

TOWARDS AN INCREMENTAL MAINTENANCE OF CYCLIC ASSOCIATION RULES

Eya Ben Ahmed¹ and Mohamed Salah Gouider²

Higher Institute of Management of Tunis, Tunisia

¹Eya.benahmed@gmail.com

²Ms.gouider@isg.rnu.tn

ABSTRACT

Recently, the cyclic association rules have been introduced in order to discover rules from items characterized by their regular variation over time. In real life situations, temporal databases are often appended or updated. Rescanning the whole database every time is highly expensive while existing incremental mining techniques can efficiently solve such a problem. In this paper, we propose an incremental algorithm for cyclic association rules maintenance. The carried out experiments of our proposal stress on its efficiency and performance.

KEYWORDS

Cyclic association rules, Incremental maintenance of cyclic association rules, incremental update of cyclic association rules

1. INTRODUCTION

Knowledge management is the most time consuming and expensive part of our daily lives [1]. In fact, it is crucial to explore a valid knowledge at any moment [2]. Mining association rules can achieve this overarching goal however this task becomes intensely more complicated in front of the wide size of databases, calling up gigabyte, terabyte, or even larger, in some applications [3]. In this respect, extracting association rules has outstandingly grasped the interest of the data mining community.

Through time, the volume of information increases, the databases must be updated with the new amounts of data [4]. Considering that an association rule generates explicitly reliable knowledge according to an explored database at an accurate time.

So that, each update of the database radically overwhelms the already stored patterns. A projection of the database's changes must be drawn on extracted association rules [1]. Since then, several proposals to solve this problem appeared [4, 8, 9, 12, 13].

Parallel to those efforts, cyclic mining of association rules was also investigated on several studies. Such investigations can be found in [5, 6, 7]. The problem of cyclic association rules mining consists in generation of association rules from articles characterized by regular cyclic variation over time. In [5], Ozdon et al. presented the first strategies of cyclic association rules extraction. Then, as a response to the anomalies characterizing the already proposed approaches, a more efficient algorithm was introduced by Ben Ahmed et al. [7].

It can be seen that the research community has proposed separate solutions for the incremental mining and the cyclic association rules mining problems.

In this paper, we present a new algorithm called IUPCAR (Incremental UPDATE of Cyclic Association Rules) for incremental mining of cyclic association rules. This algorithm provides the benefits of fast incremental mining and efficient cyclic association rules extraction.

The rest of the paper is organized as follows: section 2 studies the fundamental bases on which our proposal is built. Section 3 presents a formal description of the problem. Section 4 details our proposal and offers an illustrative example to scrutinize the mechanism of our approach.

Carried out experiments stressing on the efficiency of our proposal are sketched in section 5. Finally, section 6 concludes the paper and points out avenues for future work.

2. FOUNDATIONS OF THE PROPOSED ALGORITHM

This section provides the theoretical foundations on which is based our proposed algorithm. These bases concern the theoretical foundations of the two problems of cyclic association rules mining and incremental mining of association rules. Therefore, we present the cyclic association rules concept. Finally, we describe the incremental mining problem.

2.1. Cyclic Association Rules

Considering temporal transactional databases, several temporal patterns can be extracted [17]. Typically, one of them is related to cyclic association rules. Its general idea is to extract correlations between products which vary in a cyclic way. Thanks to its comprehension of the data, we presume that the user is the most advantaged to fix the best length of cycle according to considered products. Hence, we can analyze on depending on monthly, weekly, daily or even hourly sales of products with the respect of the length of cycle.

For example, let September and October sales in a bookstore transaction database be shown in Table 1. In fact, we assume at the beginning of September according to the transactions 9.1 and 9.2, that the sales of books and notebooks are highly important. This can be explained by the start of the academic year. Besides, this correlation is underlined also at the beginning of February by transactions 2.1 and 2.2. This fact is due to the start of the second semester of the academic year. Effectively, the considered cycle in this case is the length of the semester namely six months. The outcomes of these analyses are then used to support various business decisions in this bookstore.

<i>TRANSACTION ID</i>	<i>ITEMS</i>
9.1	<i>Books ,notebooks</i>
9.2	<i>Books, notebooks</i>
...	...
2.1	<i>Books, notebooks</i>
2.2	<i>Books, notebooks</i>

Table 1 . Transactional database DB in a bookstore

We present the basic concepts related to cyclic association rules that will be of use in the remainder.

Time unit Considering the temporal aspect, the first considered measure is the time unit. Firstly, it was introduced by Odzen et al [5].

Definition 1. Given a transactional database DB, each time unit u_i correspond to the time scale on the database [5].

