A Survey on Association Rules Mining Using Heuristics

Authors:

First author: S. Mohssen Ghafari
School of Science & Technology, International Hellenic University, 14th km Thessaloniki – Moudania, 57001 Thermi, Greece
+30 2310807575, m.ghafari@ihu.edu.gr

Second author: Christos Tjortjis
School of Science & Technology, International Hellenic University, 14th km Thessaloniki – Moudania, 57001 Thermi, Greece
+30 2310807576, c.tjortjis@ihu.edu.gr

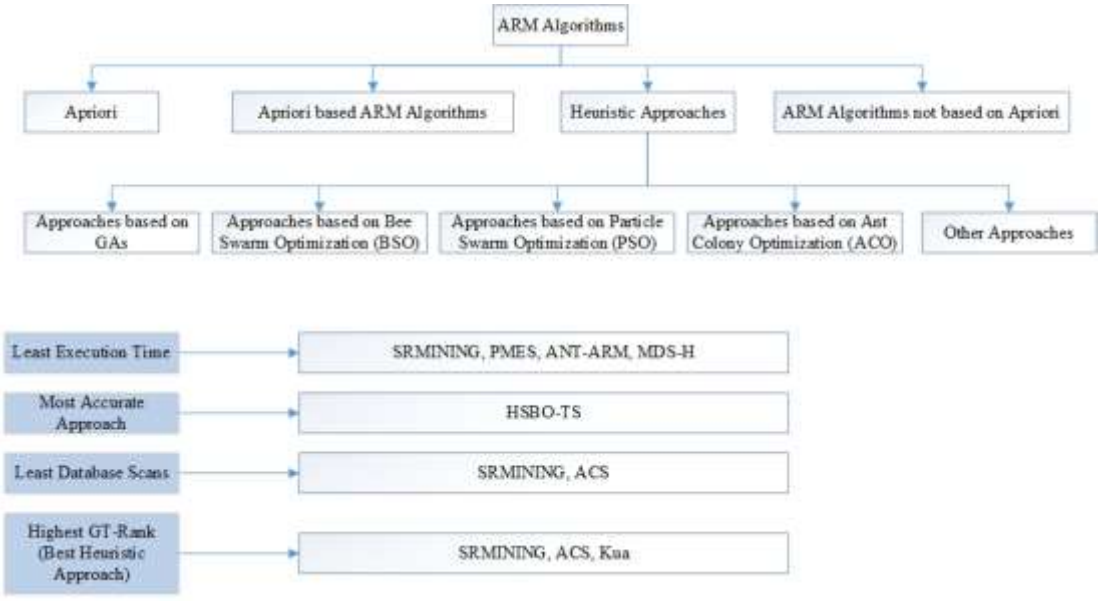
Abstract

Association Rule Mining (ARM) is a popular data mining method. There are many approaches for mining frequent rules and patterns from a database; one employs heuristics. Many heuristic approaches have been proposed but, to the best of our knowledge, there is no comprehensive literature review on such approaches, only a limited attempt. This gap needs to be filled. This paper reviews heuristic approaches for ARM and points out their most significant strengths and weaknesses. We propose eight performance metrics, such as execution time, memory consumption, completeness and interestingness, we compare approaches against these performance metrics and discuss our findings. For instance, comparison results indicate that SRmining, PMES, Ant-ARM, and MDS-H are the fastest heuristic ARM algorithms. HSBO-TS is the most complete one, whilst SRmining and ACS require only one database scan. In addition, we propose a parameter, named *GT-Rank* for ranking heuristic ARM approaches, and based on that, ARMGA, ASC, and Kua emerge as the best approaches. We also consider ARM algorithms and their characteristics as transactions and items in a transactional database, respectively, and generate association rules (ARs) that indicate research trends in this area.

Keywords

Association Rule Mining, Heuristics, Survey, Literature Review

Graphical/Visual Abstract and Caption



Reviewing heuristic ARM approaches, comparing them based on performance metrics like execution time, completeness, number of database scans etc. Finally, proposing a parameter, named GT-Rank for finding the best heuristic ARM approaches.

Introduction

The recent huge increase on the amount of available data caused the emergence of data mining techniques for extracting knowledge from the databases. *Association Rule Mining (*ARM) is one of the most popular methods in this area. In 1993, Agrawal et al. proposed *Apriori* to extract frequent rules and patterns from databases [1], [2]. Subsequently, many researchers attempted to improve this process, such as works presented by Yan et al. [3], Djenouri et al. [4], Soysal [5], Goyal et al. [6], [7], Drias [8], and in [9]. Apriori is a time consuming algorithm. This motivated many researchers to focus first on proposing faster algorithms, such as those presented by Hong and Bian [10], Li and Yang [11], Kau and Shih [12], Guo and Zhou [13], and in [14]. Many of them, like FP-growth were successful in extracting frequent rules in a short period of time: a significant improvement compared to Apriori [14].

