ARM-AMO: An Efficient Association Rule Mining Algorithm Based on Animal Migration Optimization

Le Hoang Son [a, g, *], Francisco Chiclana [b], Raghavendra Kumar [c], Mamta Mittal [d], Manju Khari [e], Jyotir Moy Chatterjee [f], Sung Wook Baik [g]

[a] *VNU University of Science, Vietnam National University, Vietnam*
sonlh@vnu.edu.vn
[b] *School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester, LE1 9BH UK*
chiclana@dmu.ac.uk
[c] *Computer Science and Engineering Department, LNCT College, MP, India*
raghvendraagrawal7@gmail.com
[d] *Department of Computer Science & Engineering, G.B. Pant Govt. Engineering College, Delhi, India*
mittalmamta79@gmail.com
[e] *Department of Computer Science & Engineering, AIACT&R, Delhi, India*
manjukhari@yahoo.co.in
[f] *Department of Computer Science and Engineering, GD-RCET, Bhilai, CG, India*
jyotirm4@gmail.com
[g] *College of Electronics and Information Engineering, Sejong University, Seoul, Republic of Korea*
sbaik@sejong.ac.kr
**: Corresponding author. Tel.: (+84) 904.171.284. Address: 334 Nguyen Trai, Thanh Xuan, Hanoi, Vietnam*

Abstract: Association rule mining (ARM) aims to find out association rules that satisfy predefined minimum support and confidence from a given database. However, in many cases ARM generates extremely large number of association rules, which are impossible for end users to comprehend or validate, thereby limiting the usefulness of data mining results. In this paper, we propose a new mining algorithm based on Animal Migration Optimization (AMO), called ARM-AMO, to reduce the number of association rules. It is based on the idea that rules which are not of high support and unnecessary are deleted from the data. Firstly, Apriori algorithm is applied to generate frequent itemsets and association rules. Then, AMO is used to reduce the number of association rules with a new fitness function that incorporates frequent rules. It is observed from the experiments that, in comparison with the other relevant techniques, ARM-AMO greatly reduces the computational time for frequent item set generation, memory for association rule generation, and the number of rules generated.

Keywords: Association rules mining; Animal Migration Optimization (AMO); Apriori algorithm; Particle Swarm Optimization (PSO).

1. INTRODUCTION

Data mining emerges as a mean for identifying patterns and trends from large quantities of data (Jerry *et al.*, 2016). It encompasses various algorithms such as clustering, classification, association rule mining, and sequence detection (Thabtah, Qabajeh & Chiclana, 2016). Among all, **association rule mining** (ARM) aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in transaction databases and other data repositories (Ratner, 2017). Association rules are widely used in various areas such as telecommunication networks, market and risk management and inventory control (Martínez-Ballesteros, Bacardit, Troncoso & Riquelme, 2015). The main aim of ARM is to find out rules that satisfy predefined minimum support and confidence from a given database (Mai, Vo & Nguyen, 2017). This task is usually decomposed into two sub problems. Problem one is to find those item sets whose occurrences exceed a predefined threshold in the database such as candidate large item sets and frequent item sets generation processes (Nithya & Duraiswamy, 2015). Problem two is to generate association rules from large item sets with constraints of minimal confidence (Nguyen *et al.*, 2012; Rauch, 2015; Song, Ding, Chen, Li, Cao & Pu, 2016).

In many cases, **ARM generates extremely large number of association rules**, which are impossible for end users to comprehend or validate, thereby limiting the usefulness of data mining results (Mlakar, Zorman, Fister & Fister, 2017). Numerous algorithms have been proposed to reduce the number of association rules (Chen, Shen, Chen, Shang & Wang, 2011; Shirsath & Verma, 2013; Mlakar, Zorman, Fister & Fister, 2017), such as generating only interesting rules or non-redundant rules, or rules satisfying certain criteria such as coverage, leverage, lift or strength (Yan, Sun & Liu, 2016). One of the most effective strategies for this problem is integrating optimization techniques with association rule mining for increasing its performance. Among the proposed algorithms are ARM-PSO (Kuo, Chao & Chiu, 2011), GA (Oladele & Sadiku, 2013), Cuckoo Search (Yun, Kim, Ryang, Lee & Lee, 2016; Mlakar *et al.*,

2017), WFIM (Wensheng *et al.*, 2017), HUIM-MMU (Jerry *et al.*, 2016), CP-Miner (Thanh-Long, Bay & Vaclav, 2017), dynamic superset bit-vector (Tahrima *et al.*, 2017) and lattice (Thang, Bay & Loan, 2017). However, most of the relevant algorithms for controlling large number of association rules are often computationally expensive and possibly generate much irrelevant rules.

To overcome these problems, this paper **proposes a new association rule mining algorithm based on Animal Migration Optimization** (AMO). AMO is one of the most typical optimization algorithms based on the behavior of animal migration. In the proposed method, rules which are not of high support and unnecessary are deleted from the data. Only frequent rules are kept and displayed. All these criteria are incorporated into the fitness function of the AMO for better generation of rules. ARM-AMO significantly improves ARM-PSO in solving complex swarm optimization problems in terms of number of rules, time and memory consumption by adopting the new algorithm.

The cohesion and structure of the article is demonstrated as follows: Section 1 introduces the problem of reducing the number of association rules existed in the current ARM algorithms and highlights the idea of the new algorithm called ARM-AMO, which is based on Animal Migration Optimization for reducing rules that are not of high support and unnecessary from the data. In the Sections 2 and 3, we give an overview of the related works and background of ARM. In Section 4, we present details of new algorithm starting from the idea (Section 4.1), pseudo-code (Section 4.2), an illustrative example (Section 4.3), and the theoretical comparison between the new and previous algorithms (Section 4.4). Section 5 explains experimental environment (Section 5.1) and comparative results by various cases (Section 5.2) consisting of both tables and figures accompanied by discussion. Finally, Section 6 draws conclusions and further works, respectively.

2. RELATED WORKS

Finding frequent rules with the help of association rule mining faces many issues such as irrelevant rules and computational time, which can degrade performance (Chen *et al.*, 2011; Rauch, 2015; Nithya & Duraiswamy, 2015). Some efforts have been made to overcome these issues with number of rules generation, time and fitness function Weighted Frequent Item set Mining (WFIM) algorithm proposed by Slim *et al.* (2014), Jerry *et al.* (2016) and Wensheng *et al.* (2017), and further extension by Das *et al.*(2012), considering not only the frequency of items but also their relative importance. Lin et al. (2015) proposed a rundown based FFI-Miner method (Fast Frequent Itemset) to find rules from quantitative databases. Yan *et al.* (2016) proposed FARMA which is an extension of the Apriori algorithm by reducing time complexity and space complexity. Again, Wang *et al.* (2016) enhanced prediction accuracy by employing quantitative association base on the improved Apriori. Thanh-Long *et al.* (2017) proposed an algorithm for mining colossal patterns for reducing candidates. Tahrima *et al.* (2017) introduced a new memory efficient data structure called the dynamic superset bit-vector to establish the relationship among frequent closed item sets in a lattice, while Thang *et al.* (2017) proposed an algorithm for mining high utility association rules using a lattice.

Besides, association rule mining was also integrated with optimization techniques for upgrading its performance. Kuo *et al.* (2011) proposed *Particle swarm optimization (PSO) based ARM algorithm* (ARM-PSO) in an application of stock market to gauge speculation conduct and stock class buying. Oladele and Sadiku (2013) proposed a hybrid Genetic Algorithm for multi-target outline tricky abusing divergent parent decision. Martínez-Ballesteros *et al.* (2015) enhanced the versatility of quantitative association rule mining systems in light of genetic calculations. Yun *et al.* (2016) suggested a modified single-objective binary cuckoo search for association rule mining (MBCS-ARM) including a novel representation of individuals, which tackles the problems of large dimensionality with an increasing number of attributes. It also

supports the mining of rules, where intervals of attributes can either be negative or positive. Song *et al.* (2016) proposed a multi-objective binary at algorithm (MBBA) based on Pareto for association rule mining. Mlakar *et al.* (2017) presented a single-objective binary cuckoo search using a novel individual representation. Other works can be found in (Pears & Koh, 2011; Song & Lee, 2017; Anuradha & Kumar, 2017; Feng *et al., 2016*; Huang *et al.*, 2017; Vo, Pham, Le & Deng, 2017; Kieu, Vo, Le, Deng & Le, 2017).

3. BACKGROUND

3.1. Association Rule Mining

Association Rule Mining (ARM) is a process to determine accessible association rules for regularities amongst items in large-scale interchange info recorded (Yun *et al.*, 2016). Let $I=I_1$, I_2, ...I_m be a set of m targeted attributes and T be a transaction that contains a group of objects such that $T \rightarrow I$. D is a database with exclusive transaction files. An association rule is an implication of type $X \rightarrow Y$ where X and Y are attributes and $X \cap Y = \emptyset$. X is known as the antecedent event and Y is known as the consequent. Two important criteria for association rule mining are *support* (S) and *confidence* (C), which indicates how frequently items are in the database and how many times the item sets are presented, respectively. The following includes some important definitions in ARM (Shirsath & Verma, 2013; Barati *et al.*, 2017).

Definition 1. Given a collection of n transactions $T= \{t_1, \ldots t_n\}$ and m items $I = \{i_1, \ldots i_m\}$, an **association rule** is expressed in the form:

$$X(Antecedent) \rightarrow Y(Consequent), \quad (1)$$

where $X, Y \subseteq I, X \cap Y = \Phi$, the left-hand and right-side rules are the antecedents and the consequents, respectively.

Definition 2.*Support(X)* describes the proportion of transactions in T including X.

$$Support\ (X) = \frac{Number\ of\ Transaction\ Containing\ X}{Total\ Number\ of\ Transactions}, \qquad X \in \mathrm{T} \tag{2}$$

Definition 3.If *Support(S)* ≥ *Min_Support* then *S* is known as **frequent item set** where *Min_Support* is a threshold value described by users.

Definition 4. Transactions Count is $N=|T|$.

Definition 5. Largest transaction length is $E=Max\ (|t_i|)$.

Definition 6. The **rule confidence** is the proportion of transactions in *T* including item set *X* which also include item set Y. Rules with both *Support(X→Y)* ≥ *Min_Support* and *Confidence(X→Y)* ≥ *Min_Confidence* are called strong rules. These thresholds values are described through customers.

$$Confidence\ (X \rightarrow Y) \quad = \frac{Support\ (X \cup Y)}{|Support\ (X)|}. \tag{3}$$

3.2. Animal Migration Optimization

The key scheme of AMO is implemented by means of concentric zones around each animal, which means that each animal looks for keeping safe itself from its neighbor to avoid collision. AMO is divided into two parts: animal migration process and animal updating process. During the migration process, an animal should follow three rules: avoid collisions with your neighbors, move in a similar direction as your neighbors, and stay close to the neighbors.

The idea of restricted neighborhood of an individual is described through the topological ring, which is stationary and described on the indices of vectors. The suggestion of neighboring area is described by topological circle (Li *et al.*, 2014). If an animal index is *i*, then its nearest neighbor has index of $i-2, i-1, i, i+1, i+2$. Once the topology of nearest neighbor is built, the nearest is determined as follows (Badhe *et al.*, 2015):

$$(X_{new}) = (X_{old}) + \delta\left(X_{neighborhood} - X_{old}\right), \quad (4)$$

where $X_{neighborhood}$ is the neighborhood's current position, X_{old} is the current position of individual, X_{new} is the new position of individual, and δ is a random number generator controlled through a Gaussian distribution. After producing new results, an objective fitness is computed (Li *et al.*, 2014).