<i>TRANSACTION ID</i>	<i>ITEMS</i>
1	B
2	A,B
3	A,B,C,D
4	A,B,C
5	C
6	A

Table 2. Initial database DB

Example 1 Let the following example be highlighted in table 2. The transactions illustrated in table 2 are extracted hourly. So that, the time unit is the hour.

Cycle The concept of cycle was primarily introduced by Odzen et al [5].

Definition 2. A cycle c is a tuple (l, o) , such that l is the length cycle, being multiples of the unit of time; o is an offset designating the first time unit where the cycle appeared.

Thus, we conclude that $0 \leq o < l$.

Example 2 If we consider a length of cycle $l=2$ and the corresponding offset is 1. So that, the cycle $c=(l,o)=(2,1)$.

Several approaches addressing the cyclic association rules issue are proposed in the literature:

- **The Sequential Approach:** is a two-phase based algorithm.

The basic idea is:

- (1) To generate the large itemsets and to extract straightforwardly the corresponding association rules. Undoubtedly, all small k -itemsets are discarded during the support-pruning step and all weak rules are deleted during the confidence-pruning step.
- (2) To detect the cycles of rules. Only, those are cyclic will be kept and the remainder is pruned.

- **The Interleaved Approach:** is a three-phase based algorithm.

Three optimization techniques are used:

- **Cycle pruning:** Technique for approximating the cycles of itemsets. In fact, we consider: "*If an itemset X has a cycle, then any of the subsets of X has the same cycle*";
- **Cycle skipping :** Technique for avoiding counting the support of an itemset in time units, which we know, cannot be part of a cycle of the itemset;
- **Cycle elimination:** Technique relying on this property: "*If the support for an itemset X is below the minimum support threshold in time segment then X cannot have any of the cycles in subtime segments.*"

Thanks to cycle pruning, we generate the potential cycles. For every unit of time, we apply the cycle skipping to extract the itemsets and we count their support thanks to the cycle elimination.

- **The PCAR Approach:** Radically, it is based on the segmentation of the database in a number of partitions fixed by the user. The scan of the database is done sequentially partition by partition.

This latter is scanned to generate the frequent cyclic itemsets. Achieving the last one, we obtain the set of frequent cyclic itemsets. Thus, we extract from them the cyclic association rules.

2.2. Incremental Maintenance of Association Rules

Various streams of approaches were reported to update incrementally the discovered association rules. We present the most well-known ones.

Firstly, the strategic idea of incremental update was proposed by Cheung et al by introducing FUP for incrementally updating frequent itemsets [8]. This approach assumes batch updates and takes advantage of the relationship between the original database DB and the incrementally added transactions db . Inspired from the Apriori algorithm, the basic idea of FUP is that by adding db to DB , some previously frequent itemsets will remain frequent and some previously infrequent itemsets will become frequent (these itemsets are called winners). At the same time, some previously frequent itemsets will become infrequent (these itemsets are called losers). The major contribution of FUP is to use information in db to filter out some winners and losers, and therefore reduce the size of candidate set in the Apriori algorithm. FUP improves the

performance of Apriori greatly because the performance of this latter relies heavily on the size of candidate set.

The BORDERS Algorithm proposed by Thomas et al. [9] and Feldman et al.[10], is another approach using the concept of "negative border" introduced by Toivonen [11] aiming to indicate if it is necessary or not to check any candidate against the initial database. The main contribution outlined is its need at most of one scan of the original database in the update operation. In this respect, the algorithm maintains information about the support of frequent itemsets in the original database along with the support of their negative border. If any itemset becomes a winner in the updated database, it follows that some itemset formerly in the negative border will also become a winner. Consequently, the negative border can be considered as an indicator for the necessity of looking for winners in the original database. If no expansion happens in the border, no need brings a point up to scan the original database.

The WEIGHT approach is another strategy for maintaining association rules in dynamic databases proposed by Shichao Zhang et al. [12]. This method used weighting technique to highlight new data. Indeed, it is a four-phase based approach: (i) Firstly, all frequent and hopeful itemsets related to the initial database are stored; (ii) Secondly, each incremental dataset DB_i is mined and all frequent itemsets are stored. According to the requirements given by users, an assignment of a weight to each set DB_i is subjectively done, (iii) Thirdly, the aggregation of all rules based on hopeful itemsets by weighting is accomplished, (iv) Finally, a selection of high rank itemsets is achieved being the founded output.

3. INCREMENTAL MINING OF CYCLIC ASSOCIATION RULES

Along this section, we present a description of the tackled problem. First, we formally define the problem of cyclic association rules. Then, we present the basic notions.