However, researchers uncovered more drawbacks of Apriori to be addressed. One of them is the large number of generated candidate frequent itemsets. Researchers focused on reducing this number [7], [14]. Apriori has another drawback: the need for user input to set the required parameter values, such as the *minimum confidence* and *support*. This manual approach is not easy, especially in large databases and could even have negative impact on the quality of generated rules [3]. As a result, researchers proposed approaches for setting these parameters automatically or even without any need for such predefined values, as discussed by Yan et al. [3], Kuoa et al. [15], Dash et al. [16], and Bidgoli et al. [17]. Another subject that became popular was extracting *interesting* rules and patterns [3], [16], [17], [18]. Many methods have been proposed to define interestingness and extract interesting patterns and rules.

Despite these improvements, there are still many open challenges. Many ARM approaches have been designed for small or medium datasets. Therefore, algorithms need to address scalability. In addition, an important gap in ARM literature is the consideration of all desirable aspects that an ARM approach should have. There are many algorithms, which only improve one such aspect. For instance, they are executed in a short period of time, but they generate some wrong rules, i.e. rules which do not reflect the actual transactions stored in the database [3]. They may also have small memory usage, but they scan the database several times [19].

Applying *heuristics* could be a convenient solution for many ARM algorithms problems and challenges. Heuristic algorithms tend to have low execution time and they do not need any knowledge about the problem domain in advance. Combination of these advantages makes them a good choice even for optimization problems. Many heuristic algorithms have been proposed, and many of them have been employed to design ARM approaches. In this paper, we discuss these approaches and point out their advantages and disadvantages. It should be noted that although there are some limited attempts to review classic ARM algorithms by Borgelt [20], Nath et al. [21], Le et al. [22], Zhang and He [23], Elloumi and Zolmaya [24], Mukhopadhyay et al. [25], only one minor attempt on reviewing heuristic approaches was, to the best of our knowledge, made by Del et al. [26]. However, there is no comprehensive study, which compares heuristic approaches and points out their strengths and weaknesses. Clearly, a comprehensive survey on ARM algorithms using heuristics would benefit researchers in this area, particularly in helping them to identify open challenges and directions for future work.

The rest of this paper discusses ARM algorithms background and presents Apriori and FP-growth in section 2. Section 3 shapes a comprehensive literature review on heuristics used for ARM. Section 4 introduces some performance metrics for comparing heuristic approaches. Section 5 compares heuristic approaches using the proposed parameters. Finally, a detailed discussion and conclusions can be found in sections 6 and 7, respectively.

ARM Algorithms

This section introduces key concepts on ARM algorithms, by presenting one of the best-established algorithms in the field: Apriori. It also discusses two broad ARM algorithm categories: Exhaustive approaches, like Apriori, and heuristic approaches. All the algorithms are briefly discussed along with their main features.

Preliminaries

This section defines several important concepts and measures in the ARM field, as follows:

- *Rule*: is a conditional structure (If (condition) then result) that indicates the relation between an antecedent and a consequent.

  $$A \rightarrow B$$

  where A is the antecedent and B is the consequent.
- *Itemset*: is a set of items in a (transactional) database.
- *Support*: is the frequency of an itemset in a database.
- *Confidence*: is the percentage of occasions that a rule is true, over the times it is applicable in the database (see formula 3).
- *Lift* value of an association rule: is the ratio of the confidence of the rule and the expected confidence of the rule.

$$Lift(A \rightarrow B) = Support(A \cup B) / (Support(A) \times Support(B))$$

- *Conviction*: is the proportion of the expected support of A that may be true without B or it is the incorrect prediction by the rule.

$$Conviction(A \rightarrow B) = (1-Support(B))/ (1-Confidence(A \rightarrow B))$$

- *Interestingness*: employing some metric to evaluate the extracted rules and extracting the ones that are more interesting to the user (see formula 7 for example).
- *Comprehensibility*: is the ratio of the length of antecedent and consequent. It is based on the "Rule of Simplicity", i.e. a rule is simpler, if it is shorter.

  $$Comprehensibility = \log(1+|B|)/\log(1+|A \cup B|)$$

  where $|B|$ is the size of consequent and $|A \cup B|$ is the total size of the rule.
- *Fitness Function*: is used to evaluate the quality of generated rules. Each ARM approach could have different fitness function.
- *Rule reliability*: usually addressed by the confidence quality measure, by considering support of the antecedent, support of the consequent, and support of the rule as a whole.
- *Completeness:* in the context of this paper, we use the term completeness as the proportion of rules generated by a heuristic method over that of an exhaustive method like Apriori [2], which does not trade off the number of rues extracted over speed.