4. THE PROPOSED ARM-AMO

4.1. Description

The **basic idea** of ARM-AMO is to derive rules based on Animal Migration Optimization in which those which are not of high support and are not necessary will be deleted from the data. Only frequent rules are kept and displayed. Figure 1 indicates a broad view about the proposed algorithm in which the first step finds rules by Apriori algorithm (Lin *et al.*, 2015; Nguyen *et al.*, 2012). It generates candidate item set and frequent item set by joining and pruning. In the second and third steps, the support and confidence for the rules and the fitness function for the animal migration are calculated. The fitness function of rules and velocity of particles are then updated iteratively until the global optimum solution is reached. In the fifth step, the rules having high fitness value are removed.

After this, the remaining rules are optimized. It comes from the fact that evaluating the rules which are below the fitness value is significant as they are less fit rules and need to be migrated. Animal migration is applied to the rules having small fitness values by calculating their migrating probability. For every instance, probability is updated and next position for the movement is evaluated. In this method, those rules which are less fit will be initially moved to a better place. This increases their survival probability and thus better rules can be mined (Mai, Vo & Nguyen, 2017).

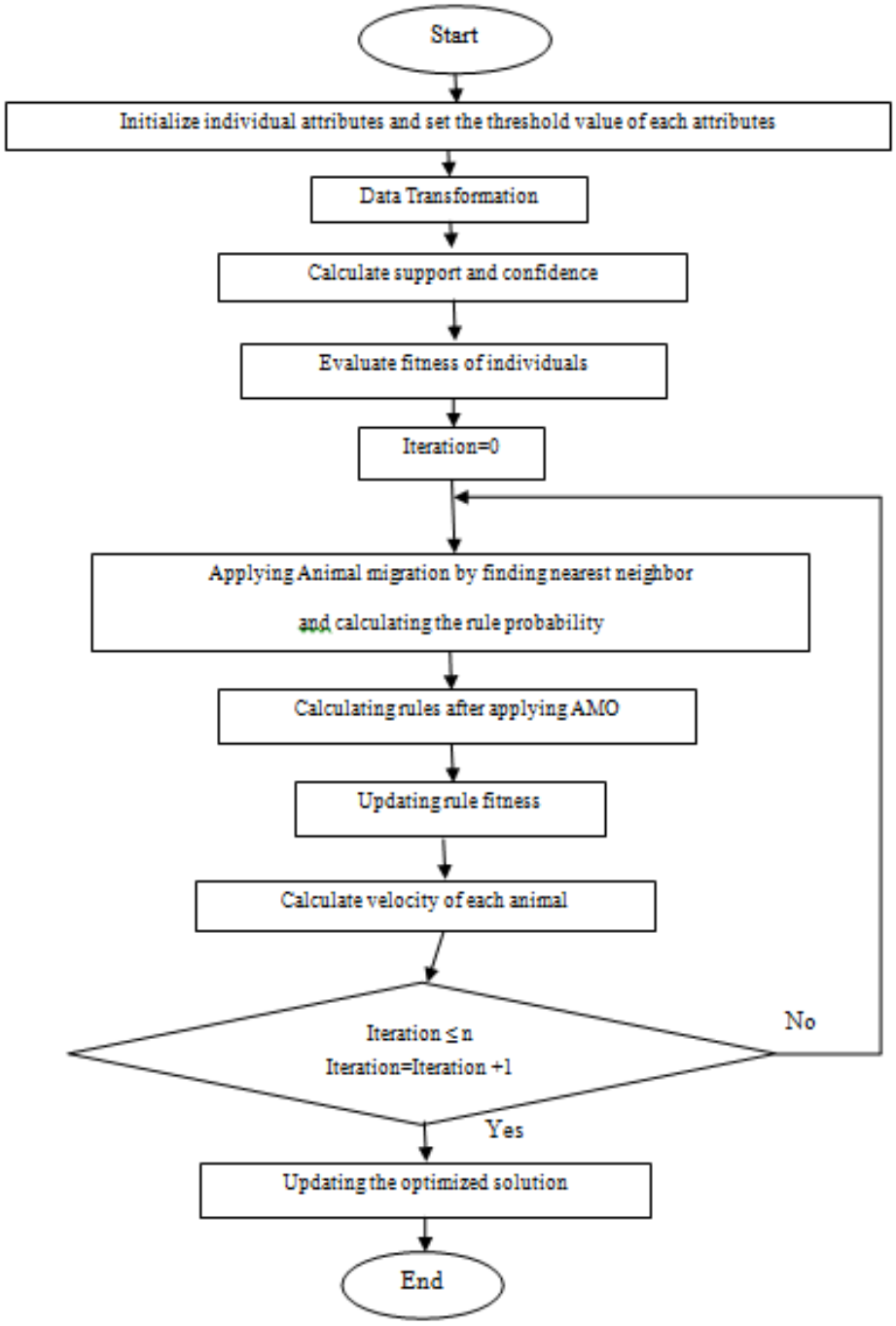

Fig. 1. Flow chart of the proposed algorithm

Specifically, ARM-AMO comprises of two parts: *rules calculation* and *rule optimization*. In the **first part**, data are transformed to binary values for storage purposes. This in turn hastens the database scanning operation and results in quick calculation of the support value. Let us consider, as shown in in Figure 2, that there are five customers, namely, T1 to T5 in the

transactional database. All the transactions are transformed into binary form and stored. In the above example, as there are four different products, hence four columns will exist. Consider T3, because he/ she has purchased the products P1, P2, P3 and P4. Therefore, for B3, the rows under P1, P2, P3 and P4 will have the value "1".Consider another example T1, because he / she has purchased the products P1, P2 and P4. Therefore, for B1, the rows under P1, P2 and P4 will have the value "1" and P3 have the value "0".

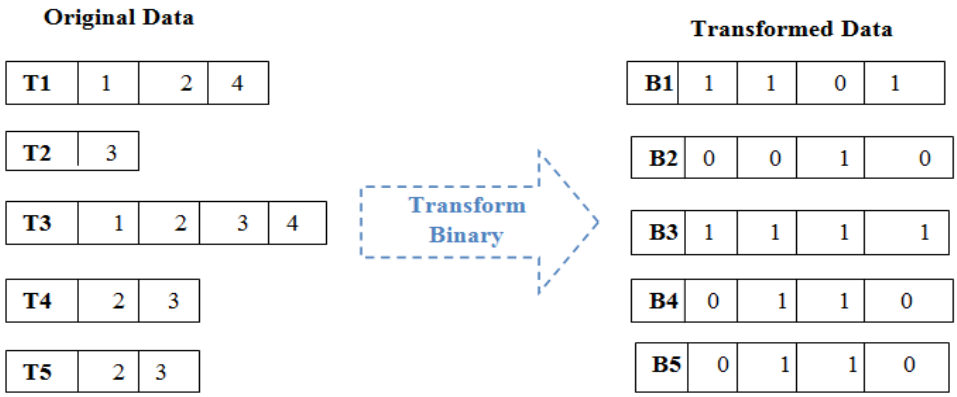

Fig. 2. Data types Transformation

After that, the *fitness value is computed* based on the confidence and support derived from the Apriori algorithm. Also, a net fitness value is calculated for the overall fitness. The fitness analysis is exploited to decide rules that are to be modified using AMO. Less suitable rules are found by comparing their fitness value with net fitness. If it is less than net fitness, they are weak rules.

ARM-AMO takes the result obtained from the Apriori algorithm as input to generate the *optimized association rules*, which are helpful to analyze the products frequently purchased by customer and to discover customer shopping patterns and to find interesting rules with derived shopping patterns. This helps organizations, i.e. supermarkets, to increase sales growth by getting customers data and obtaining customer behavioral pattern for developing new business

strategies. The optimized rule generation process in ARM-AMO is shown as follows. In ARM-AMO, fitness value is employed to **estimate the importance of each rule**. The fitness value of any rule depends on the support and confidence which are mathematically formulated below (Indira & Kanmani, 2012).

$$Fit(x) = Con(x) \times \log\left(Sup(x) \times Length(x) + 1\right). \quad (5)$$

In equation (5), *Fit(x)* is the fitness value of rule type *x*, *Sup(x)* and *Con(x)*are discussed in equations(2) and (3), and *Length(x)* is the length of rule type *x*. ARM-AMO maximizes the fitness value function since large support and confidence values result in great association, which represents significant rules. The particle in the population of AMO that has the highest fitness value is selected as gbest, and its support and confidence are employed as the minimum thresholds.

In ARM-AMO, each particle characterizes a rule and each rule includes of a series of decision variables which signify the status of every item in the rule. Each particle in ARM-AMO has a 'position' and 'velocity', where position is represented as a solution suggested by the particle and velocity is the rate of changes to the next position with respect to current position. The position and velocity are arbitrarily initialized in ARM-AMO, thus containing a collection of random particles i.e. rules. During each iteration, all particles are updated using pbest and gbest values. Herein, pbest represents the best solution it has achieved so far. Afterward, the particle updates its velocity as follows:

$$Vnew(i,d) = Pold\ (i,d) + FF \times P(p(b),d) - FF \times P(g(b),d). \quad (6)$$

For solving the AMO problems, De-Jong's function is used (Randy & Haupt, 2003):

$$Unew\ (i,d) = \frac{Vnew\ (i,d)}{\sqrt{\sum_{k=1}^{n}\left(\left(Vnew\ (i,d)\right)ik\right)^2}}. \quad (7)$$

The position of a particle is updated at each iteration as follows:

$$Pnew = Pold + Unew(i, d) + FF. \quad (8)$$

where *FF* is the fitness function within (0, 1), *P* is the particle position, *d* is the current particle, *p(b)* is the best value of a particle, *g(b)* is the global best value, and *Pold(i, d)* is the velocity of particle i^{th}. By the above strategies, rules are optimized in an efficient way.

4.2. Pseudo code

The pseudo code of ARM-AMO for optimizing association rules is explained in Table 1.

Table 1. Pseudo code of ARM-AMO

ARM-AMO	
Input	*X,Y-Attributes, T-Total Number of Transactions, i-Current Rule, D- Dimensions (Lower, Upper, Middle), d-last rule, Minimum* $_{Support}$*, Minimum* $_{Confidence}$*, Length (dataset), Pold(i, d)-*i^{th} *particle velocity, n-Number of iteration, OD- Original data*, BD- Binary Data
Output	*The best solution g(b) // Global best*
1	$Begin$ $t \leftarrow 0$ $BD \leftarrow OD$ **Calculating support and confidence** $Support\ (X \cup Y) = \frac{Probability\ (X)}{T}$ $Confidence\ (X \cup Y) = \frac{Support(X \cup Y)}{Support\ (X)}$
2	**Calculating fitness for Animal Migration** $i \leftarrow 0\ to\ n$ $Overall\_fiteness\ (i) = \frac{Log(Confidence\ (i) + Support\ (i))}{\log(Confidence\ i) + Log\ (supporti)}$ $Net_{fitnees} = \frac{Sum\ (Overall\ Fitness)}{Length\ (Dataset)}$ $\text{Net}_{\text{fitness}} = Netfitness \times \frac{Support\ (i)}{Minimum_{Support} \times Mimimum_{Confidence} \times Length\ (Dataset)}$
3	**Evaluate the overall fitness and fitness value of each records by fitness function**

	$Fitness_{Function} = Benchmark_{Function}(X, Problem)$ $Fitness_{value} = Sum\ (2X^2)$ $Pop\ Size = Length\ (Dataset)$ $D = n;$ $L = [1 \times Lb\ (1, D); 1000 \times Ub\ (1, D)]$ // Define lower and upper bounds $P = (X, Y)$// Population initialization $Global\ Minimum\ Support = \sum_{i=1}^{n} \text{Support i}$ $if\ (Fitness_{value} = Global\_minimum\_support\ )$ $for\ (i = 1; i \leq n; i = Popsize + 1)$// Popsize=Population Size $Lseq = (Popsize - 1, Popsize, i, i + 1, i + 2)$ // Searching for neighbors $Else$ $Lseq = 0;$ $Popsize = Popsize + 1;$ $End;$
4	**Updating the velocity of animal** $FF = Fitness_{Funcion}\ range\ (0, 1)$ $Vnew\ (i, d) = Pold\ (i, d) + FF \times P(p(b), d) - FF \times P(g(b), d)$ $Unew\ (i, d) = \frac{Vnew\ (i,d)}{\sqrt{\sum_{k=1}^{n}\left((Vnew\ (i,d))ik\right)^2}}$// DeJong function is used for solving the AMO problem $Pnew = Pold + Unew(i, d) + FF$// New particle position
5	$if\ \left(f(Pnew) < P(b)\right)$ $then$ $P(b) = Pnew$ // Update the best local position End $g(b + 1) = Min\ \left(P(b), g(b)\right)$ // Update the global best position $t \leftarrow t + 1$

As shown in Table 1, ARM-AMO initially takes association rules generated from Apriori as inputs. Then, it initializes particles with random positions and velocities where each particle represents a rule. After that, ARM-AMO computes fitness value for each particle to evaluate the importance of each rule. With the support of determined fitness value, the particle with highest fitness value is selected as gbest. The particles update their positions and velocities, at each iteration, and choose gbest. This process is continued until a maximal number of iterations or a minimum error criterion is attained.