3.1 Formal problem description

Regarding cyclic association rules, we stress on rules that are repeated in a cyclical way. Indeed, given a length of cycle, we extract itemsets that appear sequentially in the database. Let consider X and Y two itemsets appearing in DB at transaction number i and sequentially at the transaction number $i+length$ of cycle until the end of the database (Table 3). According to given support threshold, we prune the non frequent cyclic itemsets. Thus, we generate the cyclic association rules based on minimum confidence threshold.

TRANSACTION_ID	Items
i	X,Y
$i+length$ of cycle	X,Y
...	...
$i+(length \text{ of cycle} * k)$	X,Y

Table 2. Initial database DB

To summarize, the cyclic association rules mining problem can be reduced to extraction of frequent cyclic itemsets, because once we have frequent cyclic itemsets set, cyclic association rules generation will be straightforward.

After several updates of DB, an increment db of ldbl transactions is added to DB. The problem of incremental maintenance of cyclic association rules is to compute the new set of the frequent cyclic itemsets in $DB'=DB \cup db$ according to a support threshold *MinSup*.

In order to extract cyclic association rules from databases, the only plausible solution is to rerun one of the classical algorithms dedicated to the generation of cyclic association rules i.e., *Sequential*, *Interleaved* or *PCAR*. As a result, two drawbacks are quoted:

- If the original database is large, much computation time is wasted in maintaining association rules whenever new transactions are generated;
- Information previously mined from the original database, provides no help in the maintenance process.

3.2 Basic notions

We start this subsection by presenting the key settings that will be of use in the remainder.

Frequent Cyclic itemset: This concept refers to cyclic itemsets having supports exceeding the considered threshold. The formal definition is as follows.

Definition 3.1

Let XY be an itemset, the $\text{sup}(XY)$ is the support of the itemset in the database, reminding that only cyclic occurrences are considered on the support computing, and the minimum support threshold reminding MinSup .

The itemset XY is considered as **Frequent Cyclic** denoted **FC** if the cyclic occurrences of the itemset XY are greater or equal to the given support threshold otherwise if $\text{sup}(XY) \geq \text{MinSup}$.

Example 3

We consider the context shown by table 2, the MinSup equal to 2 and the length of cycle is 2. The binary sequence representing the itemset AB is 011100 so $\text{sup}(AB) = \text{MinSup} = 2$ then AB is called **Frequent Cyclic itemset FC**.

Frequent Pseudo-Cyclic itemset: The frequent pseudo-cyclic concept is presented as follows.

Definition 3.2

Let XY be an itemset, the $\text{sup}(XY)$ is the support of the itemset in the database, the MinSup the minimum support threshold. The itemset XY is considered as frequent pseudo-cyclic denoted **FPC** if its support is less than MinSup . Simultaneously, the computed support is greater than a given threshold called **MinFPC**.

$$\text{MinFPC} \leq \text{sup}(XY) < \text{MinSup}.$$

Example 4

Given the previous context, we consider MinSup equal to 2, the MinFPC is 0.2 and the length of cycle is 2. The binary sequence representing the itemset AD is 000100 so $\text{sup}(AD) = 1 < \text{MinSup} = 2 \geq \text{MinFPC} = 0.2$ then AD is called **Frequent Pseudo-Cyclic itemset FPC**.

Minimum FPC threshold: According to this measure, we classify the remainder of the itemsets after MinSup pruning on hopeful cyclic itemsets that are not frequent in the initial database but are more likely to move to this status in the increment database.

Definition 3.3

The Minimum FPC threshold, denoted by MinFPC , refers to a threshold dedicated to prune the none hopeful itemsets. It is computed according to this formula:

$$\text{MinFPC} = [(\text{MinSup} / |\text{DB}| + |\text{db}|) + \text{MinSup}] / [|\text{DB}| + |\text{db}|]$$

Example 5

Continuing with the same database DB considered as initial one, let the database db containing 4 transactions be the increment one. In addition we fix the MinSup to 2. Then the MinFPC is computed as follows:

$$\text{MinFPC} = [(2 / (6 + 4)) + 2] / (6 + 4) = 0.2$$

Non Frequent Cyclic Itemset: This concept refers to cyclic itemsets that are not both frequent cyclic itemsets and frequent pseudo-cyclic itemsets.

Definition 3.4

Let XY be an itemset, the $\text{sup}(XY)$ is the support of the itemset in the database and the MinSup the minimum support threshold. The itemset XY is considered as **non frequent cyclic itemset** denoted **NFC** if the cyclic occurrences of the itemset XY are less than the given MinFPC threshold otherwise if $\text{sup}(XY) < \text{MinFPC}$.