Exhaustive ARM Approaches

Such ARM approaches try to exhaustively find all possible solutions, in contrast with heuristic ones. However, this strategy may face problems in bigger databases and larger search spaces and requires high amount of computational power and memory space. In this paper, we divide exhaustive ARM approaches into two subcategories: Apriori-based and non Apriori-based. First, we briefly present Apriori.

Apriori

Apriori is the best known ARM algorithm for extracting *frequent patterns* from databases [1], [2]. It works in a breadth first fashion.

Assuming there is a set of transactions $T = \{t_1, t_2, \dots, t_n\}$ and a set of items $I = \{i_1, i_2, \dots, i_n\}$, the main goal is to find all frequent {X, Y}, where X and Y are called *itemsets*, i.e. they may contain one or more items. Each extracted *rule* is of the form:

$$X \rightarrow Y \quad (1)$$

meaning that when a transaction t_x contains itemset X, it should also contain itemset Y. Itemsets X and Y should have no items in common:

$$X \cap Y = \Phi \quad (2)$$

Apriori, and most ARM algorithms, have two stages: First, they find *frequent itemsets*, i.e. sets of items that frequently occur in the set of transactions, and then generate rules from them. Apriori employs a technique called *pruning* and *joining*. First, it produces a list of candidate frequent itemsets. Then it uses a concept, named *Minimum Support*, to remove itemsets, which have lower support than the minimum support required. The *support* for an itemset is the number of its occurrences in all the

transactions. Then it joins each itemset in the candidate list with other itemsets to generate 2-length items. In this step, Apriori removes items, which have lower support compared to the minimum support. At that point, the algorithm executes the same process Iteratively, to produce 3-length, 4-length, … itemsets. As a result, this algorithm needs a user-defined parameter, minimum support, to find frequent itemsets.

Next, the algorithm checks each itemset, based on a process named *Pruning*, for each level producing candidate lists. According to this process, each nonempty subset of all items in the candidate list should belong to the previous level's candidate list. If there is an itemset that does not satisfy this condition, the algorithm removes it from the candidate list.

Finally, the last candidate list is the frequent list. Apriori then extracts all non-empty subset of each item and generates rules. In this step, the algorithm applies another user-defined parameter called *Minimum Confidence* to remove weak rules. Confidence for each rule is (S represents the support value):

$$C=S(X \cup Y)/S(X) \quad (3)$$

Since Apriori was proposed, many ARM algorithms emerged to improve its efficiency; most of them ~~. A lot of them~~ were based on Apriori, whilst others followed a different approach to extract rules. We discuss both categories in the following subsections.

Table 1. Apriori Pseudocode

Input: Database, minimum support and confidence

Output: generated frequent rules

C_i=Candidate itemsets with the size of i

F_i=Frequent itemsets with the size of K

While (F_i.lenght() !=0) {

C_{i+1}= the generated candidates from F_i

F_{i+1}= The candidates items in C_{i+1} that their frequency is more than the minimum support.

}

Consider the last not null frequent list and then generate the associated rules.

Check that the confidence of generated rules be more that the minimum confidence value otherwise remove them.

Return the generated rules

Apriori-based ARM approaches

Wang and Tjortjis [27] proposed an algorithm, named PRICES, to improve Apriori. Its main advantage is decreased number of database scans. PRICES scans the database only once. This algorithm uses a concept called price, i.e. a unique value for each item in the database. Then the database is scanned once, and the price of each transaction is calculated. For example, assume that A, C and D are three items with values of 2^4, 2^2 and 2^1, respectively. Hence, the price for transaction I, which consists of {A, B, C}, is 10110 [27]. All prices are stored in a Price Table. Under this assumption, the occurrences of each sort of itemset in all transactions can be calculated, without scanning the database again. Experimental results indicate that, execution time of PRICES is lower than Apriori. It has also better results on larger databases.

Liao proposed another ARM algorithm using an array that consists of a value of 0 or 1 for each item in each transaction [28]. At the same time, support is calculated for each item, thus eliminating items with low support, by scanning the database only once. This approach assigns a weight value to each item, with the help of the user, based on its importance, and calculates support based on these weights. Results illustrate that the approach is faster than Apriori, but has some drawbacks, such as using many predefined parameters and not considering the completeness of the generated rules. It should be noted that when an algorithm is claimed to be faster than Apriori, it should also be shown that it produces complete rules just like Apriori, which is not the case in this paper.

Liang et al. also tried to improve Apriori [29]. First, they read the database and store it in an array. With such an approach, they did not need to deal with the external database, thus reducing time consumption. Then they remove transactions that do not contain any of the k-itemsets, because, such transactions cannot produce (k+1)-itemsets. Thus, many transactions are eliminated, which results in

decreasing execution time. They applied this approach on the physiological indices of patients. Their experimental results indicate that this approach could generate some rules. However, this paper is missing a comparison with Apriori, as well as a discussion about the completeness of generated rules.

