An Efficient Compression Algorithm for Uncertain Databases Aimed at Mining Problems

Mahmoud M. Gabr, Saad M. Darwish, and Sayed A. Mohsin

***Abstract*—Many studies on association rule mining have focused on item sets from precise data in which the presence and absence of items in transactions was certainly known. In some applications, the presence and absence of items in transactions are uncertain and the knowledge discovered from this type of data will extracted with approximation manner. Data compression offers a good solution to reduce data size that can save the time of discovering useful knowledge. In this paper we suggest a new algorithm to compress transactions from uncertain database based on modified version of *M²TQT* (Mining Merged Transactions with the Quantification Table) approach and fuzzy logic concept. The algorithm bands the uncertain data to set of clusters using *K-Mean* algorithm and exploits fuzzy membership function to classify the transaction items as one of those clusters. Finally, the modified version of *M²TQT* has been employed to compress the classified transactions. The key idea of our algorithm is that since uncertain data is probabilistic in nature and frequent item set is counted as expected values so, compressed transactions will give us approximate values for the item set's support. Experimental results show that the proposed algorithm is better than *U-Apriori* algorithm in case of large uncertain database.**

***Index Terms*—Rule mining, database compression, Uncertain database, fuzzy logic.**

I. Introduction

Mining information and knowledge from large databases has been recognized by many researchers as a key study topic in database systems and machine learning and by many industrial companies as an important area with a prospect of major revenues. Several emerging applications in information-providing services, such as data warehousing and online services over the Internet, call for various data mining techniques to better understand user behavior, to improve the service provided and to increase business opportunities.

Data mining has been attracted huge attention in many research groups due to its extensive applicability in many areas such as retail industry, financial forecast, decision support, and intrusion detection [1]. Data mining methods include associations clustering, classification, and prediction. One of the most significant fields of the data mining domain is the association mining. Association rule mining, the task of finding correlations between items in a dataset, has received considerable attention, particularly since the publication of the *Apriori* algorithms [2]. Early research was mainly motivated by the analysis of market basket data, the results of which allowed companies to more fully understand purchasing behavior and, as a result, better target market audiences.

There have been numerous studies on mining frequent patterns from precise data such as databases of market basket transactions, web logs, and click streams. In these databases, users definitely know whether an item (or an event) is present in, or is absent from, a transaction in the databases. However, there are situations in which users are uncertain about the presence or absence of some items or events [3]-[5]. For example, a physician may highly suspect (but cannot guarantee) that a patient suffers from flu. The uncertainty of such suspicion can be expressed in terms of existential probability. This is because data collection methodologies are often inaccurate and are based on incomplete or inaccurate information. To this end, several pattern mining methods for uncertain data have recently been proposed [5]-[7]. Most of them assume statistical independence of the items in all transactions and adopt the possible world's interpretation of probabilistic data [8].

Although, there have been many scalable methods developed for frequent-pattern mining, the real bottleneck of the problem is not at the efficiency but at the usability. Typically, if *min-sup* threshold is high, mining may generate only commonsense patterns, however, with a low *min-sup*, it may generate an explosive number of results. This has severely restricted the usage of frequent-pattern mining. To solve this problem, it is natural to explore how to "compress" the patterns, i.e., find a concise and succinct representation that describes the whole collection of patterns.

Recently, many systems are proposed for knowledge discovery from compressed databases [9]-[11] that start by transform the original database into a new data representation where several transactions are merged to become a new transaction then it uses an Apriori-like algorithm for association rule mining to find useful information. However, there are some problems in the approach suggested by M. Ashrafi *et al*. [9]; first, the compressed database is not reversible after the original database is transformed by the data preprocessing step. Second, although some rules can be mined from the new transactions, it still needs to scan the database again to verify the result. The authors in [10] proposed *M²TQT* approach and presented it as the algorithm that will recover the problem of decompressing the database to its original state; but the problem still exists due to the lack of rules and constraints while merging transactions in the data compression phase.

Manuscript received May 25, 2014; revised July 28, 2014.

Mahmoud M. Gabr is with the Department of Mathematics and Computer Sciences, Faculty of Science, Alexandria University, Egypt (e-mail: mahgabr@yahoo.com).

Saad M. Darwish is with Computer Science, Institute of Graduate Studies and Research (IGSR), Alexandria University, Egypt (e-mail: saad.darwish@alex-igsr.edu.eg).

Sayed A. Mohsin is with the IGSR, Alexandria University, Egypt, and is also with Amiral, Egypt (e-mail: sayed.abdelmohsin@amiral.com).