Example 6

Given the previous context, we consider *MinSup* equal to 4, the *MinFPC* is 2 and the length of cycle is 2. The binary sequence representing the itemset AD is 000010 so $\text{sup}(AD)=1 < \text{MinFPC}=2 < \text{MinSup}=4$ then AD is called non frequent cyclic itemset denoted NFC.

The major contribution of this paper is to propose a new strategy dedicated to the incremental update of cyclic association rules aiming to reduce efficiently the runtime required for the generation of cyclic association rules in the case of addition of transactions at the maintenance process of databases. Indeed, this proposal is detailed in the following section.

4. IUPCAR ALGORITHM

In order to maintain incrementally the cyclic association rules, we introduce a novel approach called Incremental Update of Cyclic Association Rules denoted IUPCAR. Indeed, the IUPCAR algorithm operates in three phases:

- In the first phase, a scan of the initial database is done to class the founded itemsets on three classes namely the frequent cyclic itemsets, the frequent pseudo-cyclic itemsets and non frequent cyclic itemsets.
- In the second phase, according to the second database, we categorize the itemsets into the three mentioned classes. Then, depending of the ancient class of the itemset with its ancient support and the new class with its new support in the increment database, an affectation of the suitable class is made according to a weighting model.
- In the final phase, given the founded frequent cyclic itemsets after the update operation, the corresponding cyclic association rules are generated.

As highlighted by figure 1, first and foremost, the IUPCAR algorithm takes on input the initial database, the minimum support threshold *MinSup*, the minimum confidence threshold *MinConf* and the length of cycle. According to those key settings, a generation of frequent cyclic itemsets, frequent pseudo-cyclic itemsets and non frequent cyclic itemsets from the initial transactions is done. Stressing on the dynamistic feature of the databases, we add the novel transactions building the db database.

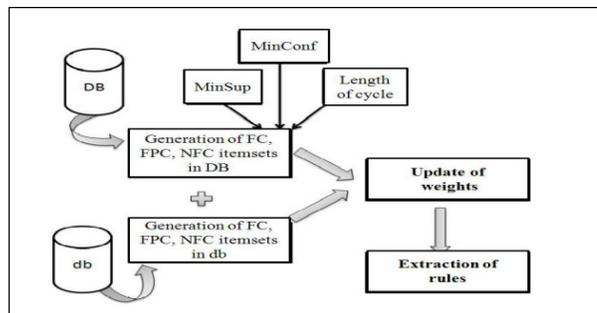


Fig 1. The flowchart of IUPCAR.

To accomplish this update operation, a scan of the new database is done and a generation of the itemsets and their classification are straightforwardly realized. After that, an update of the status and the weights of itemsets are done without rescanning the initial database. Finally, we generate the cyclic association rules based on the retained frequent cyclic itemsets.

Intuitively in the updating problem, we assume the following cases shown by table 4:

db\DB	FC	FPC	NFC
FC	Always FC (case A)	Computation based on db and DB (case B)	Computation based on db and DB (case C)
FPC	Computation based on db and DB (case D)	Always FPC (case E)	Computation based on db and DB (case F)
NFC	Computation based on db and DB (case G)	Computation based on db and DB (case H)	Always NFC (case I)

Table 4. Possible cases in update operation.

- Frequent cyclic itemset FC is already saved as frequent cyclic FC (case A), frequent pseudo-cyclic FPC (case B) or non frequent cyclic itemset NFC (case C);
- Frequent pseudo-cyclic itemset FPC is already saved as a frequent cyclic FC (case D), frequent pseudo-cyclic FPC (case E) or non frequent cyclic itemset NFC (case F);
- Non frequent cyclic itemset NFC is already saved as frequent cyclic FC(case G), frequent pseudo-cyclic FPC (case H) or non frequent cyclic itemset NFC (case J).

To handle those various cases, we introduce the following weighting model.

In the update operation, a dramatic change in the status of the itemsets between the first and the coming database is intuitively plausible. That's why, we refer to the weighting model as a technique dedicated to decide which status is the suitable to the itemset after the new added transactions.

Indeed, we sketch the mechanism of weighting model as follows.

For an itemset X, we:

- compute the relative support of X in the initial database DB, denoted by $Sup(X_{DB})$, computed according to the given formula:

$$Sup(X_{DB}) = Sup(X) / |DB|$$

- compute the relative support of X in the increment database db, denoted by $Sup(X_{db})$, computed according to the given formula:

$$Sup(X_{db}) = Sup(X) / |db|$$

- compare the relative support of X in the initial database $Sup(X_{DB})$ vs. that of the increment database $Sup(X_{db})$.