Chen et al. proposed a new approach to improve Apriori, named BE-Apriori, addressing the issues of multiple database scans and generation of many candidate frequent itemsets [30]. Two new strategies are employed: first, a pruning strategy, where in each level of k-itemset generation, the algorithm calculates the frequency for each item based on the number of occurrences of that item in all frequent itemsets. For each item, if this number is less than k, each itemset that contains that item is removed. Secondly, a transaction reduction strategy is then employed, by deleting each item, which does not exist in the k-itemset. At that point, for generating the (k+1)-itemsets, they calculate the length of each transaction and eliminate each transaction with length less than k+1. Their results indicate that their approach generates less frequent itemsets compared to Apriori and this improves when increasing the minimum support value. In addition, BE-Apriori has shorter execution time than Apriori.

Yu et al. proposed a new ARM algorithm [31]. They also claim that the biggest drawbacks of Apriori are: a) the large number of Input/Output (I/O) ~~I/O~~ operations and b) the generation of large amounts of candidate frequent itemsets. For solving these problems, they propose a Boolean matrix, which uses "1s" and "0s" to describe the existence of an item in each transaction. They applied just an AND operator to calculate support for each item and store them in the last column of the matrix. They then remove each item, which has lower support than the minimum. An interesting part of their work is that they ran their algorithm in a Hadoop platform executing each part of their method in parallel. This approach scans the dataset only once and runs in a parallel platform, which may result in great reduction of execution time compared to Apriori. However, they did not test their idea on an actual database and just used an example to show the efficiency of their approach instead.

Non Apriori-based Approaches

FP-growth is a very well-known ARM algorithm [14]. Its most interesting innovation is that it generates frequent rules without generating candidate itemsets. It scans the database only twice. This approach employs a structure called Frequent-Pattern tree or FP-tree. In the first scan, FP-growth finds frequent items and in the second, it generates the FP-tree. After producing the FP-tree, the algorithm only needs this tree for further processing, instead of the number and length of frequent patterns. For each transaction, there is a path in the tree, which starts from its first item of its prefix sub-tree. Moreover, items in frequent itemsets are in descending frequency order. Therefore, having more support could help items to be shared by more paths and be at the top of the FP-tree. The main concept in this algorithm is that all itemsets that it produces are in a path of the FP-tree. In other words, all the itemsets in FP-growth are in the database, in contrast to Apriori that may generate candidate itemsets that are not even in the database. FP-growth always follows a pattern-fragment growth method. Having built the FP-tree, FP-growth divides it into smaller trees and then mines them. This is a Divide and Conquer approach, which names each smaller tree a conditional tree. Although FP-growth has many advantages compared to Apriori, it also has a significant drawback: its memory usage, especially when dealing with large databases that could even cause a crash.

Table 2. Psseudocode of FP-growth

Input: Database, minimum support and confidence, FP-tree FT, FPset
Output: generated frequent rules
(1) If (FT has a single path as P) {

Foreach combination of nodes in this path (H)

H∪FPset with support of minimum support of nodes in H

} Else foreach heather in FT as u {

H = u ∪ FPset with the support = u.support

Create the conditional FP-tree of H as CTree

If CTree ≠ ∅

{

Go to the (1) and set the FT=CTree and FPset=H

}

}

With the generated frequent list generate the associated rules.

Check that the confidence of generated rules be more that the minimum confidence value otherwise remove them.

Return the generated rules

Yuan and Ding proposed an improved algorithm based on FP-growth [9]. The authors claim that FP-growth has drawbacks that should be addressed. Three are the main ones: 1- It scans the database twice and requires a sorting process in the first scan, causing performance degradation. 2- It uses a complex process to build the tree. 3- Finding the largest frequent itemset requires a lot of storage and time. Consequently, a new approach based on Huffman transform algorithm, named HSP-growth (Huffman Sequence Pattern-growth) was proposed [9]. It employs a binary structure to store information and a Huffman code platform to describe information. This algorithm employs a Huffman tree for frequent itemsets. Each node (except the leaf nodes) may have two children (left or right) and its value can be 1 which represents existence of an item in itemset otherwise 0. Their approach scans the database only once. Experiments indicate that HSP-growth has shorter execution time compared to Apriori and FP-growth. However, the completeness of generated rules was not discussed.

