

# Origin based Order Independent Association Rule Mining using multiple OIMASP tree<sup>☆</sup>

Elsevier<sup>1</sup>

*Radarweg 29, Amsterdam*

*Elsevier Inc<sup>a,b</sup>, Global Customer Service<sup>b,\*</sup>*

*<sup>a</sup>1600 John F Kennedy Boulevard, Philadelphia*

*<sup>b</sup>360 Park Avenue South, New York*

---

## Abstract

We discuss the problem of mining association rules in the transaction dataset. The previous approach(MASP) is dependent on the order of items in the dataset. We proposed a novel method which is order independent and also takes into consideration the origin of items to calculate unbiased support and confidence values. This approach outperforms the previous approach concerning the number of association rules and length of the longest association rule.

*Keywords:* Data Mining, Association Rule Mining, MASP, Support, Confidence, Transaction Database, Transaction

---

## 1. Introduction

Association rule mining is a rule-based machine learning procedure to find interesting patterns in the transaction database based on individual and conditional frequencies. In the traditional approach, two steps are involved in generating rules. First, generate all frequent itemsets and pruned non-frequent ones and then in the second stage rules are derived from those frequent itemsets. An association rule e.g. {bread, milk}  $\Rightarrow$  {butter} in market basket analysis

---

<sup>☆</sup>Fully documented templates are available in the elsarticle package on CTAN.

<sup>\*</sup>Corresponding author

*Email address:* [support@elsevier.com](mailto:support@elsevier.com) (Global Customer Service)

*URL:* [www.elsevier.com](http://www.elsevier.com) (Elsevier Inc)

<sup>1</sup>Since 1880.

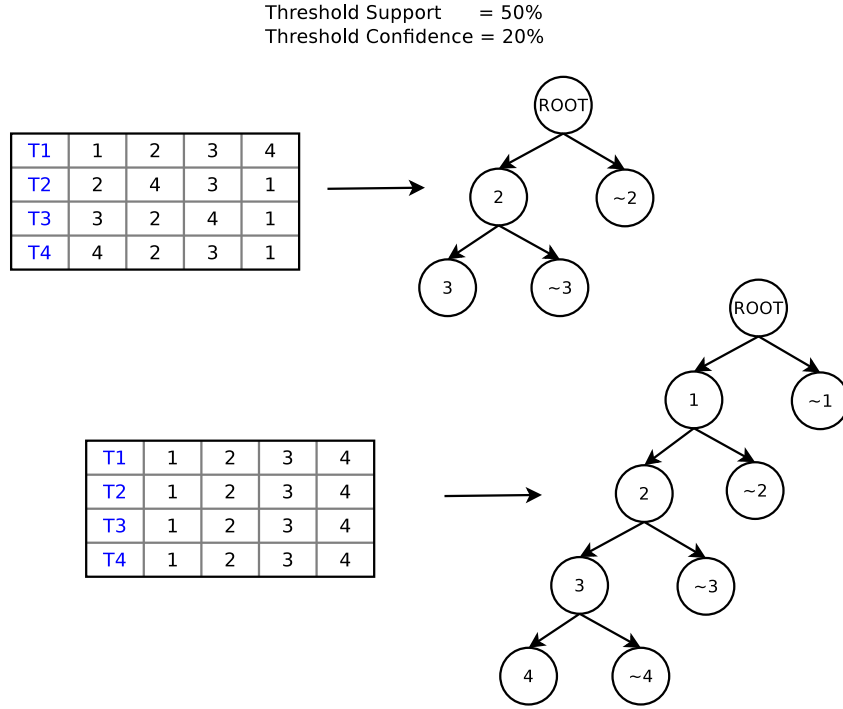


Figure 1: Different *MASP* tree for the same dataset

means if one purchase bread and milk together it is highly likely that they will also buy butter. Apart from market basket analysis, association rule mining is useful in intrusion detection, bioinformatics, and many other applications.

In 2014 O. M. Soyal [1] proposed a new approach to extract mostly associated sequential patterns (MASPs) using less computational resources in terms of time and memory while generating a long sequence of patterns that have the highest co-occurrence.

This approach may produce different outcomes if we change the order of items in transactions. Figure 1 shows this issue. We propose an approach which is order independent. An association rule of the form  $A \Rightarrow B$  must satisfy the threshold support and threshold confidence i.e. probability of occurrence of  $A$  and  $B$  together must surpass threshold support, and the likelihood of occurrence of  $B$  in transactions containing  $A$  must be greater than or equal

to threshold confidence. It means, to calculate support and confidence, it is required to traverse complete transaction database. What if an item appears for the first time in the  $i$ th transaction? It is biased to take the entire dataset for calculating support and confidence for the rules containing that particular item. So to generate all rules containing a particular item  $x$ , it is reasonable to ignore all transactions(for calculating support and confidence) that come before the transaction in which that particular item appears for the first time. Taking into account the origin of the item is the second proposed improvement. Embedding these two changes to the Omer M. Soyal [1] approach is the basis of our research.