4.3. An illustrative example

To illustrate the ARM-AMO algorithm, an example of supermarket analysis is shown below (Table 2).

Table 2. Supermarket dataset

TID	Item purchased
1	Laptop, Fan, Headphone, Keyboard, Mouse, Hard disk, Power bank, Charger, processor, battery, Wi-Fi
2	Laptop, Fan, Headphone, Mouse, Hard disk, Power bank, Charger, processor, battery, Wi-Fi
3	Laptop, Fan, Headphone, Keyboard, Hard disk, Power bank, Charger, processor, battery, Wi-Fi
4	Headphone, Keyboard, Mouse, Hard disk, Power bank, Charger, processor, battery, Wi-Fi
5	Laptop, Headphone, Keyboard, Mouse, Hard disk, Power bank, Charger, processor, battery, Wi-Fi
6	Laptop, Fan, Headphone, Keyboard, Mouse, Hard disk, Power bank, Charger, processor, battery
7	Laptop, Fan, Headphone, Keyboard, Mouse, Hard disk, Power bank, Charger, processor

After converting the original dataset into binary values, we have the results in Table 3.

Table 3. Binary dataset

TID	Item purchased
1	{1,1,1,1,1,1,1,1,1,1,1}
2	{1,1,1,0,1,1,1,1,1,1,1}
3	{1,1,1,1,0,1,1,1,1,1,1}
4	{0,0,1,1,1,1,1,1,1,1,1}
5	{1,0,1,1,1,1,1,1,1,1,1}
6	{1,1,1,1,1,1,1,1,1,1,0}
7	{1,1,1,1,1,1,1,1,1,0,0}

A. Measurement of Number of Rules Generated:

In ARM-AMO, the number of rules is generated based on the support and confidence values. When the number of rules is low, the method is said to be more efficient. In what follows, we present the number of rules created by ARM-AMO and the relevant algorithms namely HUIM-

MMU (Lin *et al.*, 2015), WFIM (Wensheng *et al.*, 2017), HUIL (Mai *et al.*, 2017), CP Tree (Thanh-Long *et al.*, 2017), AMO (Li *et al.*, 2014) and ARM-PSO (Kuo *et al.*, 2011).

Table 4 shows the number of rule of the algorithms by different support and confidence values. It is clear that the number of rules generated by the proposed ARM-AMO method is smaller than those generated by the above mentioned relevant algorithms.

Table 4. Number of rules generated (Bold values indicate the best among all in a row)

Support and Confidence	**HUIM-MMU**	**WFIM**	**HUIL**	**CP Tree**	**AMO**	**ARM-PSO**	**ARM-AMO**
0.1 and 0.1	3350	3231	3212	3210	4210	3521	**3120**
0.2 and 0.2	3241	3214	3210	3142	3952	3412	**2832**
0.3 and 0.3	3124	3124	3142	3125	3852	3321	**2752**
0.4 and 0.4	3020	3085	3085	3014	3720	3310	**2652**
0.5 and 0.5	2752	2785	2742	2785	3342	3104	**2541**
0.6 and 0.6	2140	2142	2135	2054	2952	2421	**1952**
0.7 and 0.7	1985	1965	1946	1854	2254	1989	**1853**
0.8 and 0.8	1758	1795	1754	1768	1984	1854	**1652**
0.9 and 0.9	1487	1354	1325	1421	1798	1524	**1324**
1 and 1	654	698	712	612	958	785	**514**

B. Measurement of Memory Consumption for Association Rule Generation:

Memory consumption refers to the amount of memory taken for generating the association rule, which is measured in terms of Mega Bytes (MB) (Sathya & Thangadurai, 2016):

$$M(Memory\ Consumption) = n \times M(n), \tag{9}$$

where M is the memory consumption for association rule generation, n is the number of frequent generated rules, and $M(n)$ is the memory required for generation of rules. When the memory consumption for association rule generation is low, a method is said to be more efficient. Table 5 shows the memory consumption of the algorithms by different support and confidence values. It has been realized that in most cases of support and confidence, ARM-AMO requires less memory consumption than the other algorithms used for comparison.

Table 5. Memory consumption of all algorithms for generating rules (MB) (Bold values indicate the best among all in a row)

Support and Confidence	HUIM-MMU	WFIM	HUIL	CP Tree	AMO	ARM-PSO	ARM-AMO
0.1 and 0.1	45.0	43.1	45.2	**42.1**	48.1	45.2	41.2
0.2 and 0.2	34.1	33.4	34.1	31.4	38.5	32.1	**30.3**
0.3 and 0.3	**22.4**	**22.4**	23.2	31.2	38.5	33.2	27.5
0.4 and 0.4	**22.0**	28.5	22.5	30.1	37.2	33.1	26.5
0.5 and 0.5	**21.2**	24.1	**21.2**	27.8	33.4	31.0	**21.2**
0.6 and 0.6	21.0	21.2	21.3	20.5	29.5	24.2	**19.5**
0.7 and 0.7	18.5	**16.5**	21.1	18.5	22.5	19.8	**16.5**
0.8 and 0.8	15.8	14.4	19.4	17.6	19.8	18.5	**13.9**
0.9 and 0.9	11.7	11.4	12.5	14.2	17.9	15.2	**11.2**
1 and 1	5.4	5.8	5.9	5.1	9.5	7.8	**3.1**

C. Measurement of computational time:

The computational time for frequent item set generation measures the amount of time taken for generating the frequent item sets with respect to given support and confidence values (Sathya & Thangadurai, 2016). It is measured in terms of milliseconds (ms) and mathematically formulated as follows,

$$RT = n \times T(n), \tag{10}$$

where *RT* is the running time, *n* represents the number of frequent item sets generated, and *T(n)* represented time taken for frequent item set generations. When the running time for frequent item set generation is low, the method is said to be more efficient. Table 6 shows the total running time of the algorithms by different support and confidence values. From this table, we realize that ARM-AMO runs quickly than the other algorithms in most case.

It has been realized that the association rule mining is modified using animal migration optimization is in most cases the best optimization approach among few. The weaker rules are migrated to have better fitness value so that the rules derived can be better.

Table 6. Total running time of all algorithms for generating rules (sec) (Bold values indicate the best among all in a row)

Support and Confidence	HUIM-MMU	WFIM	HUIL	CP Tree	AMO	ARM-PSO	ARM-AMO
0.1 and 0.1	350	331	352	**312**	421	352	**312**
0.2 and 0.2	**241**	234	241	314	395	341	**241**
0.3 and 0.3	**224**	**224**	232	312	385	332	275
0.4 and 0.4	**220**	285	225	301	372	331	**220**
0.5 and 0.5	**212**	241	212	278	334	310	**212**
0.6 and 0.6	210	212	213	205	295	242	**195**
0.7 and 0.7	185	**165**	211	185	225	198	**165**
0.8 and 0.8	158	144	194	176	198	185	**165**
0.9 and 0.9	117	114	125	142	179	152	**116**
1 and 1	54	58	71	61	95	78	**31**

4.4. Theoretical comparison

In what follows, we demonstrate the comparative analysis of all algorithms in Tables 7 and 8.

Table 7. Theoretical comparison of the algorithms

No.	Algorithms	Data Support	Advantage	Disadvantage	Published Year
1	HUIM-MMU	Minor Item set	Better than Apriori algorithm with minimum utility threshold	Takes lots of memory for large dataset	2016
2	WFIM	Closed item set	Consider both frequency and relative importance	Real life applications	2017
3	HUIL	Lattice item set	Mining all high association rules	Does not used to whole database to count frequent item set	2017
4	CP-Tree	Small dataset	Generate colossal patterns	Using tree structure create complexity	2017
5	AMO	Not frequently used	Better than Apriori and easy to use	Candidate set generated on the fly and size of the candidate set are large	2014

6	ARM-PSO	Small and large dataset	Generate interesting and understandable association rules only single scan	Takes lots of memory, time and memory consumption for large dataset	2014
7	**The proposed algorithm (ARM-AMO)**	Small and large dataset	Frequent item set generation only single scan	Useful for small and large dataset and generate minimum number of rules, memory consumption and time	

Table 8. Comparison by different criteria

Features	**HUIM-MMU**	**WFIM**	**HUIL**	**CP Tree**	**AMO**	**ARM-PSO**	**ARM-AMO**
Speed in Initial Phase	Slow	Slow	High	High	Slow	Slow	High
Speed in latter phase	High	Medium	Slow	Medium	Slow	Medium	High
Time	Medium	Medium	Less	Less	High	Medium	Very less
Memory Consumption	Medium	Medium	Less	Less	High	Medium	Very less
Accuracy	Less	More accurate than Apriori, Medium	Medium	Medium	Less	High and more accurate than Apriori Tid, Medium	Very High and more accurate

Table 8 indicates the performance measurement in term of speed at initial and later phases, time, memory consumptions and accuracy by five different measures namely Less, Medium, High, Very less and Very high. Specifically, ARM-AMO has high speed at initial and later phases, very less time and memory consumptions (e.g.~100 ms), and very high accuracy. Details of quantitative comparison will be performed in the next section. However, we aim to give the

first sight on the comparison between all methods based on our trials on many simulated and real datasets. Table 8 is a summary of those trials with the main aim for theoretical comparison.

5. EXPERIMENTS

5.1. Environment

The proposed algorithm (ARM-AMO) has been implemented in Matlab against the relevant algorithms namely HUIM-MMU (Lin *et al.*, 2015), WFIM (Wensheng *et al.*, 2017), HUIL (Mai *et al.*, 2017), CP Tree (Thanh-Long *et al.*, 2017), AMO (Li *et al.*, 2014) and ARM-PSO (Kuo *et al.*, 2011). They were executed in a computer with configuration: Microsoft Windows 7, 1.60GHz hard disk and 512MB RAM. Each algorithm is run 20 times for a case, and the average results are recorded.