Narvekar and Seyed proposed a new improvement for FP-growth [7]. At first, it scans the database and generates a tree, named D-tree. This is a simple tree, with a path for each transaction, which also contains the number of occurrences of each node. When the D-tree is built, the algorithm scans it and calculates the support for each item. If an item is frequent, it may occur in more than one nodes. In such a case, the algorithm considers the sum of all the nodes in calculating the support for that item. A Node table consists of the spare nodes. It is claimed that the existence of an item in more than one nodes could have required further processing. Instead, they process each item only once, by considering a condition, which adds to the Node table every node from the current node to the root node that did not occur before. This approach consumes a large amount of memory but scans the database only once. Like FP-growth, it does not generate candidate itemsets. However, in contrast to FP-growth it does not require generating many conditional patterns and conditional FP-trees and generates much less conditional pattern bases. As a result, it is faster than FP-growth. However, its completeness is not discussed.

Heuristic Approaches

To the best of our knowledge, only one attempt has been made to review heuristic ARM algorithms: that by Del et al. [26]. However, this is not a comprehensive study, nor does it compare heuristic approaches. It discusses Genetic Algorithms (GAs) and swarm based approaches. It focuses on some of these approaches, and differentiates between algorithms applied directly for generating rules, and those for optimizing some part of their functionality. For instance, as we discuss in this section, many fuzzy ARM algorithms apply GAs to optimize membership functions [32]. Furthermore, Del et al. also classified ARM algorithms depending on their real world applications and discussed the environment that they were applied on, such as education, manufacturing, computer security etc. [26]. After all, this paper does not cover all the heuristic approaches, it does not critically review heuristic ARM algorithms pointing out their drawbacks and strengths, and it does not compare them with each other. All in all, in this section, we

classify ARM algorithms according to the heuristic algorithm they apply.

There are several heuristic algorithms for ARM. In this section, we review these algorithms categorised in 5 groups: 4 groups based on the algorithms that they use, such as Genetic Algorithms (GAs), Bee Swarm Optimization (BSO), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) and one group based on other heuristic algorithms.

Approachses based on GAs

Genetic algorithms are evolutionary algorithms based on the neural selection process in biological evolution. This algorithm begins with a randomly initialized population and iteratively evaluates each population (called generation) based on a defined fitness function. In each generation, it selects the best individuals and generates the next generation. Yan et al. proposed a novel approach based on GAs for ARM [3]. In their method, they did not use any fixed minimum support threshold. Instead, they employed relative minimum confidence as a fitness function for selecting only the best rules. At the beginning, they proposed an algorithm, named ARMGA, which was designed to deal with Boolean ARM. However, since they also wanted to deal with quantitative Association Rule (AR) discovery, they proposed another GA based method, named EARMGA, which is an expansion of ARMGA [3]. They also designed a FP-tree approach based on FP-growth for implementing EARMGA. Experimental results illustrate that their algorithms reduce computation costs and generate only interesting ARs.

Wang et al. proposed a new ARM algorithm based on GAs, named AGA [33]. AGA is an adaptive GA, which employs a mutation matrix and a crossover matrix for multi-dimensional ARM. For quantitative attributes, they employ a pre-processing step. According to that, for instance, for the age values, they consider intervals such as “0-29” or “30-40”. Moreover, they compared their method with the original GA. Evaluation results indicate that AGA has better performance compared to GA from the point of view of the average *fitness value*, and it generates more rules. In many heuristic algorithms, the fitness value determines the quality of each solution. For instance, it determines the quality of food source in the BSO algorithm. There is a similar case for GAs, PSO etc. As a result, each new algorithm could have its own design of the fitness function, which calculates the fitness value for each solution.

Djenouri et al. studied AGA and they claimed that the main drawback of AGA is that it generates some false rules [4], [34]. They also compared their method with AGA and experimental results indicate that AGA fails in terms of both fitness value and computation time.

Two more heuristic approaches were proposed by Drias based on GAs [8]. One named IARMGA and a Memetic algorithm, named IARMMA. They claim that most of bio-inspired-based algorithms have two main drawbacks: generating false rules and considering some rules with low support and confidence as high quality rules. They considered two parameters to evaluate their approaches and compared them with each other: Execution time and completeness of generated rules. Completeness in this approach has been associated with the value of their fitness function. For this reason, they propose a new method, named “delete and decomposition strategy” to achieve better completeness. Finally, their experimental results indicate that IARMMA has higher execution time compared to IARMGA especially for large datasets. However, IARMMA has better solution quality. In the end, they claimed that their approaches solved the problems of generating false and incomplete rules. One drawback of this work is the lack of comparison with other well-known ARM methods like Apriori.