Applying the compression techniques on uncertain data suffer from many challenges. The most important one is item's probability that must be considered when merging the transactions. Furthermore, the item's probability with small values will affect the compression processing performance. With the purpose to alleviate these problems, we made some modifications to M^2TQT approach and devised a new algorithm in which the original transactions with uncertain data are clustered into specific groups using *K*-Mean algorithm and fuzzy logic membership function is used to represent this transactions as discrete values. Then, an adapted version of M^2TQT approach is employed to compress the clustered transactions into new database with merged transactions.

The rest of this paper is organized as follows: Section II explains preliminaries that formally define the uncertain data model. Section III contains literature survey of database compression methods. Section IV introduces the proposed technique for compress uncertain database. Section V reports the accuracy and performance evaluation of the proposed technique and gives experimental results. Finally, Section VI summarizes the conclusion and future work.

II. Preliminaries

Most of uncertain data models use possible world semantics to represent the uncertain relations among item [12]. Since the uncertain transactions are probabilistic in nature, it is impossible to count the frequency of itemsets deterministically. Therefore, we count the frequent itemsets only as expected value.

Definition 1: An uncertain item is an item $x \in W$ whose presence in a transaction $t \in T$ is defined by an existential probability $P(x \in t) \in [0, 1]$, where *W* is the possible world for the item x in the set of transactions *T* [13]. A certain item is an item where presence of an item x is either 0 or 1.

Definition 2: An uncertain transaction *t* is a transaction that contains uncertain items. Let *X* is set of items or an item set consists of $X=\{x_1, x_2, \ldots, x_n\}$. There are two possible worlds for an item *x* and a transaction t_i: 1) W_1 where $x \in t_i$ and 2) W_2 where $x \notin t_i$. The probability for W_1 to be the true world is thus $P(x, t)$, and for W_2 is $1–P(x,t)$, where $P(x, t)$ represents the existential probability of x in t [12].

The probability of two items x_1 and x_2 to exist at the same time in a transaction *t* is thus $P(x_1, t)*P(x_2, t)$. In an uncertain database, the expected count of an item or an item set is used. The expected support, $ExpSup$, of set of items *X* in all the transactions $N = |D|$ of uncertain database is then calculated as follows [13]:

$$ExpSup = \sum_{i=1}^{N} (\prod_{x \in X} (P(x, t_i))) \quad (1)$$

Definition 3: in each transaction in database *D* consisting of *N* uncertain transactions, the expected support of a pattern (or a set of items) *X* in *D* can be computed by summing the support of *X* in possible world W_j (while taking into account the probability of W_j to be the true world) over all possible worlds [13]:

$$ExpSup = \sum_{j} [\sup(X) \, in \, W_j \times \prod_{i=1}^{N} (\prod_{x \in t_i \, in \, W_j} p(x, t_i) \times \prod_{y \in t_i \, in \, W_j} (1 - p(y, t_i))] \quad (2)$$

Definition 4: An item set *X* is said to be frequent when the expected support *ExpSup* of the item set is larger than the user-defined threshold *MinSup*.

A. Problem Definition

1) Objective

Consider a database *D*, consisting of *N* uncertain transactions and *m* items. The items had values defined by an existential probability $P(x \in t) \in [0,1]$. The aim of compression is to construct a new modified database *D`* from *D* that have *N`* transactions such that *N`*< *N*, which will reduce the database size and the number of association rules generated from the original database *D* without losing the knowledge.

2) Inputs

1) A database *D* of *N* uncertain transactions and *m* items with values defined by an existential probability $P(x, t)$.
2) An arbitrary integer *k* that represents the number of clusters in which the original item values are classified to it.
3) An expected item support that will be used by *U-Apriori* (Apriori algorithm for uncertain data) to generate the frequent item set.

III. Literature Survey

Conventional data compression's techniques require that compressed data be decompressed before read or write operations can be carried out. As a result, it is not practical to compress databases in active routine using the conventional data compression techniques. Research in uncertain and probabilistic database management is abundant (see [12], [13] for recent surveys). Recently, also the data mining community has started dealing with uncertain data, tackling the problems of clustering, classification and frequent pattern mining, but finding small and informative pattern-based descriptions of uncertain databases has not been addressed before.