And we choose the greatest one.

- Two alternatives are plausible:

1. If the itemset has the same state in the initial and the incremental database, we will enhance its weight;
2. If the state of the itemset has changed from the initial to the incremental database, we will check which one of its states has the greatest weight and we will decrease its value and affect this state as its new one.

In this respect, let the new weight of X be denoted by $W(X_{db})$.

The table 4 sketches the possible cases that can be summarized on three possible scenarii:

1. No change in the status simply happens. So, the itemset remains frequent cyclic FC (case A) or frequent pseudo-cyclic itemset FPC (case E) or non cyclic frequent CNF (case J). The new weight is computed as follows:

$$W(X_{db}) = [Sup(X_{DB}) + Sup(X_{db})] / [|DB| + |db|]$$

2. A change in the status between the initial transactions and the new ones occurs. So one of the cases depicted on the table 4 by (case B), (case C), (case D), (case F), (case G) or (case H) happens. Then, two situations are obviously outlined:

(a) If the previous support of the itemset is greater than the new one in the increment database, the affected status is remained the same and its novel weight is computed as follows:

$$W(X_{db}) = [Sup(X_{DB}) / |DB|] - [Sup(X_{db}) / |db|]$$

(b) If the new support of the itemset is greater than the previous one, the affected status is the new one and its novel weight is computed as follows:

$$W(X_{db}) = [Sup(X_{db}) / |db|] - [Sup(X_{DB}) / |DB|]$$

– Considering the update operation of the itemsets’ status and weights, we extract cyclic association rules based on frequent cyclic itemsets.

5 IUPCAR EXAMPLE

Aiming to illustrate deeply the mechanism of our approach with its different steps, we consider the context sketched in table 2 as an initial database.

We introduce the following parameters: Length of cycle equal to 2; *MinSup* equal to 50% and *ldbl* equal to 4. We propose to illustrate the possible cases; we choose one itemset to facilitate the explanation of our proposal. Indeed, based on the initial database, we can extract:

– classified as FC: AB and classified as FPC: AC and classified as NFC: AD.

For the itemset AB, recognized as FC, we simulate the various cases that can be handled in the update operation.

Furthermore, the table 5 summarized the possible new status of the itemset AB and its eventual supports in db.

According to db, we find:

1. *Scenario (a)*: AB is generated as a FC:

$$W(AB_{db}) = [Sup(AB_{DB}) + Sup(AB_{db})] / [|DB| + |dbl|] = [3+2] / [6+4] = 1/2$$

So that the new state affected is clearly F C but the weight of AB is increased due to its keeping the same status in DB and db;

2. *Scenarii (b, c)*: AB is generated as a F P C:

(a). *Scenario (b)*: the support of AB in DB is greater than the support of AB in db

$$W(AB_{db}) = [Sup(AB_{DB}) / |DB|] - [Sup(AB_{db}) / |db|] = 1/2 - 1/4 = 1/4$$

(b). *Scenario (c)*: the support of AB in DB is less than the support of AB in db

$$W(AB_{db}) = [Sup(AB_{db}) / |db|] - [Sup(AB_{DB}) / |DB|] = 3/4 - 1/2 = 1/4 .$$

3. *Scenarii (d, e)*: AB is generated as a FCN:

(a). *Scenario (d)*: the support of AB in DB is greater than the support of AB in db

$$W(AB_{db}) = [Sup(AB_{DB}) / |DB|] - [Sup(AB_{db}) / |db|] = 1/2 - 1/4 = 1/4.$$

(b). *Scenario (e)*: the support of AB in DB is less than the support of AB in db

$$W(AB_{db}) = [Sup(AB_{db}) / |dbl|] - [Sup(AB_{DB}) / |DB|] = 3/4 - 1/2 = 1/4.$$

AB	FC	FPC		NFC	
Sup(AB _{db})	2	1	3	1	3
	<i>Scenario (a)</i>	<i>Scenario (b)</i>	<i>Scenario (c)</i>	<i>Scenario (d)</i>	<i>Scenario (e)</i>

Table 5. The possible cases in db for FC itemset

Consequently, the new state affected to AB is FC (*Scenarii (b, d)*) because its support in DB is greater than its one in db. Nevertheless, the new affected status will have a weight less than the previous support because in the incremental database, we notice the change of its status;

Likewise, we affect for AB FPC or FCN (*Scenarii (c, e)*) as new status because its support in the incremental database is greater than its one in the initial database.

However, the new affected status will have a weight less than the one extracted in the incremental database because it does not maintain its initial status.