Another ARM algorithm, based on Parallel GAs (PGA), was presented by Dash et al. [16]. They demonstrate that the problem of Apriori and similar algorithms is the process of setting the minimum support and confidence manually. For instance, in large datasets it

is hard to predict the value of these parameters. They claimed that in most ARM approaches, a user-defined min support and min confidence play the main role in finding interesting rules. In their approach, interestingness is determined by the fitness function, automatically. They also discuss that the combination of the problem mentioned with other issues, such as the large search space, render the application of heuristics popular to many researchers. However, some heuristic methods like GAs suffer from long computation times. As a result, they apply PGA as a solution. Moreover, they denote that the fitness evaluation process for all frequent itemsets is related to the size of the dataset. They use a parallel evaluation system to evaluate the fitness function, which results in decreasing computation time. Their experimental results indicate that their approach has 85% completeness when detecting interesting rules. However, they did not evaluate the computational time of their approach and they did not compare it with other approaches.

Lim et al. proposed a new ARM algorithm, named GA-AssocRule [35]. According to this algorithm if we have n-items that are correlated, then the superset that is associated with n-items, could also be correlated. The correlation value of the superset should minimally be equal to n = Items' correlation. Moreover, they reviewed the algorithm proposed by Yan et al. [3] as well as ARMGA and EARMGA. They demonstrated that the only difference between their approach and EARMGA is in the way they calculate the fitness function. EARMGA sets the fitness function with the help of a concept, named *relative confidence*, which users can set between 1 and -1. Therefore, in GA-AssocRule they use correlation to calculate the fitness function. Their experimental results illustrate that both algorithms generate positive correlated ARs of variable lengths. However, unlike GA-AssocRule, EARMGA cannot generate all rules with all possible lengths. For instance, it produced rules with lengths 2 and 4, but failed to produce rules with length 3.

Another application of GA in ARM is reported by Sagar and Argraval [36]. The authors claimed that most of the current approaches do not consider negative occurrences of attributes. Moreover, they determine a limitation for mined rules: there is no limitation to the number of antecedents, but there should be only one consequent in a rule. Along with support and confidence, they also propose some other parameters like *Complete* and *Comprehensibility* of a rule. Complete is the percentage of coverage that a consequent has on its antecedents. They considered simpler rules as rules with more Comprehensibility. Their experimental results illustrate that although their approach produces desired rules, it can also generate all possible rules.

Li and Yangalso propose a new ARM algorithm based on GAs for Manufacturing Information Systems (MIS) [11]. They claimed that one of the main drawbacks of Apriori based algorithms is that when they increase the number of 1-itemssets, they have more qualifying 2-itemset which may result in decreasing efficiency. They also proposed a new parameter, named *Compatibility*, which was defined by formula (4):

$$COM = C / T \tag{4}$$

where COM is the compatibility value and C is the number of records with a specific gene list and T is the number of whole records.

According to them, at the beginning, a user should specify a minimum Compatibility. After that, the algorithm calculates the Compatibility for each gene list. If the value is greater than the minimum Compatibility, then the algorithm stores it. The main drawback of their approach is that at the beginning, users should set many parameters like Compatibility, mutation-rate (Mutation is the rate of a gene's changes of an individual to create variants and reduce homogeneity), and Cross over rate. Their experimental results illustrate that their approach has lower execution time compared to Apriori. It also indicates that in contrast to the proposed method,

computation time for Apriori increases for larger datasets. In addition, they demonstrated that although GA decreases completeness a little, it could increase efficiency, especially in MIS.

Using GAs for ARM was also proposed by Guo and Zhou [13]. They have improved the method of choosing chromosomes and other parts of GAs. Their experimental results indicate that their approach has lower execution time compared to Apriori. Moreover, they apply a fixed threshold for minimum support and confidence.

Applying GAs or Multi-Objective GAs to extract frequent rules is a popular approach and has been proposed several times [37]. For instance, Bidgoli et al. proposed MOGAR [37], a new ARM algorithm based on multi-objective GA for numerical data [17]. They applied three parameters to evaluate generated rules: Confidence, Interestingness, and Comprehensibility. Moreover, they apply MOGAR to deal with discretizing numeric attributes. They also employ a Pareto based method to automate the process of setting evaluation measures, such as confidence. A rule is considered to be of high Comprehensibility, if the number of items in the antecedent is less than the items in the consequent. In their approach, they employed a list called *Pareto archive* and save all the solutions for all generations in that list. At that point, they select from these solutions. With this approach, they claim that they do not miss any high quality rule. Their experimental results indicate that in most cases their approach generates more rules, with higher confidence. Moreover, since they use a Pareto based approach, it does not need minimum support.

Alim and Ivanov proposed an ARM algorithm based on GA for fuel assembly loading management problems [38]. They considered an aging process, which results in decreasing the size of the problem. Their approach also decreases time consumption. It could find solution for loading problem in 176^{th} generations when classic GAs would obtain it in 250^{th} generations.

Another ARM method based on GAs was proposed by Du et al. [39]. First, it applies an improved version of Apriori to extract frequent rules. The authors assume that each transaction which does not have i-length itemsets, it neither has (i+1)-length itemsets. As a result, they remove this transaction. In addition, they store the database in a table in memory to avoid accessing the external memory for future references. Finally, with the help of a GA, they select some of these rules as the best possible generated rules. They do not discuss the effects of their approach on execution time, completeness or memory consumption. In addition, there is no comparison between their approach and other approaches like Apriori.