Table 9. Experimental datasets

Name	**Size**	**Total transaction**	**Total items**	**Average number of Items per transaction**
Mushroom	3 MB	8124	497	23
Retails	3.97MB	11020	524	26
Accidents	33.8MB	189364	143	34
T10I4D100K	16MB	100000	119	10

The validation of the performance of all algorithms is carried out using experimental datasets were taken from Frequent Item-set Mining Dataset Repository named as Mushroom, Retails, Accidents and T10I4D100K (http://fimi.ua.ac.be/data/). Their statistics are shown in Table 9. They are well known data sets for the association rule mining with different sizes, total numbers of transaction, total items and average number of items in per transactions. The largest dataset is Accidents with 33.8 MB size (total transaction: 189364, total item: 143 and the average number of items per transactions: 34). The dataset contains largest number of items is Retails with 524 items and the average number of item per transaction is 26.

Parameters setting: Some values of parameters such as minimum support and confidence are ranged from 0.1 to 0.8. $P_{old}\,(i,d)$ is (1, 3) and n=4.

Objectives: We aim to evaluate the performance of association rule mining algorithms through the number of rules, the computational time and the memory consumption by various cases of parameters (Glass, 2013).

5.2. Comparative results by various cases

Experiments are divided into different typical cases (Cases 1 to 3) according to the minimum support (or support) and the minimum confidence (or confidence) values as follows.

5.2.1. Case 1: support = 0.4 and confidence = 0.3

Table 10. Comparison of algorithms in terms of number of rules, time and memory consumption in Case 1 (Bold values indicate the best among all in a row)

Dataset	**Parameters**	**HUIM -MMU**	**WFIM**	**HUIL**	**CP Tree**	**AMO**	**ARM-PSO**	**ARM-AMO**
Mushroom	Number of Rules (%)	77.18	77.35	77.14	77.54	75.82	77.24	**72.14**
	Time (Milliseconds)	159	158	158	157	164	158	**151**
	Memory Consumption (MB)	5.64	5.65	5.62	5.66	6.66	5.66	**4.65**
Retails	Number of Rules (%)	74.52	75.34	75.47	76.52	75.98	77.84	**73.97**
	Time (Milliseconds)	202	204	203	205	206	205	**192**
	Memory Consumption (MB)	5.85	5.84	5.85	5.86	6.88	5.88	**4.87**
Accidents	Number of Rules (%)	86.25	86.54	85.87	86.85	87.45	86.92	**81.18**
	Time (Milliseconds)	2121	2122	2121	2122	2201	2105	**1915**
	Memory Consumption (MB)	34.98	34.99	34.96	34.97	35.89	34.94	**32.96**
T10I4D100K	Number of Rules (%)	78.02	77.46	76.52	77.44	75.41	78.16	**71.10**
	Time (Milliseconds)	1205	1205	1204	1202	1213	1202	**1101**
	Memory Consumption (MB)	18.82	18.83	18.82	18.81	18.78	18.82	**17.80**

Table 10 indicates the average number of rules, the average computational time and the memory consumption of all algorithms. It is obvious that the proposed ARM-AMO algorithm

has better performance than all the others. For instance, in Mushroom dataset, the average number of rules and the time of ARM-AMO are 77.14 and 156 milliseconds respectively which are smaller than those of the other algorithms. Again, in the Accidents, the number of rule, the computational time and the memory consumption of ARM-AMO are 86.18, 2115 milliseconds and 34.96 MB which are much better as compared to the other algorithms. Similar cases happen in the T10I4D100K and Retails dataset.

Figures 3-5 indicate the overall performance of HUIM-MMU, WFIM, HUIL, CP Tree, AMO, ARM-PSO and ARM-AMO of all algorithms with different datasets in terms of the number of rules, time and memory consumption, respectively. It is clear that the ARM-AMO algorithm has smaller numbers of rules (Fig. 3), computational time (Fig. 4), and memory consumption (Fig. 5) than the other algorithms on all datasets namely Mushroom, Retails, Accidents and T10I4D100K.