Similarly, the new two states of itemsets AC and AD are respectively as FPC and NFC, the same possible scenario simulated for AB are obviously in the update operation plausible. As a final step, after the update of the status of the itemsets and their weights, only frequent cyclic itemsets are considered in the extraction of the novel cyclic association rules related to the both databases DB and db.

6. EXPERIMENTAL STUDY

To assess the IUPCAR efficiency, we conducted several experiments on a PC equipped with a 3GHz Pentium IV and 2GB of main memory. The figure 2 is an illustration of the IUPCAR user-interface. During the carried out experimentation, we used benchmarks datasets taken from the UC Irvine Machine Learning Database Repository. Table 6 depicts the characteristics of the datasets used in our evaluation.

DATABASE	#TRANSACTIONS	#ITEMS	#AVERAGE SIZE OF TRANSACTIONS	SIZE (KO)
T10I4D100K	100000	1000	10	3830
T40I10D100K	100000	775	40	15038
Retail	88162	16470	10	4070

Table 6: Description of benchmark databases

It shows the number of items, the number of transactions, the average size of transactions and the size of each database. Through these experiments, we have a twofold aim: first, we have to stress on the performance of our proposal by the variation of *MinSup* on the one hand and the variation of the cardinalities of initial and incremental databases on the other hand. Second, we put the focus on the efficiency of our approach vs. that proposed by the related approaches of the literature.

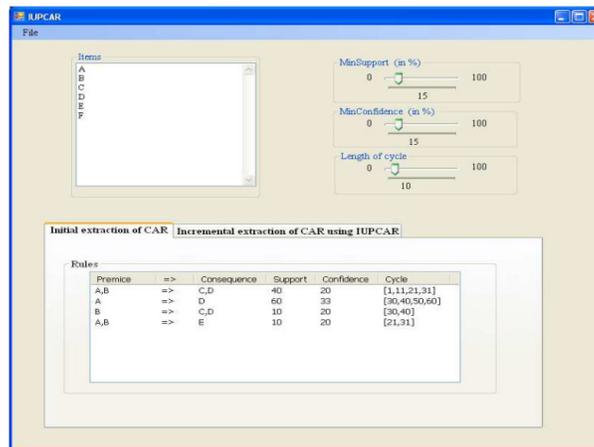


Fig 2. The user-interface design of IUPCAR

6.1 PERFORMANCE ASPECT

▪ Minimum support variation

In the carried out experimentations, we divided database in two partitions: DB is the initial database and db is the incremental one. Firstly, the DB is constituted of 70% of the size of the benchmark dataset and db is constituted of the remainder namely the 30%. Secondly, we increase the size of the initial database to achieve 80% from the size of the benchmark dataset

so the db presents only 20%. To finish with an initial database representing 90% and an incremental one providing only 10%.

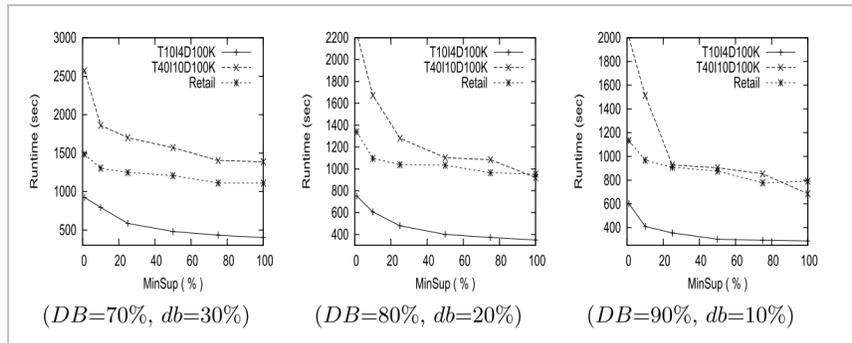


Fig 3. Experimental results of IUPCAR for an incremental database =10%, 20% or 30%

Considering the given parameters: the *MinConf* =50%, the length of cycle =30, we present the variation of *MinSup* and the corresponding runtime of IUPCAR in figure 3.

Indeed, by varying the support, it is obvious that the more support is increasing the more runtime of IUPCAR decreases. For the T10I4D100K dataset, having DB=70% and db=30%, the runtime of IUPCAR increases from 926,295 seconds for 1% as a *MinSup* to 482,726 seconds for 50% as a *MinSup*, to stabilize at around 400, for *MinSup* exceeding 50%. As expected, this fact is similarly conceivable for T40I10D100K and Retail datasets.