QuantMiner is another GA based approach focusing on extracting Quantitative ARs [40]. It requires some predefined values like minimum support and confidence, the number of generations etc. A user can decide which item is more desired and could set a template for generated rules. They defined a rule template as “pre-set format of a quantitative association rule”. For instance, he or she could decide that the selected item should be on the consequent or the antecedent of the generated rules. They claim that their approach is 10 to 20 times faster than GAs. However, this paper did not consider many subjects. It does not give any actual specific results to readers to decide whether this approach is fast or not. Neither does it compare its approach with other algorithms and does not consider other parameters such as completeness.

Mata et al. proposed another ARM approach, which only focused on one aspect of ARM, like Du et al. [39] and Mukhopadhyay et al. [25] and was not compared with other approaches [41]. They applied GAs to extract numeric ARs. Mata et al. [42] also employed a fitness function as formula 5:

$$F = C - (M * PF) \quad (5)$$

where F is the fitness value and C is the number of records that match the rule and M indicates whether this record has been covered by another rule or not. In addition, PF is a user-defined parameter that sets the weight of the marked parameter. This approach is also not compared with other algorithms.

Ghosh and Nath considered ARM as a multi-objective problem [43]. In addition to the support and confidence, they apply two other parameters, named Comprehensibility and Interestingness. They claim that if the number of conditions in the antecedent was smaller rules would have higher Comprehensibility. They also employ formula (6) to determine the *Comprehensibility* of a rule (assuming a rule: A→C).

$$\text{Comprehensibility} = \log (1 + |C|) / \log (1 + (AUC)) \quad (6)$$

where |C| is the number of items in the consequent and (AUC) is the number of items in both antecedent and consequent. They also claim that finding rules with more frequency in the database may be more interesting for users. In addition, they employed an equation to determine the *interestingness* of a rule:

$$IN = [S(AUC)/S(A)] * [S(AUC)/S(A)] * [1-(S(AUC)/|D|)] \quad (7)$$

where IN represents the interestingness value and S is the support value and |D| is the total number of transactions. They also applied a Pareto based structure to extract interesting rules from the database. This approach was not compared with other algorithms.

Puig et al. proposed CSar, an ARM approach based on GAs [44]. CSar was designed for extracting rules that contain both quantitative and categorical attributes. It uses classifiers based on seven parameters: i) support, ii) confidence, iii) fitness, iv) experience, which represents the number of times that the antecedent of a rule matches the inputs, v) the Consequent matching sum, which indicates the number of times that all parts of the rule match the input, vi) numerosity, which is the number of copies of a classifier in the problem domain and vii) tcreate, which is the time taken to create the classifier. CSar classifies generated rules based on two methods. The first is classification based on the antecedent. This mechanism considers two rules with similar antecedents as similar. The second method also considers two rules with similar consequent as similar. If two rules have the same antecedent or consequent, they can be merged. At that point, they also remove rules that express the same knowledge. Their experimental results indicate that CSar could generate rules that Apriori also generates, which is a sign of completeness.

Kaya and Alhaji [45], Hong et al. [32], and Wang and Bridges [46], proposed GA based approaches to optimize membership functions. This process is a mandatory stage in extracting weighted ARs especially in the quantitative database. Moreover, Kaya and Alhaji claim that some rules may mislead users [45]. Consider a rule A→ B with support 50% and confidence 66.7%. If support for A is 75%, this is a weak rule, because there is a negative connection between A and B. In other words, a customer who wants to buy A may have a smaller chance to buy B compared to an anonymous customer that we have no prior information about. As a result, they consider a rule *interesting*, if the value of Support(X, Y)/ Support(X), Support(Y) is greater than 1. There is a similar approach by Hong et al. who tried to employ GAs for adjusting and optimizing membership functions in fuzzy ARM algorithms [32]. In their next paper, Chen et al. also focused on extracting frequent rules from quantitative databases [47]. This algorithm is a cluster-based fuzzy-GA approach that defines membership functions. For clustering chromosomes, they apply K-means clustering. Their experimental results indicate that their new approach has better performance compared to the previous one. Although their approach could find large itemsets, it also could find convenient values for membership functions. This approach also has good execution time and completeness compared to their previous approach. Interestingly, in their next paper, Chen et al. improved

their approach, and applied a multiple support approach GFMMS, instead of using single minimum support [48]. Experimental results indicate that the new approach has better fitness value compared to the previous ones. In their more recent approach, Chen et al. proposed yet a new method, named FCGFMMS [49]. This method includes two phases: one is finding the minimum support and membership functions and the other involves extracting frequent fuzzy ARs. Their experimental results indicate that FCGFMMS is ten times faster than GFMMS and has a better average fitness value. Finally, Chen et al. proposed DGFMMS, an improvement over their previous approaches [50]. This approach also employs a divide and conquer policy. The experimental results indicate that DGFMMS has a higher fitness value compared to GFMMS [48].