To handle uncertain data, *C. Chui et al.* [3] proposed the *U-Apriori* algorithm, which is a modification of the *Apriori* algorithm. Specifically, instead of incrementing the support counts of candidate patterns by their actual support, *U-Apriori* increments the support counts of candidate patterns by their expected support. *U-Apriori* suffers from the following problems: 1) Inherited from the *Apriori* algorithm, *U-Apriori* does not scale well when handling large amounts of data because it also follows a level wise generate-and-test framework. 2) If the existential probabilities of most items within a pattern *X* are small, increments for each transaction can be slightly small. Consequently, many candidates would not be recognized as infrequent until most (if not all) transactions were processed.

The authors in [14] proposed a representation scheme to

characterize uncertain data based on possibility distributions. The possibility theory establishes a close connection between the concepts of similarity and uncertainty, providing an excellent framework for handling uncertain data. Also, they developed an algorithm to mine fuzzy association rules from uncertain data represented by possibility distributions. The difficulty is that a possibility distribution may match a pattern with multiple support values; where a support value is obtained for every plausible value in the distribution. Their method computes two measures from these values, support and deviation. A pattern is considered as a good pattern if its support is high but its deviation is low, because low deviation means the pattern is more certain and believable.

Since most data occupy a large amount of storage space, it is beneficial to reduce the data size that makes the data mining process more efficient with the same results. Compressing the transactions of databases is one way to solve the problem. For example, the work in [9] proposed a new approach to compress database to reduce the size of transactions' database. The algorithm was divided into data preprocessing and data mining. The data preprocessing stage transforms the original database into a new data representation. It uses lexical symbols to represent raw data. The merge-mining algorithm is then used to find frequent itemsets from the new merged transactions. Although the algorithm successes in reducing the database size but the compressed database cannot decompressed to its original data. Furthermore, a lot of candidate itemsets could be generated in large transactions with higher process cost since it needs to scan the database more than once.

J. Dai *et al*. [10] proposed a new approach called Mining Merged Transactions with the *Quantification Table* (M^2TQT) to solve the problems inherited form the application of the above-mentioned algorithm. M^2TQT uses the relationship between transactions to merge related transactions and builds a quantification table to prune the candidate itemsets that are impossible to become frequent in order to improve the performance of mining association rules. The algorithm was divided into three phases: merge related transactions to generate a compressed database, build a quantification table, and discover frequent itemsets. The advantage of M^2TQT over the work proposed in [2] is that no need for multiple database scans, because M^2TQT save information about the original database in the merged transactions and quantification table. Although, the algorithm tries to recover the problem of decompressing the database to its original state, the problem still exists due to the lack of rules for merging in the data compression phase.

To the best of our knowledge, this is the first work addressing the problem of uncertain database compression. The main contribution is introducing, characterizing and solving a novel database compression algorithm for uncertain data that is utilized for data mining applications. The proposed algorithm applies the compression technique as used in [10] but on uncertain data with considering the item probabilities while merging the transactions. Moreover, using the mean value of the clusters instead of item probability in the merged transactions will avoid outlier probabilities with small values that affect the compression performance. The proposed technique uses a modified version from *U-Apriori* algorithm, a method to mine frequent sequences in the presence of uncertainty in transactions, to extract frequent patterns from the compressed uncertain database.