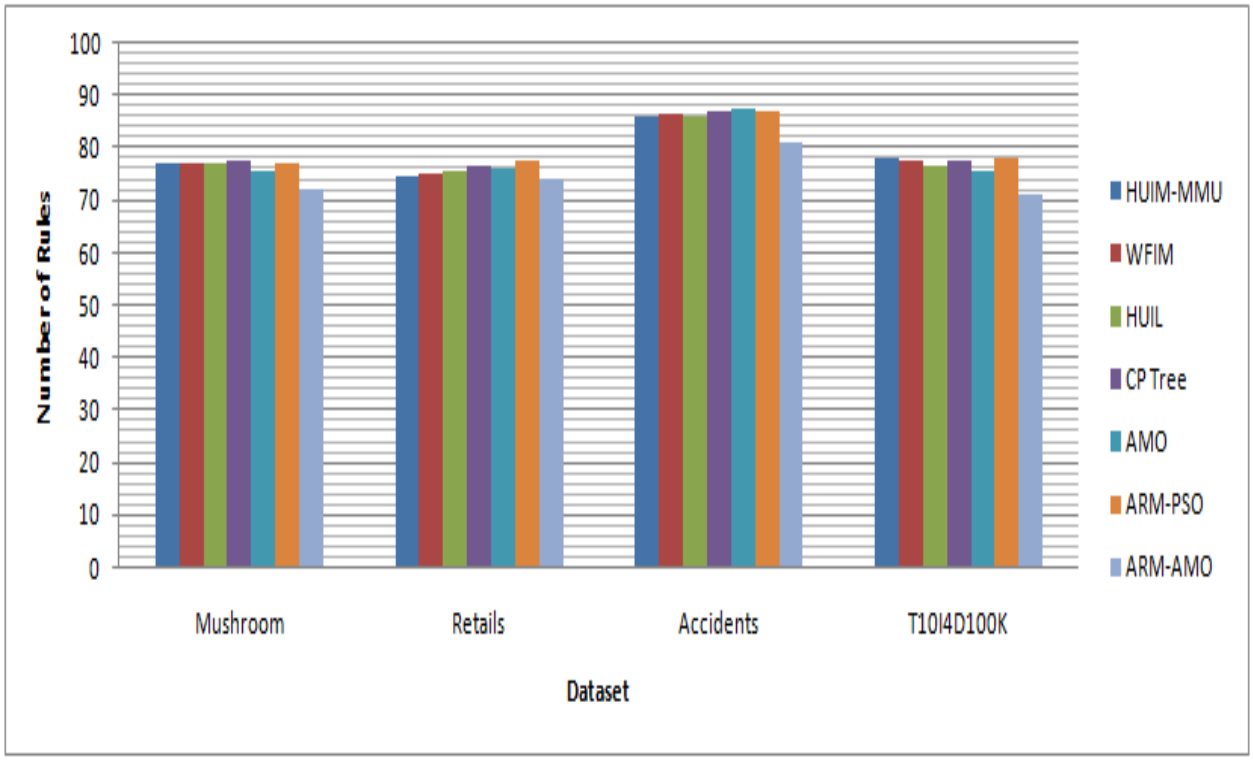

Fig. 3. Performance analysis in term of number of rules in Case 1

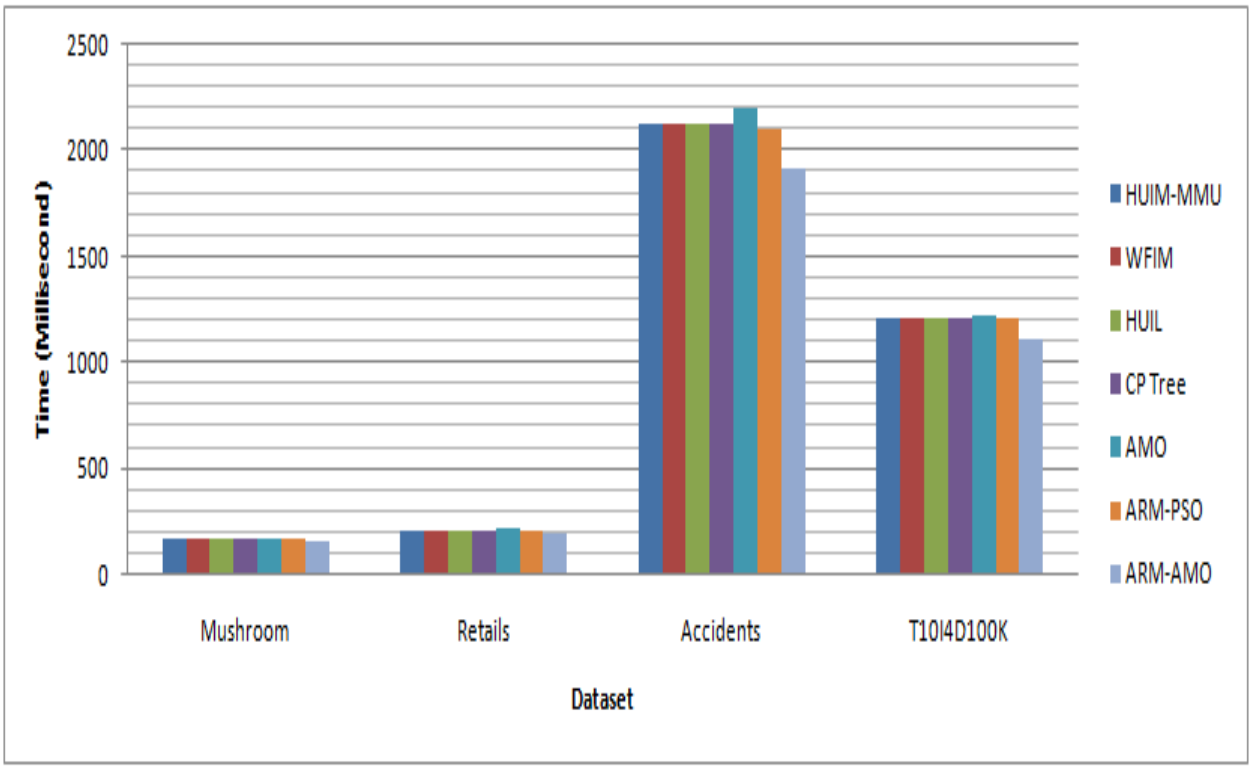

Fig. 4. Performance analysis in term of time in Case 1

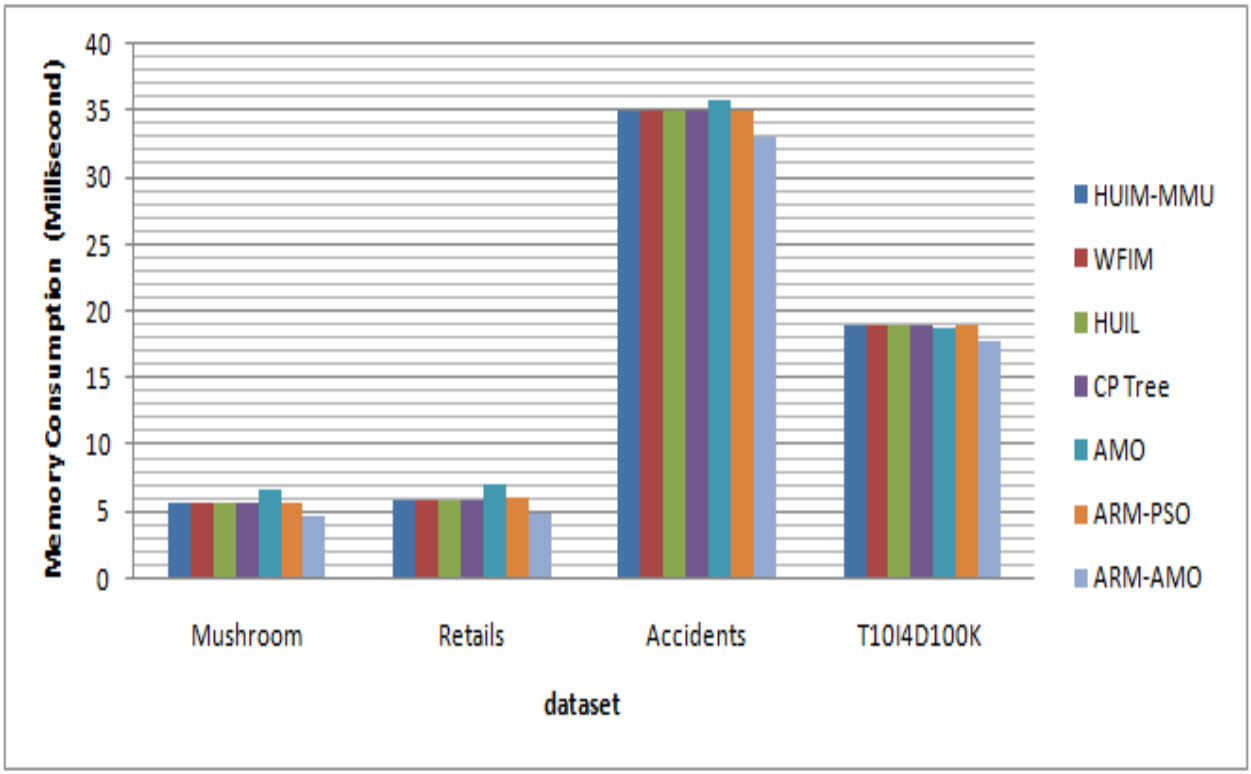

Fig. 5. Performance analysis in term of memory consumption in Case 1

5.2.2. Case 2: support = 0.5 and confidence = 0.6

Table 11 indicates the average number of rules, the average computational time and the memory consumption of all algorithms in this case. Herein, the value of support is smaller than that of the confidence. It can be seen that the proposed ARM-AMO algorithm has better performance than the others.

Table 11. Comparison of algorithms in terms of number of rules, time and memory consumption in Case 2 (Bold values indicate the best among all in a row)

Dataset	**Parameters**	**HUIM -MMU**	**WFIM**	**HUIL**	**CP Tree**	**AMO**	**ARM-PSO**	**ARM-AMO**
Mushroom	Number of Rules (%)	64.18	64.35	62.13	65.54	62.81	63.21	**52.14**
	Time (Milliseconds)	154	152	152	153	154	153	**131**
	Memory Consumption (MB)	4.77	4.76	4.74	4.75	4.69	4.73	**4.78**
Retails	Number of Rules (%)	88.52	87.74	85.77	86.82	89.88	88.84	**74.97**
	Time (Milliseconds)	195	196	192	195	198	194	**181**
	Memory Consumption (MB)	4.85	4.82	4.82	4.84	5.01	4.87	**4.81**
Accidents	Number of Rules (%)	77.25	78.54	75.47	76.85	77.45	76.95	**65.98**
	Time (Milliseconds)	2104	2105	2202	2103	2112	2103	**1852**
	Memory Consumption (MB)	34.05	34.04	34.02	34.05	35.05	34.95	**34.01**
T10I4D100K	Number of Rules (%)	77.52	76.46	76.52	76.64	76.52	77.36	**58.42**
	Time (Milliseconds)	1128	1125	1125	1131	1152	1132	**985**
	Memory Consumption (MB)	17.79	17.78	17.76	17.79	18.07	17.78	**17.29**

Specifically on Mushroom dataset, the average number of rules generated by ARM-AMO is 52.14 which is equal to (79%-83%) of those of the other algorithms. Likewise, the average computational time of ARM-AMO is 131 (seconds) which is equal to (85%-86%) of those of the other algorithms. However, memory consumption of ARM-AMO is 4.78 MB which is not smaller than memory of the others. We also made another test on the Retails dataset and got the results of average number of rules, computational time, and memory consumption of ARM-AMO are 74.97, 181 seconds, and 4.81 MB respectively. These numbers are approximately 85%, 93%, and 98% of those of the other algorithms respectively. If we take the average results on all datasets, the average number of rules, computational time, and memory consumption of ARM-AMO are 62.8, 787 seconds, and 15.2 MB respectively. They are still smaller than those of the other algorithms which the reduced percentages being 18%, 14%, and 1%.

Figures 6-8 indicate the overall performance of HUIM-MMU, WFIM, HUIL, CP Tree, AMO, ARM-PSO and ARM-AMO of all algorithms with different datasets in terms of the number of rules, time and memory consumption, respectively in this case. It can be seen that the ARM-AMO holds the best results among all. Specifically, ARM-AMO has smaller numbers of rules (Fig. 6), computational time (Fig. 7), and memory consumption (Fig. 8) than the other algorithms on all datasets namely Mushroom, Retails, Accidents and T10I4D100K.