## 2. Related works

In 1994 R. Agrawal, et al. published non-trivial algorithm(Apriori) [2] for finding association rules in large databases of the sales transaction. Apriori algorithm produces association rules in two steps. First generates all frequent itemsets(prune non-frequent candidate itemsets) and then make rules from those itemsets. This algorithm first finds frequent itemsets of length one then frequent itemsets of length 2 using frequent itemsets of length 1 and so on until generation of all frequent itemsets. This algorithm gave the better result than the previously known fundamental algorithms AIS [3], SETM [4]. In 1996 Fukuda, et al. [5] proposed an approach to find two-dimensional association rules. A state in this scenario is of the form  $((X, Y) \in P) \Rightarrow (Z = z)$  where  $X$  and  $Y$  are numeric attributes,  $P$  is a subspace of 2-D plane, and  $Z$  is boolean attribute i.e.  $z$  can be either true or false. E.g.  $(\text{Age} \in [30, 50] \wedge \text{Balance} \in [10^5, 10^6]) \Rightarrow (\text{CardLoan} = \text{yes})$ . It means if a bank user age and balance lies in the given subspace it is very likely that they will use card loan. This approach works for specific types of structured data. R. Feldman, et al.(1997) [6] introduced the notion of maximal association rules. These are the rules extracted from frequent maximal itemsets. Frequent maximal itemsets are those itemsets which appear just once among all transactions. It is useful in finding association rules containing negated attributes. As an example a rule  $\{\text{milk}, \neg\text{bread}\}$

$\Rightarrow \{\neg\text{butter}\}$  contains negated attributes. It means if a user purchases milk but not bread then the probability that the user will not buy butter is very high. This approach helps to capture inference rules which might be lost using regular associations. Till now items in transaction databases were treated uniformly. In 1998 C.H. Cai, et al. [7] gave an approach to find association rules which take into account weight(importance) of items in transaction databases. FP-Growth algorithm(2000) [8] also take two steps. The second phase is same as apriori. FP-Growth does not generate candidate frequent itemsets. First, it creates a tree(FP-Tree) and then finds frequent itemsets. This algorithm is about an order of magnitude faster than the Apriori algorithm. Lin, Weiyang, et al. [9] proposed an approach that uses association rule mining for collaborative recommender systems. This approach does not require threshold support value. Instead, based on the number of rules(given) to be generated, threshold support is decided by the system. Thus it reduced the running time and produced enough rules for good recommendation performance. In 2004 F. Conen, et al. [10] proposed two structures(T-Trees and P-Trees) which offer improvement concerning storage and execution time. In 2005 K. G. Srinivasa, et al. [11] took advantage of genetic algorithms principles to generate large itemsets within dynamic transaction database. Their algorithm was better than the pre-existing FUP and E-Apriori concerning execution time and scalability. If transaction database is static, then life will be easy. In other scenario transaction database keeps on changing at high speed leading to change in data distribution. Hence it will be difficult to apply previously mentioned Association Rule Mining techniques. Jiang, et al.(2006) [12] came up with an approach to overcome this difficulty. In the same year G. Chen, et al. [13] used association rule mining for solving classification problems. It gave satisfactory results when compared to existing classification algorithms like C4.5, CBA, SVM, NN. Modification in traditional algorithm(apriori) was done [14] for building book recommendation system based on the data obtained from historical data of university library. Association rules having low support and high confidence are exception rules. In 2008 D. Taniar, et al. [15] proposed a new approach to finding

exception rules. First, generate candidate exception rules and based on exceptionality measure obtain final ruleset. The quality of association rules depends on the threshold value of support and confidence. R.J. Kuo, et al.(2011) [16] proposed an approach to find best threshold values which can produce quality rules. It gave promising results when compared to the genetic algorithm. Cloud computing provides an efficient and cheap way to store and analyze data. In 2011 L. Li, et al. [17] proposed an effective strategy to perform association rule mining(frequent itemset mining) in cloud computing environment. Apriori [2] generates candidate frequent itemsets before generating the desired itemsets. The modified algorithm(2012) [18] minimizes the candidate itemsets generation. In 2013 Tseng, et al. [19] proposed an approach(Hierarchical Partitioning) to perform frequent itemset mining in big databases. It uses Frequent Pattern List(FPL) data structure. FPL partitions the transaction database into a set of sub-databases before performing frequent itemset mining. With the passage of time Association Rule Mining find its role in many applications. J. Nahar, et al.(2013) [20] proposed a way to detect factors that can contribute to heart diseases in males and females. High Utility Itemset mining is used to find all the itemsets that generate at least threshold profits. It takes into account quantity of items and benefits. It is useful in many applications such as stock market prediction and retail market data analysis. Before mining high utility itemset it is required to set the threshold value of profits(min\_util). It's difficult for users to set an appropriate value of min\_util. Low value leads to too many itemsets. On the other side, high value results into low or no itemsets at all. Cheng, et al. [21] introduced a new framework top-k high utility itemset mining, where k is the number of these itemsets to be mined. They proposed an efficient algorithm named TKU (Top-K Utility itemsets mining) for mining high utility itemset without setting any threshold value. Before 2014 there were many algorithms available to perform high utility itemset mining. These algorithms generate a significant number of candidate itemsets that diminishes the mining performance. Yun, et al.(2014) [22] proposed an approach which prunes candidate itemsets efficiently. They also proposed a tree data structure