As shown figure 3, we assume worthily that on whatever the size of the initial and increment database, the more support increases the more runtime required for IUPCAR goes down.

▪ **Variation of the size of updated database**

In this part, we concentrate on the effect of variation of initial and incremental databases sizes. Indeed, fixing parameters as follows: *MinConf* equal to 50% and length of cycle=3, according to the figure 4, we perceive regarding the T40I10D100K dataset for the same value of support equal to 50%, with DB=70% and db=30% the update operation requires 1571,541 seconds but this value goes down if we rise the size of the initial database and we diminish the size of the increment one. Identically, we notice 1102,84 seconds if the DB=80% and db=20% by far 904,254 seconds if the DB=90% and db=10% of the whole dataset. Therefore, it is crucial to deduce that the least is the size of the incremental database, the least is the runtime required to update the cyclic rules and this can be noted for the two other datasets namely retail and T10I4D100K.

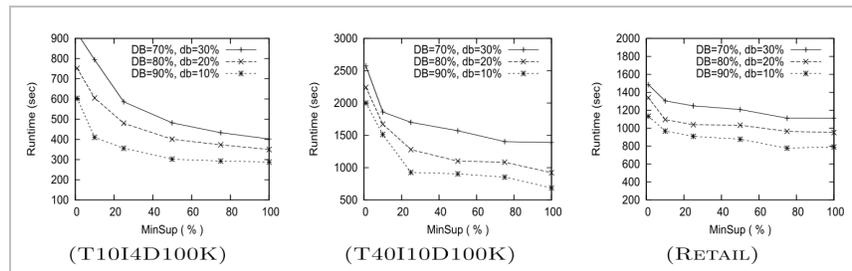


Fig 4. Comparing the runtime of IUPCAR with different incremental databases 10%, 20% and 30%.

6.2 EFFICIENCY ASPECT

In order to evaluate the efficiency of our algorithm, we conducted comprehensive experiments to compare IUPAR with the most efficient classical algorithm dedicated to cyclic association rules extraction namely PCAR algorithm.

The following values of parameters are set during the several experiences: the minimum of confidence equal to 50%, the length of the cycle equal to 30 and the runtime of the algorithms regarding the T10I4D100K, T40I10D100K and Retail datasets. The results of varying the minimum support on running the PCAR algorithm and the IUPCAR one are shown by figure 5. It indicates that the update operation of DB by the increment $db=10\%$ with a minimum support $=100\%$, requires 511,388 seconds by running PCAR and only its half 288,62 seconds by running IUPCAR for T10I4D100K dataset. For T40I10D100K, the original database $DB = 90\%$ with the increment database $db = 10\%$ required with a minimum support $=100\%$ for T40I10D100K by running PCAR 1776,12 seconds and interestingly its half 686,884 seconds for IUPCAR running. Similarly for Retail, the updating operation of the initial database by adding 10% of its size with a minimum support $=100\%$ requires 1894,416 seconds by running PCAR and efficiently 791,9 seconds for IUPCAR running.

Obviously, IUPCAR amply outperforms the PCAR algorithm in the context of maintenance of cyclic association rules and proves its efficiency in various test cases.

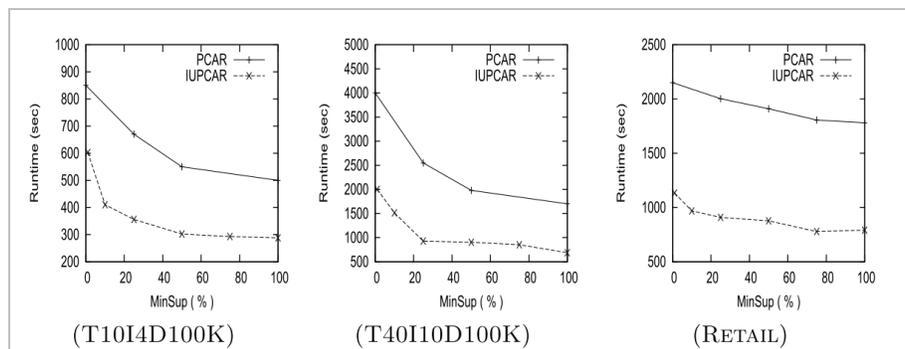


Fig 5. Comparing the runtime of IUPCAR and PCAR with incremental database

7. CONCLUSION

In this paper, we introduced the problem of incremental maintenance of cyclic association rules. Thus, the flying over the pioneering approaches handling the incremental update of association rules issue [13] conducted us to introduce a new proposal called IUPCAR algorithm dedicated particularly to update the cyclic association rules. To evaluate its efficiency, several experimentations of the proposed method are carried out. So that, encouraging results are obtained.