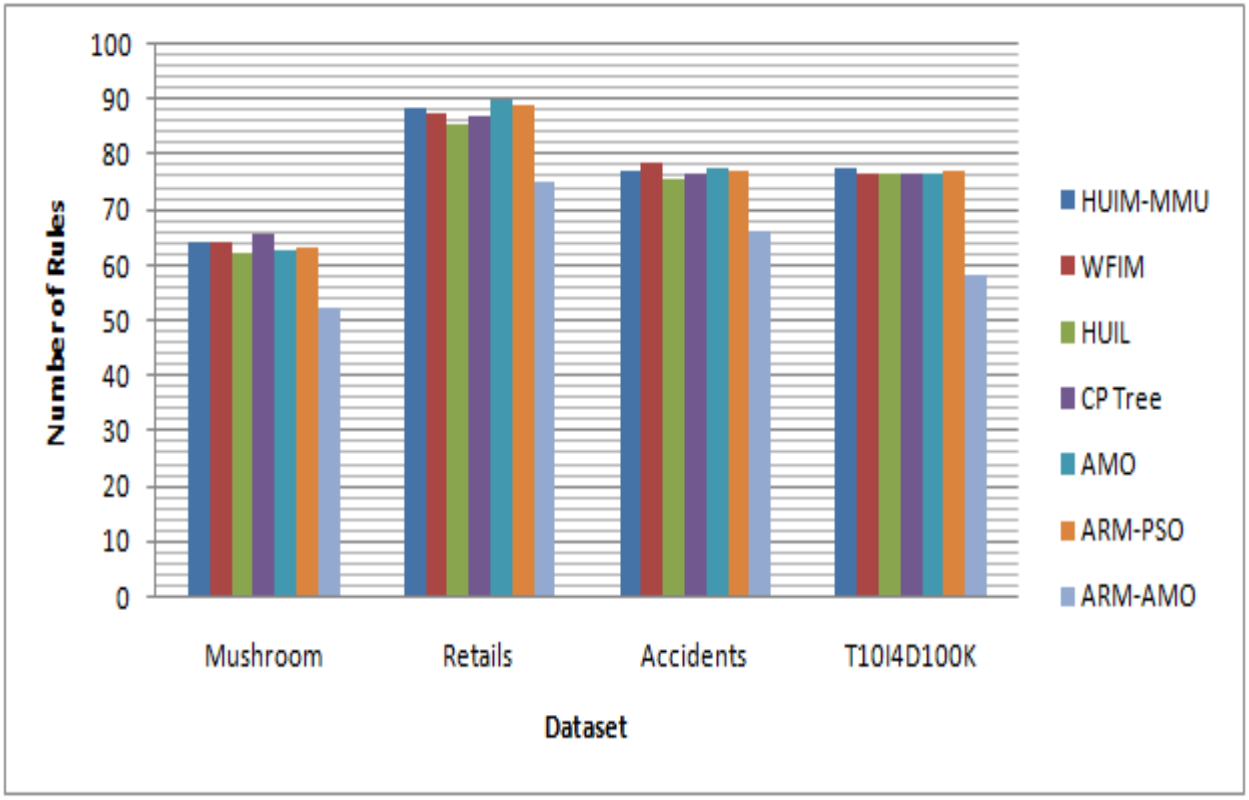

Fig. 6. Performance analysis in term of number of rules in case 2

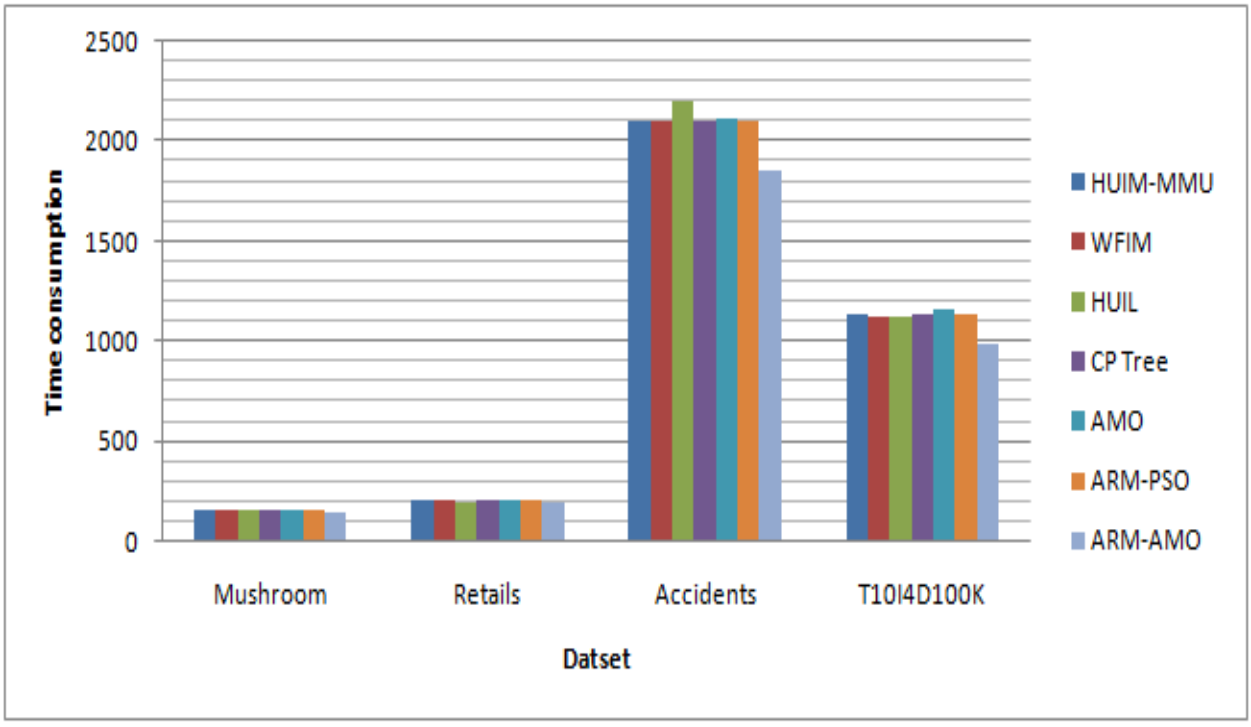

Fig. 7. Performance analysis in term of time in case 2

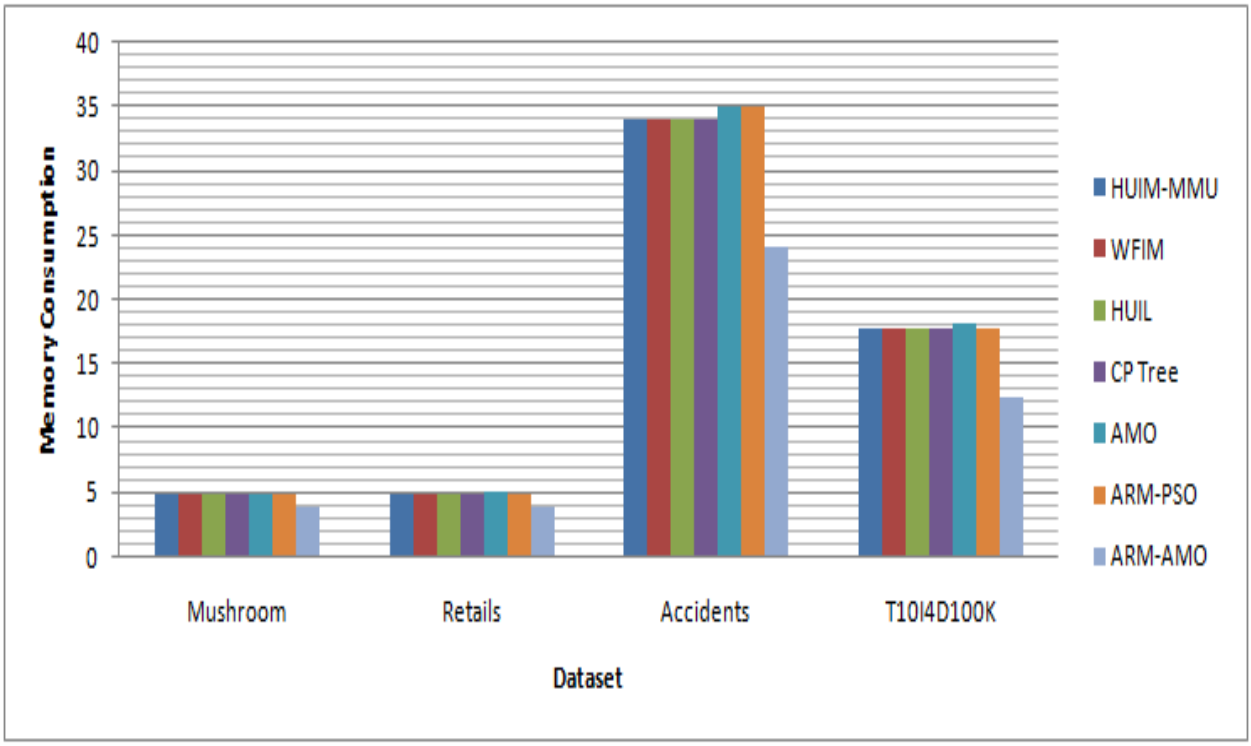

Fig. 8. Performance analysis in term of Memory Consumption (MB) in case 2

5.2.3. Case 3: support = 0.6 and confidence = 0.7

Table 12 indicates the average number of rules, the average computational time and the memory consumption of all algorithms in this case. Again, the value of support is smaller than that of the confidence. However, its values are larger than those in Case 2. It can be seen that the proposed ARM-AMO algorithm has better performance than the others. For instance, in Mushroom, Retails, Accident and T10I4D100K datasets, the number of rules generated by ARM-AMO are51.24, 52.97, 45.98 and 55.12 respectively. The computational time taken to generate these rules is 119, 185, 1850 and 1074, respectively. The memory consumption is 3.86, 3.52, 33.32 and 17.06, respectively. They are all better than those of the other algorithms. Thus, it has been concluded that the proposed algorithm performs better in term of large data size and maximum support values.

Table 12. Comparison of algorithms in terms of number of rules, time and memory consumption in Case 3 (Bold values indicate the best among all in a row)

Dataset	**Parameters**	**HUIM -MMU**	**WFIM**	**HUIL**	**CP Tree**	**AMO**	**ARM-PSO**	**ARM-AMO**
Mushroom	Number of Rules (%)	54.28	55.15	52.14	54.54	51.82	54.24	**41.24**
	Time (Milliseconds)	123	125	124	123	129	126	**119**
	Memory Consumption (MB)	3.85	3.85	3.82	3.86	3.81	3.83	**2.86**
Retails	Number of Rules (%)	54.52	55.34	55.47	56.52	59.98	57.84	**48.97**
	Time (Milliseconds)	188	189	188	187	195	191	**175**
	Memory Consumption (MB)	3.89	3.88	3.87	3.84	3.85	3.88	**2.98**
Accidents	Number of Rules (%)	49.25	48.14	48.47	46.85	47.45	48.95	**42.98**
	Time (Milliseconds)	1860	1861	1863	1862	1869	1865	**1550**
	Memory Consumption (MB)	33.85	33.86	33.86	33.85	34.85	33.86	**27.32**
T10I4D100K	Number of Rules (%)	57.42	57.46	57.52	57.64	57.42	57.36	**51.12**
	Time (Milliseconds)	1091	1189	1196	1086	1093	1089	**874**
	Memory Consumption (MB)	17.73	17.74	17.74	17.75	18.71	17.75	**14.06**

Figures 9-11 indicate the overall performance of HUIM-MMU, WFIM, HUIL, CP Tree, AMO, ARM-PSO and ARM-AMO of all algorithms with different datasets in terms of the number of rules, time and memory consumption, respectively in Case 3. Similar results to the two previous cases are realized herein where ARM-AMO has smaller numbers of rules (Fig. 9), computational time (Fig. 10), and memory consumption (Fig. 11) than the other algorithms on all datasets namely Mushroom, Retails, Accidents and T10I4D100K.