which captures entire transaction database in a single pass. Their approach outperforms the state of the art tree based algorithms concerning runtime. To enhance the quality of association rule mining researchers suggested to include factors such as value utility, temporal, etc. G. Maragatham, et al.(2015) [23] have proposed an efficient algorithm which combines both utility and temporal time periods for mining extraordinary association rules. Their results show that UTARM(Utility-Based Temporal Association Rule Mining) algorithm efficiently discovers the utility-oriented temporal association rules. In 2016 O. M. Soysal, et al. [24] proposed a data structure for sparse memory allocation for parallel and sequential data mining. They have used this data structure on apriori-TID, MASP-tree, and FP-growth algorithms to reduce memory allocation cost. They concluded that the speed-up in apriori-TID is more than FP-growth and the modified MASP algorithm becomes 3.42 times faster than the old implementation [1].

### 3. Method

Let  $I$  be a universal set of items. A single transaction( $\tau$ ) is a non-empty subset of universal itemset( $I$ ). Mathematically,  $\tau = \{item : item \in I\}$ . A transaction database( $\Gamma$ ) is a collection of such transactions. A rule of the form  $X \Rightarrow Y$  must satisfy  $support(X \Rightarrow Y) \geq \tau_s$  (threshold support) and  $confidence(X \Rightarrow Y) \geq \tau_c$  (threshold confidence) where  $X$  and  $Y$  are non-empty subsets of  $I$  and  $X \cap Y = \phi$ . What does  $support(X \Rightarrow Y)$  mean?  $support(X \Rightarrow Y)$  is defined as the probability of occurrence of  $X$  and  $Y$  together in the transaction database( $\Gamma$ ). Mathematically,  $support(X \Rightarrow Y) = \frac{Count(X \cup Y)}{Count(\phi)}$  where  $Count(Z)$  is the number of transactions in  $\Gamma$  which are superset of  $Z$ .  $confidence(X \Rightarrow Y)$  is the probability of occurrence of  $Y$  in those transactions of  $\Gamma$  which contains  $X$  or  $confidence(X \Rightarrow Y) = Probability(Y|X) = \frac{Count(X \cup Y)}{Count(X)}$ . Fix the value of threshold support( $0 \leq \tau_s \leq 1$ ) and threshold confidence( $0 \leq \tau_c \leq 1$ ) before performing association rule mining.

First, we will explain how to generate *OIMASP* tree before taking into

T1	1	2	3	4
T2	5	4	2	1
T3	2	1	4	5
T4	4	2	5	1
T5	1	5	2	4

Figure 2: Valid Transaction Dataset

T1	1	2	3	4
T2	5	4	2	1
T3	2	1	1	5
T4	4	2	5	1
T5	1	5	2	4

Figure 3: Invalid Transaction Dataset

account the origin of the item in  $\Gamma$ . Some of the terminologies that will be helpful in understanding the algorithm.

1. **Transaction( $\tau$ ):** It is a subset of universal itemset.
2. **Transaction Dataset( $\Gamma$ ):** A compilation of many transactions. Constraints imposed on transaction dataset are
  - a) Every row(transaction) in the transaction database must have the same number of items.
  - b) No duplicate items are allowed in rows of  $\Gamma$ .

Transaction database 2 is valid, and 3 is invalid(duplicate items(blue color) in  $T_3$ ).

3. **OIMASP:** A sequence of items  $I = \{I_1, I_2, I_3, \dots, I_n\}$  will be an **OIMASP** iff  $\forall j \in 1, 2, \dots, n$  the subset  $I' = \{I_1, I_2, \dots, I_j\}$  must satisfy
  - i)  $support(I') \geq \tau_s$
  - ii)  $P(I_j | I_1, I_2, \dots, I_{j-1}) \geq \tau_c$
  - iii)  $P(I_j | I_1, I_2, \dots, I_{j-1})$  is maximum.