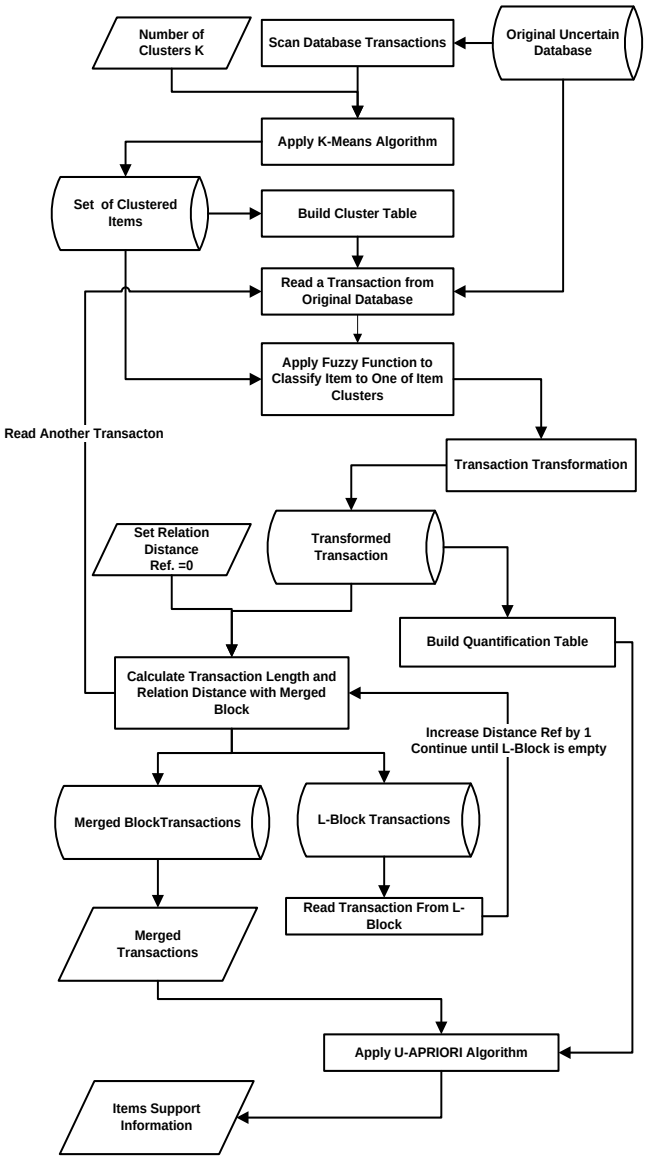

Fig. 1. The proposed compression algorithm.

IV. Proposed Technique

The proposed technique focuses on compressing related uncertain transactions by collecting the uncertain data into set of clusters and building a quantification table for those clusters to help in pruning candidate itemsets, which are impossible to become frequent itemsets. To achieve this goal the proposed algorithm relies on a modified version from M^2TQT approach for compressing uncertain transactions to reduce the size of a database's transactions. Then, it uses modified *U-Apriori* algorithm to mine the compressed database. The proposed technique has five phases:

1) Scan the database's transactions and use the *k*-mean algorithm to cluster the transactions of probabilistic data into set of clusters and calculate the mean value for each cluster. The information about items' clusters is kept in Cluster Table.
2) Transform the original transaction's items to a new

representation using fuzzy logic membership function. The membership function realizes the nearest cluster to the transaction probabilistic value and replaces the item by the cluster's mean in the transaction.

3) Merge related transformed transactions to generate a compressed database based on relation distance between transformed transactions.
4) Build a quantification table to reduce the number of generated candidate itemsets to help trimming non-frequent itemsets.
5) The compressed database is employed to generate frequent itemsets using modified U-Apriori algorithm for association rule mining.

As shown in Fig. 1, the proposed algorithm utilizes the *k*-Mean algorithm to cluster the probabilistic data in the database and then exploits the algorithm in [10] to achieve the compression, but we add three substantial modifications on this system to enhance the compression ratio:

1) Transaction's relation distance will considers the item support while calculating the distance between the merged block and the database's transaction or transaction read from length block.

Example: Consider the following transactions: T_1= {*AB*}, T_2= {*A*} and T_3= {*B*}. The relation distance $D_{T1\text{-}T2}$= 1, so T_1 and T_2 will merge to give {*AB*=2, 1}, where "=" symbol is used to separate items and their respective support counts.

Before modification: The relation distance between {*AB*=2, 1} and T_3= {*B*} is 1, so they are merged to become {*AB*=2, 2}, which gives wrong support for the itemset {*AB*}.

After modification: The relation distance between {*AB*=2, 1} and T_3= {*B*} that considers the support for item {*A*=2} gives a relation distance equal 2 and so {*AB*=2, 1} and T_3= {*B*} are not merged.

2) Prevent merging any two combined blocks created with different transaction relation's distance.

Example: Consider merged block with items {*ABCD* = 2, 2, 2, 1} created from relation distance =1. Now, when relation distance is incremented by 1 and the procedure reads transaction from length blocks with items BC then the block {*ABCD* = 2, 2, 2, 1} is not merged with BC but it settles in different merged block. This allows the merged block {*ABCD* = 2, 2, 2, 1} to decompress to its original transactions *ABC* and *ABCD*.

Initialize the relation distance by zero instead of one that helping to merge identical transactions, so it becomes peaceful in separating merged block.

A. Pseudo Code

Given: 1) The database *D* with probabilistic transaction values, 2) The arbitrary number of clusters *k*, and 3) Expected item support *E* that is used in mining phase while generate frequent itemsets. The main algorithm scans the original database and calls the k-Mean cluster algorithm that creates the items' clusters. The algorithm then loop on every transaction in *D* and 1) execute fuzzy transformation algorithm to transform the transaction items into new representation using fuzzy logic membership function, 2) execute build quantification table algorithm in order to reduce the number of generated candidate itemsets, and 3) execute merge transaction algorithm that adds the transformed transaction to either length block or merged block depending on the merging constraint. Then the algorithm execute merge length block algorithm that moves the transferred transactions from length block to merged block depending on the relation distance and finally calls U-Apriori algorithm to generate the frequent items set conditional with the predefined expected support.