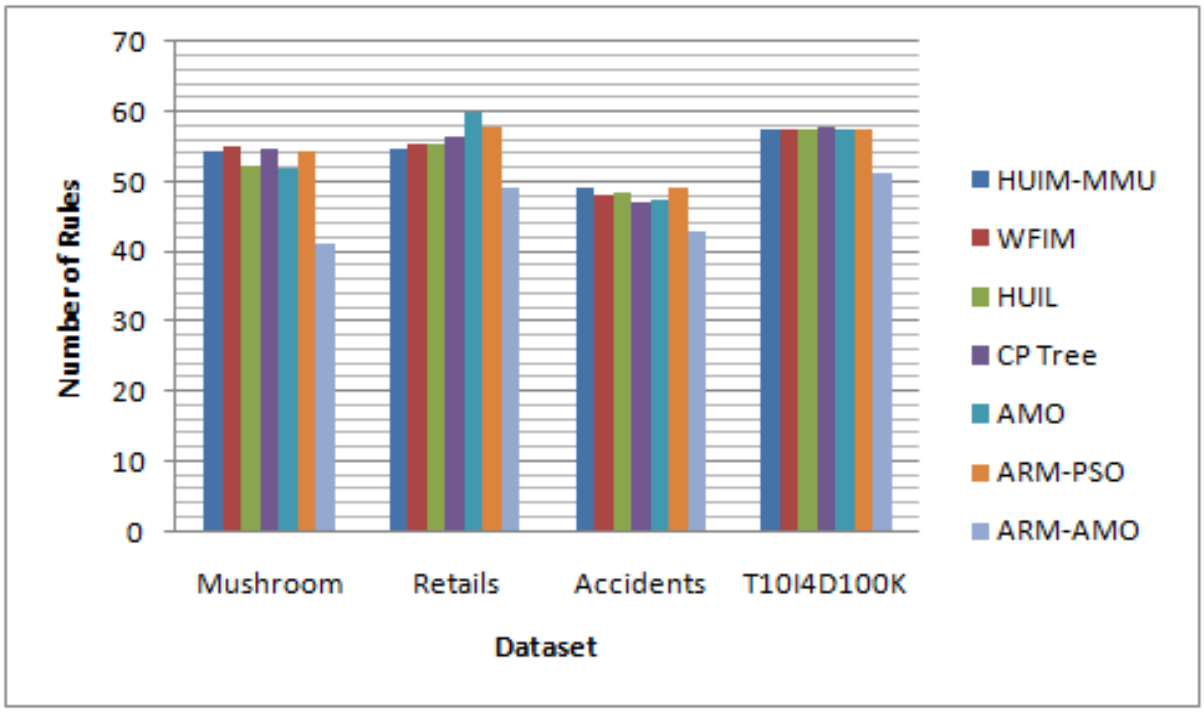

Fig. 9. Performance analysis in term of number of rules in case 3

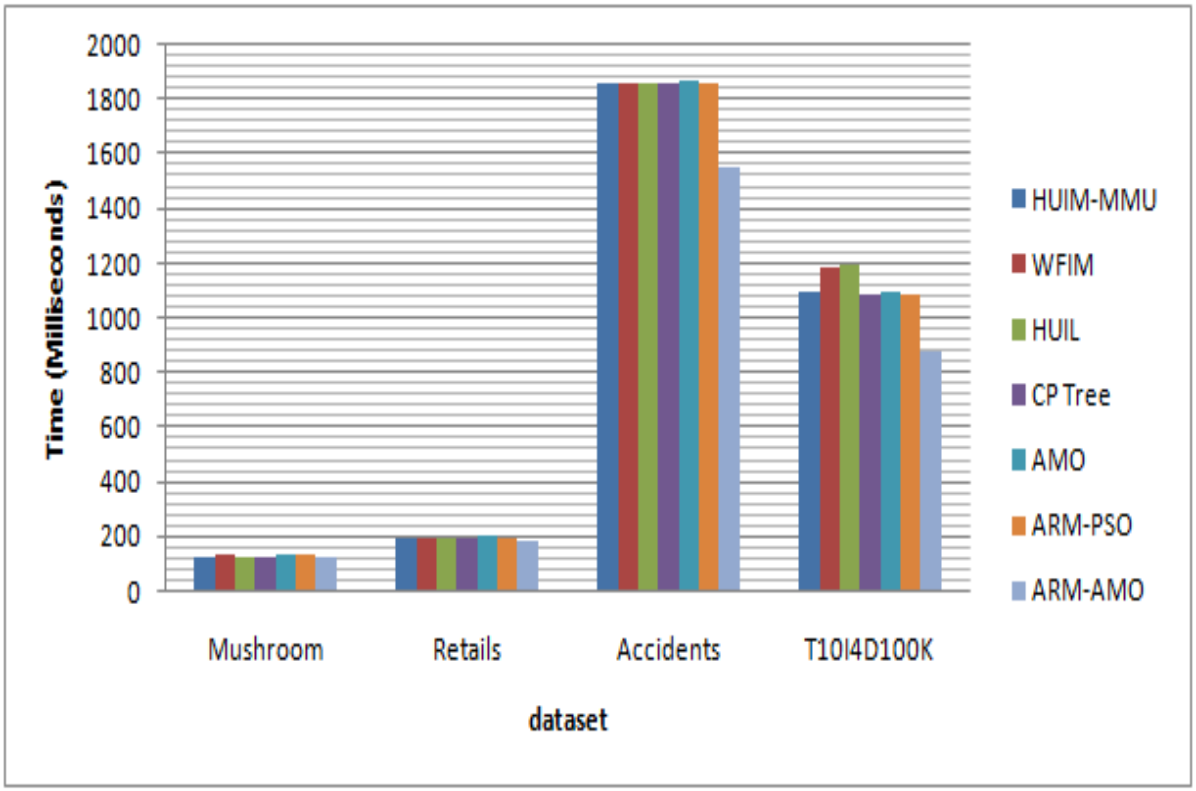

Fig. 10. Performance analysis in term of time in case 3

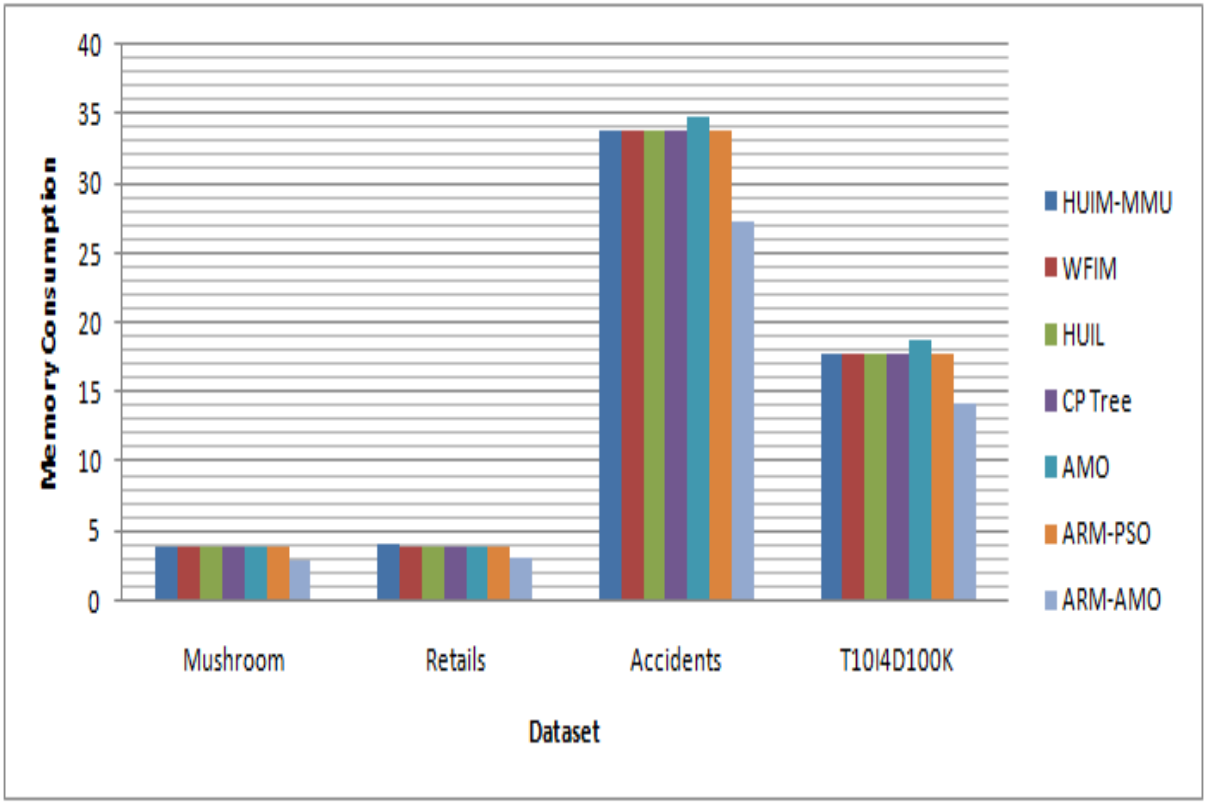

Fig. 11. Performance analysis in term of memory consumption in case 3

5.2.4. Summary

Besides three above cases, in order to verify the efficiency of the proposed algorithm, we made more cases of support and confidence values as in Table 13. These values are chosen within [0, 1] so that we have 15 cases in total. By the similar calculation process indicated above, we derive the results in Figures 12-14 demonstrating performance of HUIM-MMU, WFIM, HUIL, CP Tree, AMO, ARM-PSO and ARM-AMO in terms of the number of rules, time and memory consumption, respectively (Table 13). It can be seen that the proposed algorithm has better performance than the related ones.

Table 13. Values of support and confidence in other cases

CASE														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.1 and 0.1	0.2 and 0.2	0.2 and 0.1	0.3 and 0.2	0.4 and 0.3	0.3 and 0.4	0.5 and 0.4	0.4 and 0.5	0.6 and 0.5	0.5 and 0.6	0.7 and 0.6	0.6 and 0.7	0.8 and 0.7	0.7 and 0.8	0.8 and 0.8

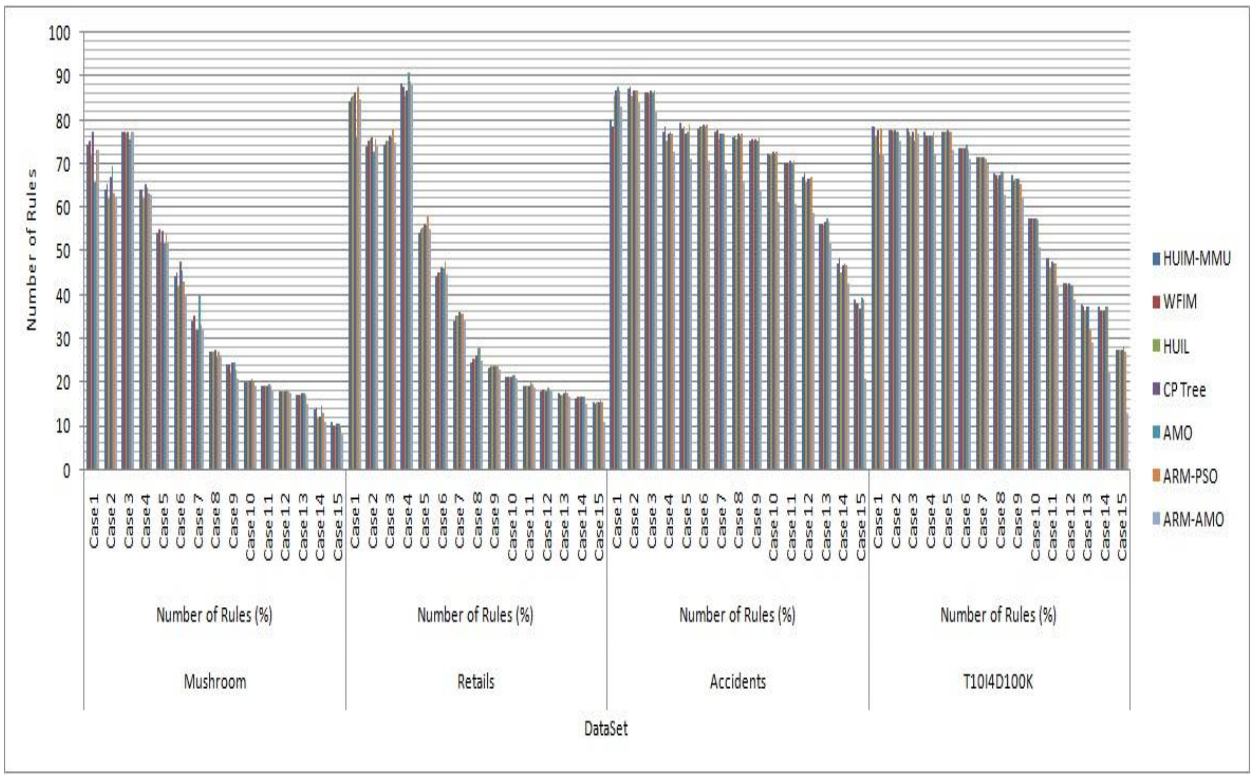

Fig. 12. Performance analysis for number of rule generation in other cases

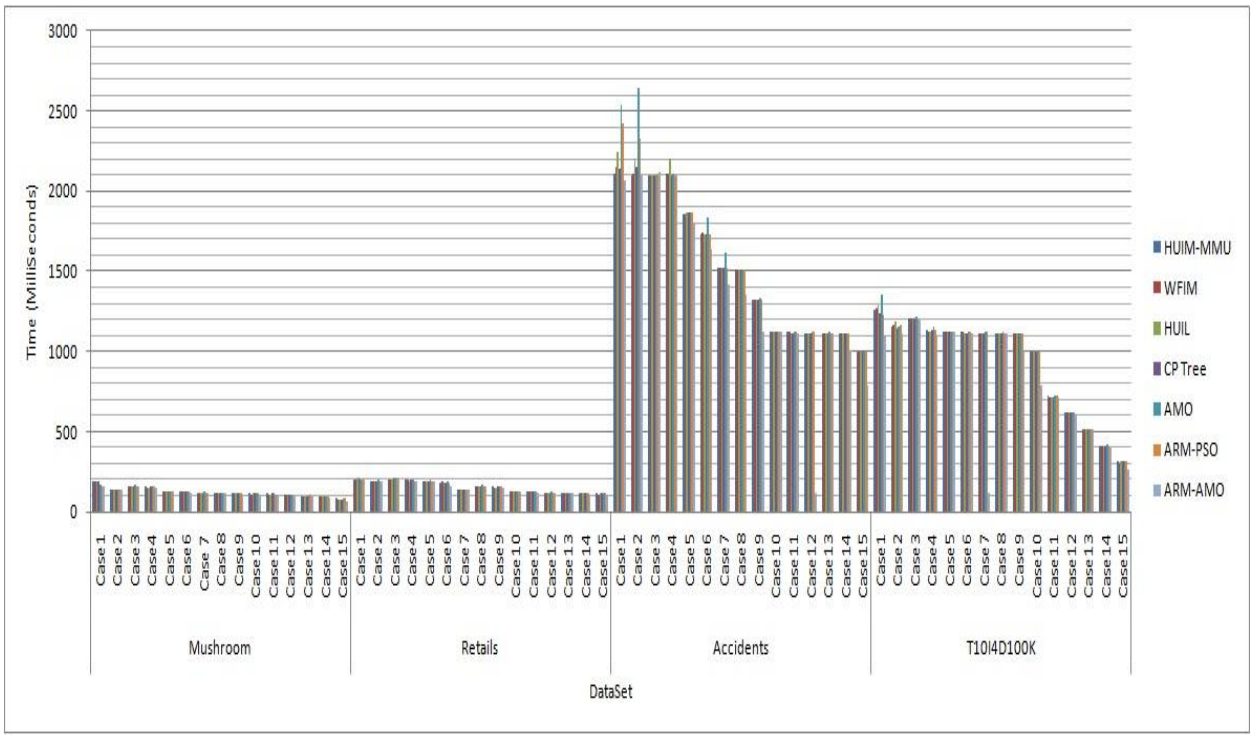

Fig. 13. Performance analysis for time in other cases (Milliseconds)

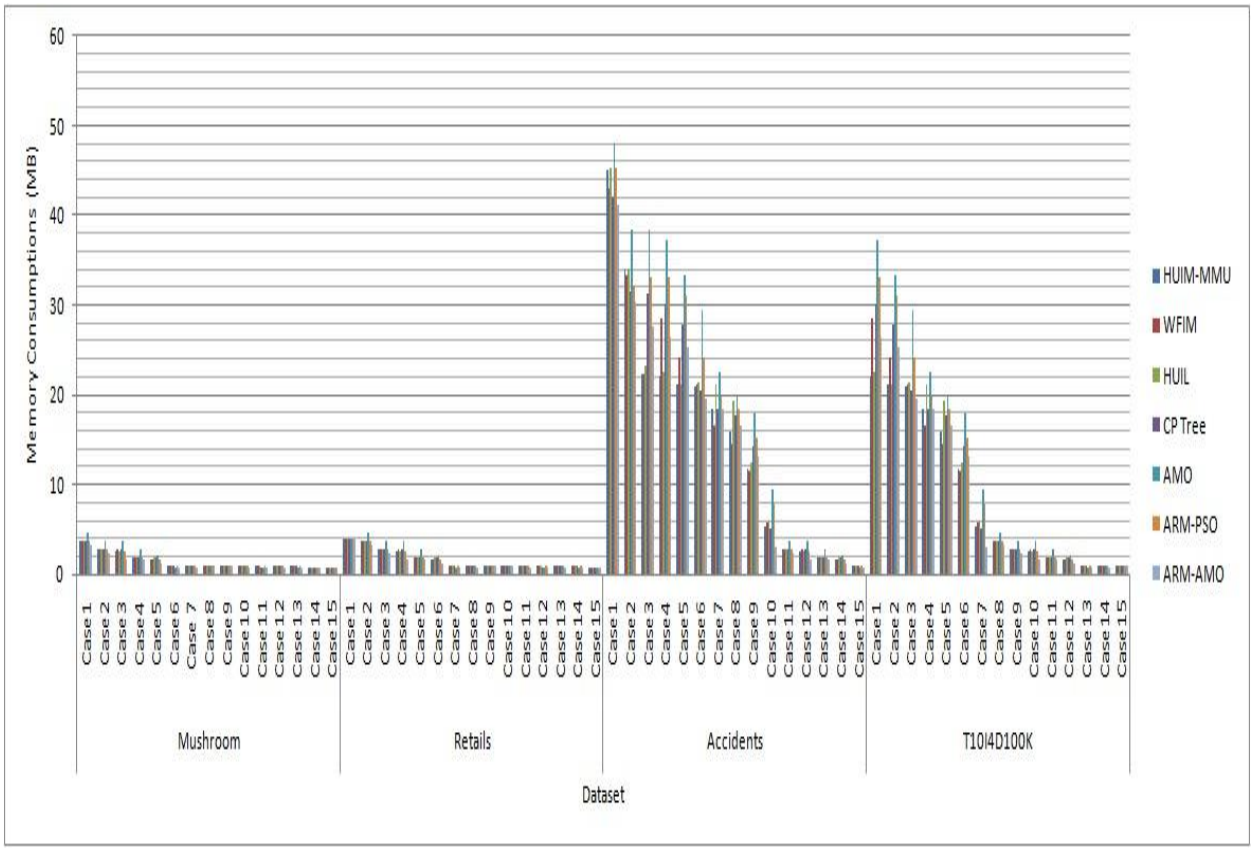

Fig. 14. Performance analysis for memory consumption in other cases (MB)