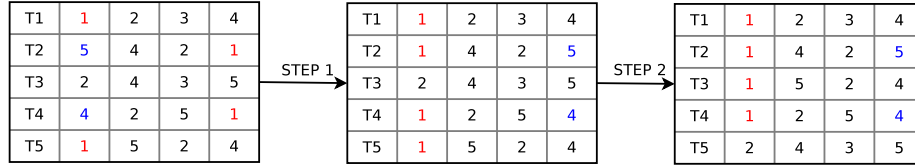


Figure 4: Shuffling of dataset as per item 1

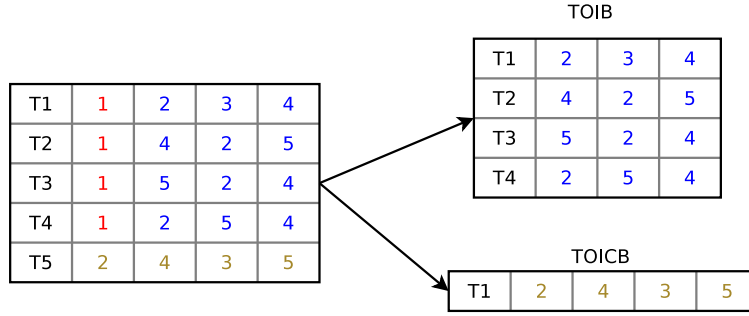


Figure 5: Splitting of  $\Gamma_{shuffled}$  into temporary order independent block and counter block

4. **Shuffle:** It is a function which takes transaction dataset, and an item as inputs and returns shuffled transaction dataset or  $\text{Shuffle}(\Gamma, I) \rightarrow \Gamma_{shuffled}$ . Shuffling is done in two steps
  - i)  $\forall$  rows, if the specified item( $I$ ) is present in the row then perform swapping to bring that item to the first column.
  - ii) Shuffle rows until there is no row left which contains item  $I$  and appears below row which do not have item  $I$ .
5. **Temporary Order Independent Block(TOIB):** After shuffling is done w.r.t the specified item, first obtain a subset of  $\Gamma_{shuffled}$  by taking transactions having specified item and then in the second step remove the first column. This newly received dataset is **TOIB**.
6. **Temporary Order Independent Counter Block(TOICB):** After shuffling is done w.r.t the specified item, the subset of  $\Gamma_{shuffled}$  obtained by taking transactions not having specified item is **TOICB**.
7. **Frequency Table:** A table which stores the frequency of items. In 6 the



frequency table corresponding to the transaction table is shown on the right.  $f_{max}$  is the maximum frequency in the frequency table.  $I_{max}$  is the item having frequency  $f_{max}$ . If multiple items have frequency  $f_{max}$ , then  $I_{max}$  can be any of them.

---

**Algorithm 1:** Algorithm to obtain the dataset associated with an *OIMASP* sequence

---

```

1 function FindDataset ( $\Gamma, OIMASP, j = 1$ );
   Input : A transaction dataset  $\Gamma$ , OIMASP sequence
   Output: Dataset associated with the OIMASP
2 if  $j > sizeof(OIMASP)$  then
3   | return  $\Gamma$ 
4 end
5 if negation is present on  $j$ th item of OIMASP then
6   |  $temp \leftarrow TOICB(\Gamma, OIMASP[j]);$ 
7   | return FindDataset( $temp, OIMASP, j + 1$ )
8 else
9   |  $temp \leftarrow TOIB(\Gamma, OIMASP[j]);$ 
10  | return FindDataset( $temp, OIMASP, j + 1$ )
11 end

```

---

### 3.1. How to generate *OIMASP* tree?

In this section, we have proposed an algorithm(*OIMASP*)[modified version of MASP [1]] which is independent of the ordering of items in transactions.

If we apply the *OIMASP* algorithm for the transaction dataset shown in 6, threshold support( $\tau_s$ ) = 0.2 and threshold confidence( $\tau_c$ ) = 0.3, we will get the final *OIMASP* tree in series of transitions as shown in 7.

### 3.2. How to generate association rules out of *OIMASP* tree?

**Definition 1.** Given  $OIMASP = \{I_1, I_2, I_3, \dots, I_k\}$ . Then  $\forall j \in \{2, 3, 4, \dots, k\}$   $(I_1, I_2, \dots, I_{j-1}) \Rightarrow (I_j)$  will be an association rule.

---

**Algorithm 2:** Algorithm to generate *OIMASP* tree.  $\Gamma$  is the input transaction database,  $|\Gamma|$  is the number of rows in  $\Gamma$ ,  $\tau_s$  is the threshold support, and  $\tau_c$  is the threshold confidence.