Algorithm 1 : Main Algorithm

Inputs: *D, k* and *E*.
Output: *Frequent itemset along with its support*

1 **Call** *k*-Mean cluster algorithm //*create cluster table*
2 Set *RD* = 0 // *initialize relation distance by zero*
3 **For** ***i* = 1 to *N* Do** // *N number of transactions in D*
3.1 **Call** Fuzzy transformation algorithm
3.2 **Call** Build quantification table algorithm
3.3 **Call** Merge transaction algorithm
3 **End For**
4 **While** (Length Block NOT Empty)
4.1 **Call** Merge length block algorithm
4.2 *RD* ++
4 **End While**
5 **Call** *U-Apriori* algorithm

Algorithm 2: K-Mean Cluster Algorithm

Inputs: *D and k.*
Output: *Cluster table.*
1 **For** ***i* = 1 to *N* DO** // *N number of transactions in D*
1.1 Add item values to Items Array *X*
1 **End For**
2 **For** ***i* = 1 to *M* Do** // *M number of different items X in D*
2.1 Get the values for current item x_i.
2.2 Use *k*-Mean algorithm for cluster x_i
// *calculate the mean value for each cluster*
2.3 Add the clusters to *Cluster Table*
2 **End For**
3 Return *Cluster Table*

The algorithm scans the database *D* to deduce the different items that exist in the database along with its probabilistic values and attaches them to external array of items *X*. This array is used to get the clusters for each item x_i in *X* and adds them with corresponding mean value to the Cluster Table as illustrated in Table I.

TABLE I: EXAMPLE OF CLUSTER TABLE

Item (*X*)	Item Clusters (X_{CL})	Mean Value (μ_{CL})
A	A_1	0.80
	A_2	0.70
B	B_1	0.50
	B_2	0.72
C	C_1	0.82
	C_2	0.75

Algorithm 3: Fuzzy Transformation Algorithm

Inputs: *Transaction T*
Output: *Transformed Transaction* T_f
1 **For** ***i* = 1 to *M* Do** //*M number of different items X in T*
1.1 $x_{cl} = \eta(x_i)$ //η *fuzzy membership function*
// *get suitable cluster for the current item* x_i
//*depend on the item transaction value*
1.2 $x_{cl} \rightarrow x_i$ // *replace* x_{cl} *instead of* x_i *to get* T_f
1 **End For**
2 Return T_f

The algorithm replaces the item in each database's transaction by proper cluster's mean value using the fuzzy

logic membership function defined as:

$$d_{cl} = \sqrt[2]{(x - \mu_{cl})^2} \quad (3)$$

where d_{cl} represents the distance between the item's value and mean value of the item's cluster μ_{cl}, and x symbols the item probabilistic value. The value d_{cl} is calculated for every cluster of the item and the nearest cluster's mean value is used as a replacement of item in the transaction.

To reduce the number of generated candidate itemsets, additional information is required to help pruning non-frequent itemsets. A simple quantification table is used to record this information when each transaction is processed. The proposed system uses the same technique employed in [10] to build the quantification table as illustrated in Table II. Here, A_1 and A_2 are the clusters for the Item *A*. B_1 and B_2 are the clusters for the Item *B*. C_1 and C_2 are the clusters for the Item *C*.

TABLE II: EXAMPLE OF QUANTIFICATION TABLE

L4	L3	L2	L1
$A_1(1)$	$A_1(3)$	$A_1(3)$	$A_1(3)$
	$A_2(1)$	$A_2(1)$	$A_2(1)$
	$B_1(1)$	$B_1(2)$	$B_1(2)$
	B2(1)	B2(1)	B2(1)
	$C_2(1)$	$C_2(1)$	$C_2(2)$
		$C_1(3)$	$C_1(4)$

Algorithm 4: Build Quantification Table Algorithm

Inputs: *Transformed Transaction Tf.*
Output: *Quantification Table.*

```
1     For i = 1 to M_f Do
          //M_f number of different items X in T_f
1.1       If exist( item x_i in Quantification Table) then
1.1.1         Supp ++ //increment item support by 1
1.1       Else
1.1.2         Add item to Quantification Table
1.1.3         Supp = 1
1.1       End If
1     End For
2     Return Quantification Table
```