6. Conclusions

In this paper, we proposed a new association rule mining method based on Animal Migration Optimization to reduce the number of rules, computational time and memory consumption. It is based on the idea that rules which are not of high support and unnecessary are deleted. Only frequent rules are kept and integrated into the fitness function of Animal Migration Optimization. The experiments in the benchmark Frequent Item-set Mining Dataset Repository datasets affirmed that the number of rules, the computational time and memory consumption of the proposed ARM-AMO method are better than those of the other existing algorithms. It has been indicated that ARM-AMO provides better performance with minimization of running time for frequent item set generation by 39% and also reduces the number of rules generated by 52%,

when compared with the other algorithms in large datasets using maximum support and confidence values. The finding is significant to mining association rules in real applications.

Future works of this research will investigate the incorporation of ARM-AMO with the parallel strategy to both enhance the computational time and the quality of rules. Besides, new methods of updating rule and rule optimization should be researched intensively to boost the performance of ARM-AMO. Lastly, a framework for distributed databases like in (Goyal *et al.*, 2017) or clustering models (Son, Cuong & Long, 2013; Thong & Son, 2016; Son & Hai, 2016; Son & Phong, 2016; Son, Viet & Hai, 2017; Wang *et al.*, 2017) is also our target.

REFERENCES

Anuradha, R., &Rajkumar, N. (2017). Mining generalized positive and negative inter-cross fuzzy multiple-level coherent rules. *Journal of Intelligent & Fuzzy Systems*, *32*(3), 2269-2280.

Badhe, V., Thakur, R. S., & Thakur, G. S. (2015).Vague Set Theory for Profit Pattern and Decision Making in Uncertain Data. *International Journal of Advanced Computer Science and Applications*, *6*(6), 58-64.

Barati, M., Bai, Q., & Liu, Q. (2017). Mining semantic association rules from RDF data. *Knowledge-Based Systems*, *133*, 183-196.

Chen, C. X., Shen, J. J., Chen, B., Shang, C. X., & Wang, Y. C. (2011). An improvement apriori arithmetic based on rough set theory. In *Circuits, Communications and System (PACCS), 2011 Third Pacific-Asia Conference on* IEEE, 1-3.

Das, S. R., Panigrahi, P. K., Das, K., & Mishra, D. (2012).Improving rbf kernel function of support vector machine using particle swarm optimization. *International Journal of Advanced Computer Research, 2(4)*, 130-135.

Feng, F., Cho, J., Pedrycz, W., Fujita, H., & Herawan, T. (2016). Soft set based association rule mining. *Knowledge-Based Systems*, *111*, 268-282.

Glass, D. H. (2013). Confirmation measures of association rule interestingness. *Knowledge-Based Systems*, *44*, 65-77.

Goyal, L. M., Beg, M. M., & Ahmad, T. (2017). An Efficient Framework for Mining Association Rules in the Distributed Databases. *The Computer Journal*, 1-13.

Huang, Y., Li, T., Luo, C., Fujita, H., & Horng, S. J. (2017). Matrix-based dynamic updating rough fuzzy approximations for data mining. *Knowledge-Based Systems*, *119*, 273-283.

Indira, K.& Kanmani, S.(2012).Association Rule Mining using Self Adaptive Particle Swarm Optimization. *International journal of computer application, Special Issue on "Computational Intelligence & Information Security" CIIS (1),* 27-31.

Jerry, C. W. L., Wensheng G., Philippe F.-V., Tzung-P. H., Vincent S. T. (2016). Fast algorithms for mining high-utility itemsets with various discount strategies.*Advanced Engineering Informatics*, 30(2), 109-126.

Kieu, T., Vo, B., Le, T., Deng, Z. H., & Le, B. (2017). Mining top-k co-occurrence items with sequential pattern. *Expert Systems with Applications*, *85*, 123-133.

Kuo, R. J., Chao, C. M., & Chiu, Y. T. (2011). Application of particle swarm optimization to association rule mining. *Applied Soft Computing*, *11*(1), 326-336.

Li, X., Zhang, J., & Yin, M. (2014). Animal migration optimization: an optimization algorithm inspired by animal migration behavior. *Neural Computing and Applications*, *24*(7-8), 1867-1877.

Lin, J. C. W., Li, T., Fournier-Viger, P., & Hong, T. P. (2015).A fast Algorithm for mining fuzzy frequent itemsets. *Journal of Intelligent & Fuzzy Systems*, *29*(6), 2373-2379.

Mai, T., Vo, B., & Nguyen, L. T. (2017). A lattice-based approach for mining high utility association rules. *Information Sciences*, *399*, 81-97.

Martínez-Ballesteros, M., Bacardit, J., Troncoso, A., &Riquelme, J. C. (2015).Enhancing the scalability of a genetic algorithm to discover quantitative association rules in large-scale datasets. *Integrated Computer-Aided Engineering*, *22*(1), 21-39.

Mlakar, U., Zorman, M., Fister Jr, I., & Fister, I. (2017).Modified binary cuckoo search for association rule mining. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-12.

Nguyen L. T., Vo B., Hong T.-P., and Thanh H. C.. (2012). Classification based on association rules: A lattice-based approach.*Expert Systems with Applications*,*39*(13), 357–366.

Nithya, N. S., & Duraiswamy, K. (2015). Correlated gain ratio based fuzzy weighted association rule mining classifier for diagnosis health care data. *Journal of Intelligent & Fuzzy Systems*, *29*(4), 1453-1464.

Oladele, R. O., & Sadiku, J. S. (2013).Genetic algorithm performance with different selection methods in solving multi-objective network design problem. *International Journal of Computer Applications*, *70*(12).

Pears, R., & Koh, Y. S. (2011).Weighted association rule mining using particle swarm optimization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 327-338).Springer, Berlin, Heidelberg.

Randy L. &Haupt S. (2003).*Practical Genetic Algorithms*. Wiley-IEEE Publication.

Ratner, B. (2017). *Statistical and Machine-Learning Data Mining.* CRC Press.

Rauch, J. (2015). Formal Framework for Data Mining with Association Rules and Domain Knowledge–Overview of an Approach. *Fundamenta Informaticae*, *137*(2), 171-217.

Sathya M. & Thangadurai K. (2016).Association Rule Generation Using E-ACO Algorithm. *International Journal of Control Theory and Applications*, *27(9),* 513-521.

Shirsath, P. A., & Verma, V. K. (2013).A Recent Survey on Incremental Temporal Association Rule Mining. *International Journal of Innovative Technology and Exploring Engineering*, 3(1), 85-92.

Shmueli, G., & Lichtendahl Jr, K. C. (2017), *Data Mining for Business Analytics: Concepts, Techniques, and Applications.* John Wiley & Sons.

Slim B., Rabie S., Sadok B. Y., Engelbert M. N. (2014). Mining Undominated Association Rules Through Interestingness Measures. *International Journal on Artificial Intelligence Tools*, 23(4), 95-102.

Son, L. H., & Phong, P. H. (2016). On the performance evaluation of intuitionistic vector similarity measures for medical diagnosis. *Journal of Intelligent & Fuzzy Systems*, *31*(3), 1597-1608.

Son, L.H., Cuong, B. C., & Long, H. V. (2013).Spatial interaction–modification model and applications to geo-demographic analysis. *Knowledge-Based Systems*, *49*, 152-170.

Son, L.H., Hai, P.V. (2016). A novel multiple fuzzy clustering method based on internal clustering validation measures with gradient descent. *International Journal of Fuzzy Systems*, *18*(5), 894-903.

Son, L.H., Viet, P.V., Hai, P.V. (2017). Picture inference system: a new fuzzy inference system on picture fuzzy set. *Applied Intelligence*, *46*(3), 652-669.

Song, A., Ding, X., Chen, J., Li, M., Cao, W., & Pu, K. (2016). Multi-objective association rule mining with binary bat algorithm. *Intelligent Data Analysis*, *20*(1), 105-128.

Song, K., & Lee, K. (2017). Predictability-based collective class association rule mining. *Expert Systems with Applications*, *79*, 1-7.

Tahrima H., Rezaul K., Samiullah, Chowdhury F. A. (2017). An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure. *Expert Systems with Applications*, 67,252-271.

Thabtah, F., Qabajeh, I., & Chiclana, F. (2016).Constrained dynamic rule induction learning. *Expert Systems with Applications*, *63*, 74-85.

Thang M., Bay V., Loan T.T. N.(2017). A lattice-based approach for mining high utility association rules. *Information Sciences*, 399, 81-97.

Thanh-Long N., Bay V., Vaclav S. (2017). Efficient algorithms for mining colossal patterns in high dimensional databases. *Knowledge-Based Systems*, 122, 75-89.

Thong, P. H., Son, L.H. (2016).A novel automatic picture fuzzy clustering method based on particle swarm optimization and picture composite cardinality. *Knowledge-Based Systems*, *109*, 48-60.

Vo, B., Pham, S., Le, T., & Deng, Z. H. (2017). A novel approach for mining maximal frequent patterns. *Expert Systems with Applications, 73*, 178-186.

Wang L., Li, S. L., Sun, H., & Peng, K. X. (2016). A classification and regression algorithm based on quantitative association rule tree. *Journal of Intelligent & Fuzzy Systems*, *31*(3), 1407-1418.

Wang, G., Zhang, G., Choi, K. S., & Lu, J. (2017). Deep Additive Least Squares Support Vector Machines for Classification With Model Transfer. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. DOI: 10.1109/TSMC.2017.2759090.

Wensheng G, Jerry C.-W. L., Philippe F.-V., Han-C. C., Jimmy M.-T. W., Justin Z. (2017). Extracting recent weighted-based patterns from uncertain temporal databases. *Engineering Applications of Artificial Intelligence, 61*, 161-172.

Yan, C., Sun, H., & Liu, W. (2016).Study of fuzzy association rules and cross-selling toward property insurance customers based on FARMA. *Journal of Intelligent & Fuzzy Systems*, *31*(6), 2789-2794.

Yun, U., Kim, D., Ryang, H., Lee, G., & Lee, K. M. (2016). Mining recent high average utility patterns based on sliding window from stream data. *Journal of Intelligent & Fuzzy Systems*, *30*(6), 3605-3617.