---

```

1 function OIMASP ( $\Gamma_{current}, Node$ );
   Input : A transaction dataset  $\Gamma_{current}$  associated with the Node of the
           OIMASP tree

   Output: An OIMASP tree

2 Obtain the frequency table of  $\Gamma_{current}$ 
3 Find the item  $I_{max}$  having maximum frequency  $f_{max}$ 
4 if ( $support = \frac{f_{max}}{|\Gamma|} < \tau_s$ ) then
5   | return
6 end
7 if ( $confidence = \frac{f_{max}}{|\Gamma_{current}|} < \tau_c$ ) then
8   | return
9 end
10 Add a node on the left side of Node say  $Node_{left}$  and store  $I_{max}$  in it
11 Add a node  $Node_{right}$  on the right side of Node and store  $\sim I_{max}$  in it
12  $\Gamma_{left} = TOIB(\Gamma_{current}, I_{max})$ 
13  $\Gamma_{right} = TOICB(\Gamma_{current}, I_{max})$ 
14 OIMASP( $\Gamma_{left}, Node_{left}$ )
15 OIMASP( $\Gamma_{right}, Node_{right}$ )
16 return

```

---

Transaction Table					
T1	1	12	3	4	5
T2	1	5	6	4	12
T3	8	6	9	12	5
T4	9	2	3	6	7
T5	6	9	10	8	7
T6	1	8	3	2	7

Frequency Table	
Item	Count
1	3
10	1
12	3
2	2
3	3
4	2
5	3
6	4
7	3
8	3
9	3

Figure 6: Transaction table on the left and items and their frequencies on the right

A path from the root to the leaf of the *OIMASP* tree will be an *OIMASP*. Therefore, there can be multiple *OIMASP*. An *OIMASP* and its corresponding association rules is shown in 8.

### 3.3. Modified version of *OIMASP*(*OOIMASP*) which takes into account origins of items in the transaction database

In this section, we will discuss the second modification to the *MASP* algorithm proposed in [1]. What if we want to generate those association rules which contains item  $I$ ? Our idea is to find the row starting from the top row of the transaction dataset in which item  $I$  appears for the first time(say  $i$ th row). Then for generating the *OIMASP* tree, we will consider  $i$ th transaction and transactions afterward. Then generate all association rules and add only those rules to the solution set which contains item  $I$ . It can be done for every unique element of  $\Gamma$  and finally take the union of solution sets obtained for each unique items to get the global solution set of association rules.

We will apply this algorithm for transaction dataset in (6), threshold support( $\tau_s$ ) = 0.2, threshold confidence( $\tau_c$ ) = 0.3 and item = 10.

Step 1. A subdataset(consists of 5th and 6th row as shown in 9) is obtained based on the origin of item 10.