Given the transformed transaction's items, the algorithm starts to check of relation distance that equal zero. The identical transactions that have the same clusters are merged and are inserted into Merge Block. The remaining non merged transactions insert into Length Block. After this algorithm is completed, the remaining algorithms work separately from original database. The algorithm begins by validation of non-empty length block, and adds the transformed transaction to it in case of empty. In case of not empty length block, the algorithm calculates the relation distance between the transformed transaction and that which exist in length block. In case of relation distance equal zero, the transformed transaction merges with that exist in length block and adds to merge block; else the transformed transaction appends to length block.

After that, the algorithm receives transaction from length block and relation distance value that is used to validate the merging between the transactions from length block and merge block. The algorithm loops for every merged block's transaction and calculates the relation distance between it and received transaction from length block. In case of relation distance satisfies the received relation distance reference, the received length block transaction will merge to merge block, else the relation distance is accumulated by one and the algorithm is executed again. The accumulation of relation distance and the execution of algorithm will continue until the length block becomes empty.

Algorithm 5: Merge Transaction Algorithm

Inputs: *Transformed Transaction T_f.*
RD = 0 // initialize relation distance by zero
Output: *T_{MB} // Transactions in Merge Block*
T_{LB} // Transactions in Length Block

```
1     If Length Block is Empty then
1.1       Add T_f to T_LB
1     Else
2         For i = 0 to N_LB Do
              // L_LB Number of transactions in T_LB
2.1           Calculate RD_{T_f ↔ T_LB}
              // relation distance between every T_LB and T_f
3             If ( RD_{T_f ↔ T_LB} < > RD ) Then
3.1               Add T_f to T_LB
3             Else
3.2               Merge T_f and T_LB
3.3               Add to T_MB
2             End If
1         End For
4     End If
      Return Merge Block and Length Block
```

Algorithm 6: Merge Length Block Algorithm

Inputs: *Length Block Transaction T_{LB}.*
RD
Output: *T_{MB} // Transactions in Merge Block*

```
1     For i = 0 to N_MB // N_MB Number of transactions in T_MB
1.1       Calculate RD_{T_MB ↔ T_LB}
          // relation distance between every T_LB and T_MB
2         If ( RD_{T_MB ↔ T_LB} = RD )
2.1           Merge T_MB and T_LB
2.2           Break // exit from loop
2         End If
1     End For
3     Return Merge Block
```

The *U-Apriori* algorithm uses the data added to Cluster Table and Quantification Table for extracting the frequent itemset. Using equation (1), the expected support for itemset can be calculated according to the clustered items created in the cluster table as follows [3]:

$$ExpSup(X_{CL}) = \sum_{i=1}^{N_{MB}} (\prod_{x \in X} F_i \times \mu_{x_{CL}}) \quad (4)$$

where X_{CL} is itemset represented by the item's clusters, N_{MB} is the number of transactions in merged block; F_i represents the frequency of transferred item in merged block; and $\mu_{x_{CL}}$ symbols the cluster's mean value for specific item cluster from cluster table. Now, suppose that the item *A* is clustered to $A_1, A_2, \ldots, A_n$ cluster. In this case the expected support for *A* can be calculated as [3]:

$$ExpSup(A) = \sum_{i=1}^{N_{MB}} (ExpSup(A_1) + \ldots + ExpSup(A_n)) \quad (5)$$

$$ExpSup(A) = \sum_{i=1}^{N_{MB}} \sum_{j=1}^{n} ExpSup(A_j) \quad (6)$$

$$ExpSup(A) = \sum_{i=1}^{N_{MB}} \sum_{j=1}^{n} (F_{i_{A_j}} \mu_{A_j}) \quad (7)$$

now, suppose that B was clustered to B_1, B_2…,B_m clusters. The expected support for the itemset $\{AB\}$ can be calculated using definition (1) as the probability of occurrence of each cluster belong to A at the same time with the clusters related to B as:

$$ExpSup(AB) = \sum_{i=1}^{N_{MB}} \sum_{j=1}^{n} \sum_{k=1}^{m} ExpSup(A_j B_k) \quad (8)$$

$$ExpSup(A) = \sum_{i=1}^{N_{MB}} \sum_{j=1}^{n} \sum_{k=1}^{m} (F_{i_{A_j}} \mu_{A_j} \times F_{i_{B_k}} \mu_{B_k}) \quad (9)$$