Future work will focus mainly on: (i) the quality of the generated cyclic association rules. In fact, we plan to study deeply the significance of the extracted cyclic association rules for human experts [14] [15], (ii) tackling the change support threshold in the incremental update operation of cyclic association rules [16], (iii) using database vertical representation (Eclat (Zaki et al, 1997) [18]) to improve the IUPCAR results.

ACKNOWLEDGEMENTS

We would like to thank everyone.

REFERENCES

- [1] Yew-Kwong Woon, Wee-Keong Ng and Ee-Peng Lim, (2009), Association Rule Mining, Information System, 77-82.
- [2] R. Agrawal, T. Imielinski and A. N. Swami, (1993), Mining association rules between sets of items in large databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, D.C., 207-216.
- [3] R. Srikant and R. Agrawal, (1994) ,Fast algorithms for mining association rules. Proceedings of the 20th Conference on Very Large Data Base (VLDB), Santiago, Chile, 478-499.
- [4] S.D. Lee David W. Cheung Ben Kao, (1998), Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovered Association Rules, Data Mining and Knowledge Discovery, Volume 2, Issue 3 , 233-262.
- [5] B. Ozden, S. Ramaswamy and A. Silberschatz, (1998), Cyclic Association Rules, 14th International Conference on Data Engineering (ICDE'98),412, 16.
- [6] E. Ben Ahmed and M.S. Gouider, (2010), PCAR : nouvelle approche de génération de règles d'association cycliques, EGC, 673-674, DBLP:conf/f-egc/2010, DBLP, <http://dblp.uni-trier.de>.
- [7] E. Ben Ahmed and M.S. Gouider, (2010), Towards a new mechanism of extracting cyclic association rules based on partition aspect, Fourth International Conference on Research Challenges in Information Science, RCIS, Nice, France.
- [8] David W. Cheung, C. Y. Wong, Jiawei Han and Vincent T. Ng, (1996), Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique, Data Engineering, International Conference on, Los Alamitos, CA, USA, 106.
- [9] T. Shiby, B. Sreeath, K. Alsabti and R. Sanjay, (1997), An efficient algorithm for the incremental updation of association rules in large databases. In Proceedings of the 3rd International conference on Knowledge Discovery Data Mining (KDD 97), New Port Beach, California.
- [10] R. Feldman, Y. Aumann, A. Amir, and H. Mannila, (1997), Efficient Algorithms for discovering Frequent Sets in Incremental Databases, In Proceedings of the 1997 SIGMOD Workshop on DMKD, Tucson, Arizon.
- [11] Toivonen, H. Sampling large databases for association rules, (1996), In 22nd International Conference on Very Large Databases (VLDB'96), 134-145, Mumbai, India.
- [12] Shichao Zhang, Chengqi Zhang and Xiaowei Yan, (2003), Postmining: maintenance of association rules by weighting, Information Systems, 28, 7,691–707, [http://dx.doi.org/10.1016/S0306-4379\(02\)00079-0](http://dx.doi.org/10.1016/S0306-4379(02)00079-0), Elsevier Science Ltd., Oxford, UK.
- [13] S. D. Lee and Hong Kong and David W. Cheung, (1997), Maintenance of Discovered Association Rules: When to update?, In Research Issues on Data Mining and Knowledge Discovery, 51-58.
- [14] Jieh-Shan Yeh and Chih-Yang Chang and Yao-Te Wang, (1998), Efficient algorithms for incremental utility mining, Proceedings of the 2nd international conference on Ubiquitous information management and communication, 212-217.
- [15] Ming-Cheng Tseng and Wen-Yang Lin and Rong Jeng, (1997), Mining Association Rules with Ontological Information, Innovative Computing, Information and Control, 2007. ICICIC apos;07. Second International Conference on Volume, Volume 2, Issue 3, 300.
- [16] Tseng, Ming-Cheng and Lin, Wen-Yang, (2004), Maintenance of generalized association rules with multiple minimum supports, Intell. Data Anal., 8, 4, 417-436, IOS Press, Amsterdam, The Netherlands, The Netherlands.
- [17] Vincent Ng, Stephen Chan, Derek Lau, and Cheung Man Ying, (2007), Incremental Mining for Temporal Association Rules for Crime Pattern Discoveries, Proceedings of the eighteenth conference On Australasian database, ACM International Conference Proceeding Series, 691-707.
- [18] Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W, (1997), New Algorithms for fast discovery of Association Rules. In Proceedings of the 3rd Intl Conference on KDD and data mining (KDD97), Newport Beach, California, 283-284.