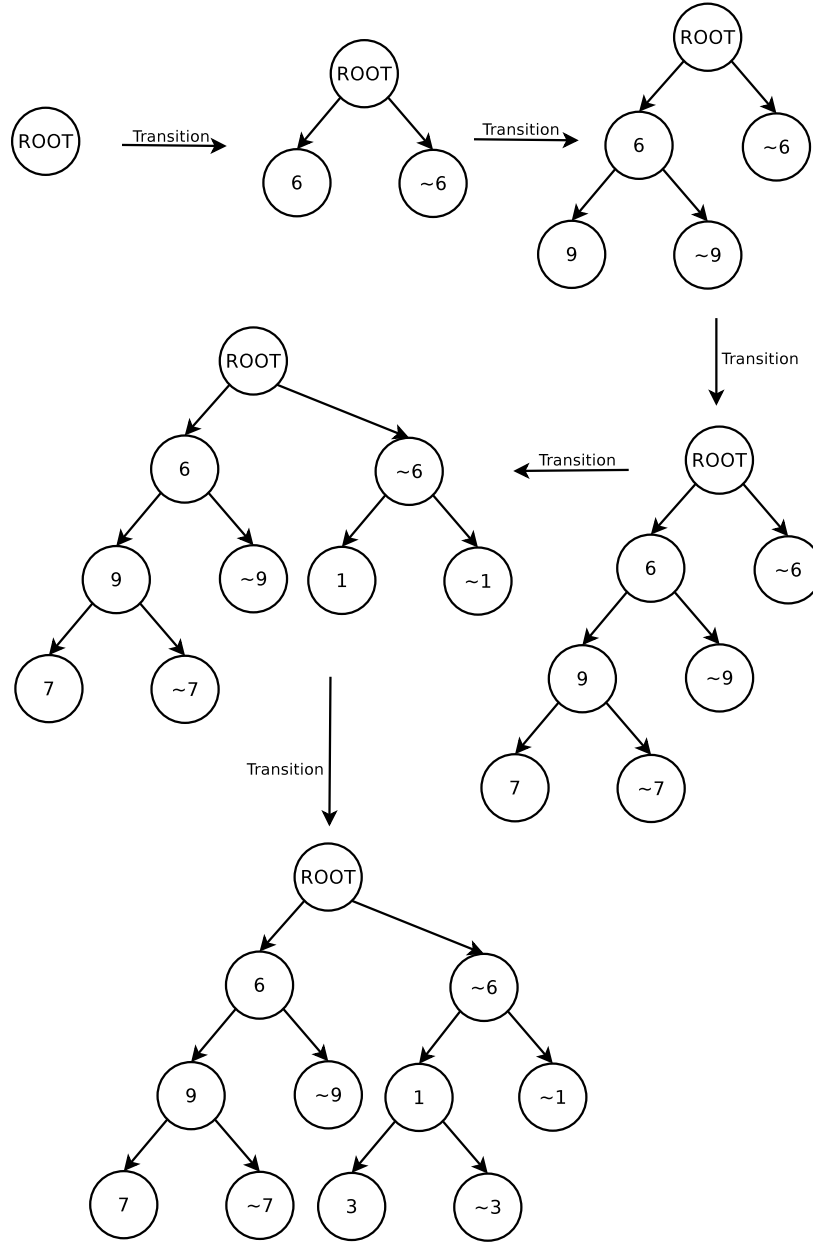


Figure 7: Generation of OIMASP tree as the algorithm proceeds

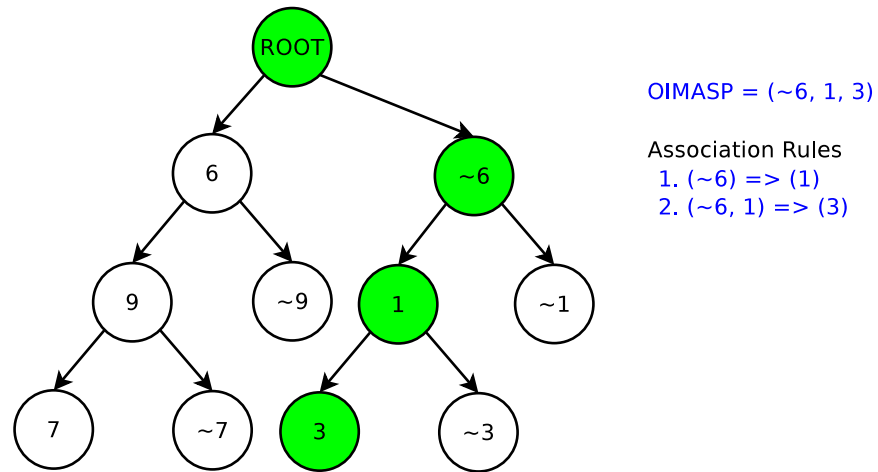


Figure 8: An OIMASP(in green) and corresponding association rules

Transaction Table					
T1	1	12	3	4	5
T2	1	5	6	4	12
T3	8	6	9	12	5
T4	9	2	3	6	7
T5	6	9	10	8	7
T6	1	8	3	2	7

← Item 10 appears for the first time in this row

Figure 9: Partition of dataset based on the origin of item 10

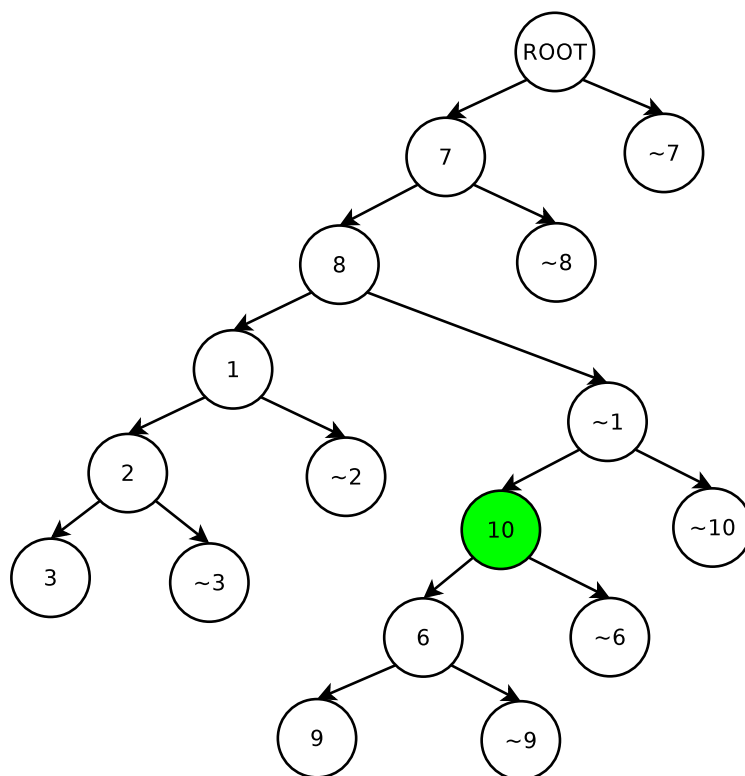


Figure 10: An OIMASP tree

$$\begin{array}{ll}
(7, 8, \sim 1) & \Rightarrow (10) \\
(7, 8, \sim 1, 10) & \Rightarrow (\sim 6) \\
(7, 8, \sim 1, 10) & \Rightarrow (6) \\
(7, 8, \sim 1, 10, 6) & \Rightarrow (\sim 9) \\
(7, 8, \sim 1, 10, 6) & \Rightarrow (9)
\end{array}$$

Figure 11: Association rules containing item 10

Step 2. Apply *OIMASP* tree generation algorithm on the new dataset to obtain 10.