Algorithm 7: U-Apriori Algorithm

Inputs: T_{MB} *// Transactions in Merge Block*
ESupp // user defined expected support
Output: *Frequent ItemSet*

1. Use *Quantification Table* to generate 1 level-itemset and employ (6) to get the expected support for each item through summation of item cluster support multiplied by cluster's mean value.
2. Frequent 1-itemsets that exceed the minimum support are used to generate candidate 2 level-itemsets
3. Use *Quantification Table* to prune non-frequent itemsets.
4. Equation (9) is used to calculate the expected support for 2 level-itemset.
5. Further candidate itemsets with more items are generated and equation (9) is used to calculate the expected support.

V. Experimental Results

The Proposed algorithm was implemented in C# programming language and all experiments run on a PC of Intel Core i5 2.4.0 GHz processor with DDR 4GB main memory. Synthetic datasets are randomly generated for our experiments. With fixed number of items I=10 and averge number of items per transaction T_I =10. Different set of transactions T ranged from 1k to 100k was generated for clustering where C (number of clusters) = 2,3 and 4 to run the proposed algorithm and to compare with *U-Apriori* algorithm.

Fig. 2 show the compersion ratio between the original database and the compresed one for different number of clusters. The compersion percentage increases dramaticly with the increment in transaction numbers. The compersion ratio exhibits the same behavior for different number of cluster values while the minimal number of clusters give the highest compersion percentage.

Fig. 3 shows the percentage of difference between the expected support value for the itemsets resulting from the proposed algorithm and *U-Apriori* algorithm (Percentage Error). The figure shows that the percentage error will be minimal for highest number of clusters with small number of transactions. By incrementing the number of transactions, the percentage error is decremented and approximatly gives the same value with large number of transactions regardless the number of used clusters.

In Fig. 4, the execution time in millisecond for the proposed algorithm was measured in comparison with the *U-Apriori* algorithm for different number of transactions when C = 2 and C = 3. The figure shows that the *U-Apriori* algorithm was better than the proposed algorithm for small number of transactions whatever the number of used clusters. With large number of transactions, the quantification table that is used to prune non-frequent itemsets effects to the perofrmance of the proposed algorithm reseulting execution time better than the *U-Apriori* algorithm.

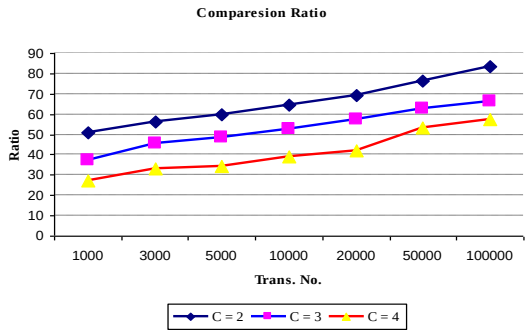

Fig. 2. Compersion ratio with number of transactions.

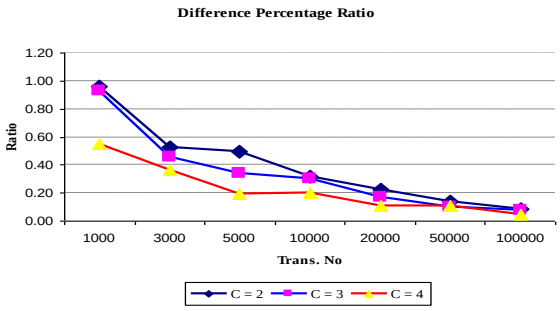

Fig. 3. Deviation from values of U-Apriori algorithm.

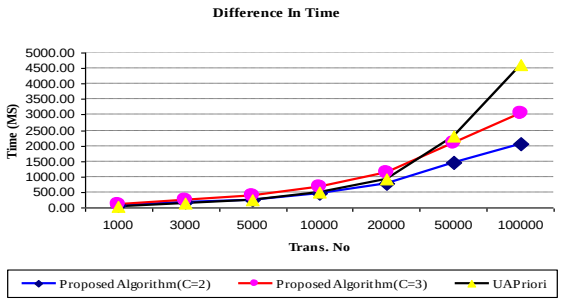

Fig. 4. Execution time difference.

