Original expSup will be used to find out whether two items are in closed range or not. That means whether two items have same or different support.

Thirdly, the number of database scans will be reduced and make it more time efficient.

Fourthly, top-k maximal and closed frequent itemsets will be tried to mine. Mining top-k itemsets is necessary because if minimum support threshold is set too low, too many itemsets (frequent, maximal and closed frequent) will be generated, which may cause the mining process to be very inefficient. On the other hand, if minimum support threshold is set too high, it is likely that no itemsets will be found.

References

Agrawal, R. and Srikant, R. (1994) 'Fast algorithms for mining association rules in large databases', *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, 12–15 September, Santiago de Chile, Chile, pp.487–499.

Agarwal, R.C., Aggarwal, C.C. and Prasad, V.V.V. (2000) 'Depth first generation of long patterns', *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 20–23 August, Boston, MA, USA, pp.108–118.

Aggarwal, C.C., Li, Y., Wang, J. and Wang, J. (2009) 'Frequent pattern mining with uncertain data', *Proceedings of 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, 28 June – 01 July, Paris, France, pp.29–38.

Bayardo, R.J. (1998) 'Efficiently mining long patterns from databases', *Proceedings ACM SIGMOD International Conference on Management of Data*, 2–4 June, Seattle, Washington, USA, pp.85–93.

Burdick, D., Calimlim, M. and Gehrke, J. (2001) 'MAFIA: a maximal frequent itemset algorithm for transactional databases', *Proceedings of the 17th International Conference on Data Engineering*, 2–6 April, Heidelberg, Germany, pp.443–452.

Chi, Y., Wang, H., Yu, P.S. and Muntz, R.R. (2006) 'Catch the moment: maintaining closed frequent itemsets over a data stream sliding window', *Knowledge and Information Systems*, Vol. 10, No. 3, October, pp.265–294.

Chui, C.K., Kao, B. and Hung, E. (2007) 'Mining frequent itemsets from uncertain data', *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD*, 22–25 May, Nanjing, China, pp.47–58.

Gouda, K. and Zaki, M.J. (2001) 'Efficiently mining maximal frequent itemsets', *Proceedings of the 1st IEEE International Conference on Data Mining*, 29 November – 2 December, San Jose, California, USA, pp.163–170.

Grahne, G. and Zhu, J. (2003a) 'High performance mining of maximal frequent itemsets', *SIAM'03 Workshop on High Performance Data Mining: Pervasive and Data Stream Mining*, May.

Grahne, G. and Zhu, J. (2003b) 'Efficiently using prefix-trees in mining frequent itemsets', *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, 19 December, Melbourne, Florida, USA.

Grahne, G. and Zhu, J. (2005) 'Fast algorithms for frequent itemset mining using FP-trees', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 10, October, pp.1347–1362.

Han, J., Pei, J., Yin, Y. and Mao, R. (2004) 'Mining frequent patterns without candidate generation: a frequent-pattern tree approach', *Data Mining and Knowledge Discovery*, Vol. 8, No. 1, January, pp.53-87.

Hashem, T., Karim, M.R., Samiullah, M. and Ahmed, C.F. (2017) 'An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure', *Expert Systems with Applications*, Vol. 67, January, pp.252–271.

Karim, M.R., Cochez, M., Beyan, O.D., Ahmed, C.F. and Decker, S. (2018) 'Mining maximal frequent patterns in transactional databases and dynamic data streams: a spark-based approach', *Information Sciences*, Vol. 432, March, pp.278–300.

Leung, C.K-S. and Khan, Q.I. (2006) 'DSTree: a tree structure for the mining of frequent sets from data streams', *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, 18–22 December, Hong Kong, China, pp.928–932.

Leung, C.K-S., Mateo, M.A.F. and Brajczuk, D.A. (2008) 'A tree-based approach for frequent pattern mining from uncertain data', *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD*, 20–23 May, Osaka, Japan, pp.653–661.

Leung, C.K. and Tanbeer, S.K. (2013) 'PUF-tree: a compact tree structure for frequent pattern mining of uncertain data', *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD, Part I*, 14–17 April, Gold Coast, Australia, pp.13–25.

Pei, J., Han, J. and Mao, R. (2000) 'CLOSET: an efficient algorithm for mining frequent closed itemsets', *Proceedings of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 14 May, 2000, Dallas, Texas, USA, pp.21–30.

Vo, B., Pham, S., Le, T. and Deng, Z-H. (2017) 'A novel approach for mining maximal frequent patterns', *Expert Systems with Applications*, Vol. 73, No. C, May, pp.178–186.

Wang, J., Han, J. and Pei, J. (2003) 'CLOSET+: searching for the best strategies for mining frequent closed itemsets', *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 24–27 August, Washington, DC, USA, pp.236–245.

Zaki, M.J. and Hsiao, C. (2002) 'CHARM: an efficient algorithm for closed itemset mining', *Proceedings of the 2nd SIAM International Conference on Data Mining*, 11–13 April, Arlington, VA, USA, pp.457–473.