Step 3. Find association rules which contains item 10(see 11). Add rules obtained to the global solution set.

Step 4. Repeat these steps for other items too.

**Discussion:** If we generate *OIMASP* tree for all the items in the transaction database, then *OIMASP* tree generation algorithm will be called multiple numbers of times(=number of unique items in the dataset). It is possible that multiple items may have the same origin hence no need for redundancy(same *OIMASP* tree generation multiple times). Avoiding the redundancy can led to fewer *OIMASP* tree production. In the worst case  $N$ (total number of transactions) *OIMASP* tree will be generated.

#### 3.4. Complexity of *OOIMASP* algorithm

The complexity of the *MASP* algorithm [1] is  $\Theta(NM \log(NM))$ . After the first improvement(*OIMASP*) complexity remains the same. In the concluding algorithm(*OOIMASP*) *OIMASP* can be called at max  $N$  times so that the complexity will be  $\sum_{i=1}^N \Theta(iM \log(iM)) = \Theta(N^2 M \log(NM))$ .

## 4. Experiments and results

The old approach and the new approach are compared using five datasets namely A, B, C, D, and E. Metrics used for comparison are maximum rule size and the total number of rules generated. As you can see in 12 that the new approach outperforms the old approach concerning number of rules generated

---

**Algorithm 3:** OOIMASP Algorithm

---

```
1 function OOIMASP ( $\Gamma, \tau_s, \tau_c$ );  
   Input  : A transaction dataset  $\Gamma[N][M]$ , threshold support  $\tau_s$  and  
           threshold confidence  $\tau_c$   
   Output: association rules  
2  $items[1 : j] \leftarrow$  unique items list  
3  $origins[1 : j] \leftarrow$  origin of corresponding items in  $items$   
4  $globalRules \leftarrow \{\}$   
5 for  $i$  in  $1 : j$  do  
6    $currentItem \leftarrow items[i]$   
7    $itemOrigin \leftarrow origins[i]$   
8   if OIMASP tree is not yet generated for origin itemOrigin then  
9     generate OIMASP tree for dataset =  $\Gamma[itemOrigin : N]$  and  
       given  $\tau_s, \tau_c$   
10  end  
11  generate all association rules from OIMASP tree having origin  
    =  $itemOrigin$ , which contains item  $currentItem$  and add these to  
     $globalRules$   
12 end  
13 return  $globalRules$ ;
```

---



Dataset	Min-Support	Min-Confidence	MASP		OOIMASP	
			total rules	max-rule-size	total rules	max-rule-size
A	0.60	0.30	2	2	7	4
B	0.10	0.05	65	8	289	19
C	0.10	0.05	9	4	108	29
D	0.03	0.001	275	23	333	94
E	0.30	0.01	No Rules	No Rules	524	132

Figure 12: Comparison of *MASP* and *OOIMASP*

and longest rule size. The amount of information obtained from the dataset has suppressed the increase in time complexity.

## References

- [1] O. M. Soysal, Association rule mining with mostly associated sequential patterns, *Expert Systems with Applications* 42 (5) (2015) 2582 – 2592. doi:<http://dx.doi.org/10.1016/j.eswa.2014.10.049>.
- [2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules (1994) 487–499.
- [3] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, *SIGMOD Rec.* 22 (2) (1993) 207–216. doi:[10.1145/170036.170072](https://doi.org/10.1145/170036.170072).  
URL <http://doi.acm.org/10.1145/170036.170072>
- [4] M. Houtsma, A. Swami, Set-oriented mining of association rules, Research Report RJ 9567, IBM Almaden Research Center, San Jose, California.
- [5] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama, Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization, *SIGMOD Rec.* 25 (2) (1996) 13–23. doi:[10.1145/235968.233313](https://doi.org/10.1145/235968.233313).  
URL <http://doi.acm.org/10.1145/235968.233313>