VI. Conclusion and Future Work

In this paper, a new approach is proposed to compress related uncertain database transactions into a new compressed format that is utilized efficiently for mining association rules. There are several advantages of our algorithm over the other approaches: 1) No multiple database scans, because the ystem reads the database only once, 2) Reduce the process time of association rule mining because the sytem prunes candidate itemsets which are impossible to become frequent, 3) A compressed database can be decompressed to the original database to support transaction updates. Our algorithm performs better when number of clusters is low and the complexity increased with the

increment of number of clusters, but the performance still better than *U-Aprioi* algorithm as a result of compression and using quantification table to prune itemsets. Future work includes optimizing the number of clusters to gain highest compression ratio with accepted percentage error.

REFERENCES

[1] A. Ceglar and J. Roddick, "Association Mining," *ACM Computing Surveys*, vol. 38, no. 2, pp. 5-22, July 2006.

[2] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proc. International Conference of Very Large Data Bases*, Swizerland, 1995, pp. 432-444.

[3] C. Chui, B. Kao, and E. Hung., "Mining frequent itemsets from uncertain data," in *Proc. International Conference of Advances in Knowledge Discovery and Data Mining*, China, 2007, pp. 47-58.

[4] R. Naik and J. Mankar, "Mining frequent itemsets from uncertain databases using probabilistic support," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 2, no. 2, pp. 432-437, March-April 2013.

[5] C. C. Aggarwal and Y. Li, "Frequent Pattern Mining with Uncertain Data," in *Proc. International Conference of Knowledge Discovery and Data Mining*, Paris, 2009, pp. 29-38.

[6] T. Calders, C. Garboni, and B. Goethals, "Efficient pattern mining of uncertain data with sampling," in *Proc. International Conference of Advances in Knowledge Discovery and Data Mining*, Hyderabad, 2010, pp. 480-487.

[7] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," in *Proc. International Conference of Very Large Data Bases*, Toronto, 2004, pp. 864-875.

[8] C. Chui and B. Kao, "A decremental approach for mining frequent itemsets from uncertain data," in *Proc. International Conference of Advances in Knowledge Discovery and Data Mining*, Osaka, 2008, pp. 64-75.

[9] M. Ashrafi, D. Taniar, and K. Smith, "A compress-based association mining algorithm for large dataset," in *Proc. International Conference of Computational Science*, Russia, 2003, pp. 978-987.

[10] J. Dai, D. Yang, J. Wu, and M. Hung, "An efficient data mining approach on compressed transactions," *International Journal of Electrical and Computer Engineering*, vol. 3, no. 2, pp. 76-83, February 2008.

[11] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-Trees," *IEEE Transactions on Knowledge and Data Engineering*, vol.17, no. 10, pp. 1347-1362, 2005.

[12] C. Leung, M. Mateo, and D. Brajczuk, "A tree-based approach for frequent pattern mining from uncertain data," in *Proc. International Conference of Advances in Knowledge Discovery and Data Mining*, Osaka, 2008, pp. 653-661.

[13] B. Subbhulakshmi and C. Hemavathi, "A compressed tree-based approach for frequent item sets mining from uncertain data," *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol. 4, pp. 66-71, March-May 2013.

[14] C. Weng and Y. Chen, "Mining fuzzy association rules from uncertain data," *International Journal of Knowledge and Information Systems*, vol. 23, no. 2, pp. 129-152, May 2010.

Mahmoud M. Gabr received his Ph.D. degree from the Manchester University, UK in 1981. His research work concentrates on the field of time series analysis, spectral and bio-spectral analysis, non-linear time series modeling, data mining, integer valued time series models and multivariate analysis. Prof. Gabr is the author of more than 40 articles in peer-reviewed international journals and conferences. Since June 1993 he has been a professor in the department of mathematics and computer science, faculty of science, Alexandria University, Egypt.

Saad M. Darwish received his Ph.D. degree from the Alexandria University, Egypt for a thesis in image mining and image description technologies. His research and professional interests include image processing, optimization techniques, security technologies, database management, and machine learning. Dr. Darwish has published in peer-reviewed journals and conferences and severed as TPC of many international conferences. Since Feb. 2012, he has been an Associate Professor in the department of information technology, IGSR.

Sayed A. Mohsin received the diploma in information technology from the Institute of Graduate Studies and Research (IGSR), Department of Information Technology, University of Alexandria in 2002. He has sixteen years of extensive experience in software architecture, design and development. Mr. Mohsin combines his software development experience and standard software process (SSP) with technical expertise using a variety of programming languages and database systems, including C#, Visual Basic, Java, Power Builder, Micro strategy, Teradata, SQL Server, and ORACLE. Since 2008, he has been a principal Software Team Leader at Amiral Management Corporation.