- [6] R. Feldman, Y. Aumann, A. Amir, A. Zilberstein, W. Kloeegen, Maximal association rules: a new tool for mining for keyword cooccurrences in document collections.
- [7] C. H. Cai, A. W. C. Fu, C. H. Cheng, W. W. Kwong, Mining association rules with weighted items, in: Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International, 1998, pp. 68–77. doi:10.1109/IDEAS.1998.694360.
- [8] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, Data Mining and Knowledge Discovery 8 (1) (2004) 53–87. doi:10.1023/B:DAMI.0000005258.31418.83.  
URL <http://dx.doi.org/10.1023/B:DAMI.0000005258.31418.83>
- [9] W. Lin, S. A. Alvarez, C. Ruiz, Efficient adaptive-support association rule mining for recommender systems, Data Mining and Knowledge Discovery 6 (1) (2002) 83–105. doi:10.1023/A:1013284820704.  
URL <http://dx.doi.org/10.1023/A:1013284820704>
- [10] F. Coenen, P. Leng, S. Ahmed, Data structure for association rule mining: T-trees and p-trees, IEEE Transactions on Knowledge and Data Engineering 16 (6) (2004) 774–778. doi:10.1109/TKDE.2004.8.
- [11] P. D. Shenoy, K. G. Srinivasa, K. R. Venugopal, L. M. Patnaik, Dynamic association rule mining using genetic algorithms, Intell. Data Anal. 9 (5) (2005) 439–453.  
URL <http://dl.acm.org/citation.cfm?id=1239098.1239101>
- [12] N. Jiang, L. Gruenwald, Research issues in data stream association rule mining, SIGMOD Rec. 35 (1) (2006) 14–19. doi:10.1145/1121995.1121998.  
URL <http://doi.acm.org/10.1145/1121995.1121998>

- [13] G. Chen, H. Liu, L. Yu, Q. Wei, X. Zhang, A new approach to classification based on association rule mining, *Decision Support Systems* 42 (2) (2006) 674 – 689. doi:<http://dx.doi.org/10.1016/j.dss.2005.03.005>.
- [14] Z. Zhu, J. Y. Wang, Book recommendation service by improved association rule mining algorithm, in: 2007 International Conference on Machine Learning and Cybernetics, Vol. 7, 2007, pp. 3864–3869. doi:[10.1109/ICMLC.2007.4370820](http://dx.doi.org/10.1109/ICMLC.2007.4370820).
- [15] D. Taniar, W. Rahayu, V. Lee, O. Daly, Exception rules in association rule mining, *Applied Mathematics and Computation* 205 (2) (2008) 735 – 750. doi:<http://dx.doi.org/10.1016/j.amc.2008.05.020>.
- [16] R. Kuo, C. Chao, Y. Chiu, Application of particle swarm optimization to association rule mining, *Applied Soft Computing* 11 (1) (2011) 326 – 336. doi:<http://dx.doi.org/10.1016/j.asoc.2009.11.023>.
- [17] L. Li, M. Zhang, The strategy of mining association rule based on cloud computing, in: 2011 International Conference on Business Computing and Global Informatization, 2011, pp. 475–478. doi:[10.1109/BCGIN.2011.125](http://dx.doi.org/10.1109/BCGIN.2011.125).
- [18] S. A. Abaya, Association rule mining based on apriori algorithm in minimizing candidate generation.
- [19] F.-C. Tseng, Mining frequent itemsets in large databases: The hierarchical partitioning approach, *Expert Syst. Appl.* 40 (5) (2013) 1654–1661. doi:[10.1016/j.eswa.2012.09.005](http://dx.doi.org/10.1016/j.eswa.2012.09.005).  
URL <http://dx.doi.org/10.1016/j.eswa.2012.09.005>
- [20] J. Nahar, T. Imam, K. S. Tickle, Y.-P. P. Chen, Association rule mining to detect factors which contribute to heart disease in males and females, *Expert Systems with Applications* 40 (4) (2013) 1086 – 1093. doi:<http://dx.doi.org/10.1016/j.eswa.2012.08.028>.

- [21] C. W. Wu, B.-E. Shie, V. S. Tseng, P. S. Yu, Mining top-k high utility itemsets, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, ACM, New York, NY, USA, 2012, pp. 78–86. doi:10.1145/2339530.2339546.  
URL <http://doi.acm.org/10.1145/2339530.2339546>
- [22] U. Yun, H. Ryang, K. H. Ryu, High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates, Expert Syst. Appl. 41 (8) (2014) 3861–3878. doi:10.1016/j.eswa.2013.11.038.  
URL <http://dx.doi.org/10.1016/j.eswa.2013.11.038>
- [23] G. Maragatham, M. Lakshmi, Utarm: An efficient algorithm for mining of utility-oriented temporal association rules, Int. J. Knowl. Eng. Data Min. 3 (2) (2015) 208–237. doi:10.1504/IJKEDM.2015.071293.  
URL <http://dx.doi.org/10.1504/IJKEDM.2015.071293>
- [24] O. M. Soysal, E. Gupta, H. Donepudi, A sparse memory allocation data structure for sequential and parallel association rule mining, J. Supercomput. 72 (2) (2016) 347–370. doi:10.1007/s11227-015-1566-x.  
URL <http://dx.doi.org/10.1007/s11227-015-1566-x>