Thilagam and Ahanthanarayana applied a multi-objective GA for fuzzy ARM [51]. Fuzzy support, fuzzy confidence, and rule length were the objectives of this approach. However, in contrast to Kaya and Alhaji [45], Hong et al. [32], and Wang and Bridges [46], they used K-mean clustering for finding the optimal values for membership functions. Finally, Wang et al. also proposed an ARM approach based on GAs and fuzzy ARM to manage networked manufacturing resources [52]. The main difference of this approach is that they employed a knowledge library and store the generated rules in it. They compare each new generated rule with this library and if it is redundant or meaningless, the algorithm removes it. However, they did not discuss the definition of meaningful or meaningless for a rule. In all of these ARM approaches authors did not consider parameters like execution time, memory usage, completeness etc. They also did not compare them with other ARM approaches.

FGBRMA is an approach, employing GAs for determining the minimum fuzzy support and membership functions that was proposed by Hu [53]. Its main goal is to generate fuzzy classification rules. Experimental results indicate that their approach has higher classification rate compared to other classification methods even that in [54]. Ishibuchi and Yammoto also proposed a method based on multi-objective GAs for pattern classification problems [54]. In [55], Fdez et al. proposed a new fuzzy ARM algorithm for mining quantitative data. Their approach is based on GAs. They claim that setting membership functions is not an easy task. Hence, their method determines the membership functions and mines fuzzy ARs. At the end, they compared their approach with Hong et al. [32] and experimental results indicate that their approach has better fitness value and produces larger 1-itemset rules. Finally, it is also faster than Hong et al. [32].

Shenoy et al. proposed another ARM approach suitable for dynamic databases, based on GAs [56]. They claim that the main issues in dynamic ARM algorithms are Additions and Deletions; Additions increase support of itemsets because of their occurrence in the updated data and Deletions decrease supports of itemset because of the lack of their existence in the updated data. This process could remove some of the generated rules. Interestingly, their approach also focuses on distributed databases. Moreover, they considered two novel terms: Intra-transactions and inter-transactions. The former refers to finding frequent rules that represent relationships between items in a transaction. The latter refers to finding relationships between items in transaction A and items in a transaction B, which is more complicated than intra-transaction. As a result, their goal was to find large inter- transaction rules and scan the database only once with the help of Distributed and Dynamic Mining of ARs using GA (DMARG) [56]. Their experimental results indicate that their approach is faster than an improved version of Apriori and Fast UPdate (FUP) [56].

Guillet et al. proposed an ARM approach, which applies graph visualization modelling to decrease the number of generated rules [57], as generating many rules is claimed to “hide” the most interesting rules.

They measured the quality of their approach's drawings with a fitness function. This value is based on the number of edge crossings and the total length of arcs. Their experimental results indicate that their approach is more robust to change, but from the point of view of quality of obtained drawings, dot, a well-known static approach, has better performance.

Kaya and Alhajj [58], proposed another ARM algorithm, which employs GAs. They claim that finding domain of quantitative attributes for Fuzzy ARM is a hard task. As a result, it should be done either by an expert or automatically. However, they claim that even an expert cannot do this task properly. Hence, they applied a clustering approach based on GAs. They also employed Clustering Using REpresentatives (CURE) for setting membership functions. Their experimental results indicate that their approach produces more interesting rules compared to previous approaches. However, a drawback of their work is that it has longer execution time compared to other methods.

Alatas and Akin proposed a new automatic ARM approach for mining positive and negative ARMs based on GAs [59]. They claim that a non-automatic approach uses a trial and error method and runs several times to get the best fitness score; however, in this work they automatically set the minimum support and confidence. Their experimental results indicate that this approach generates rules with high support and confidence. They also compared their approach with GAR [41], which only extracts positive itemsets. Results illustrate that GAR finds rules with higher support in most of the cases. However, the number of items in generated rules is less than these in rules generated by GAR.

Puing et al. proposed a fuzzy ARM algorithm, named Fuzzy-CSar, based on GAs [61]. Fuzzy-CSar includes classifiers that consist of fuzzy AR. These classifiers have 6 parameters for checking the quality of rules: support, confidence, fitness value, experience (the number of cases that the input was matched with rule antecedents), numerosity (which indicates the number of copies of the classifier among all classifiers) and the average size of association sets. Khabzaoui et al. also propose an ARM algorithm for mining DNA microarray databases based on GAs [62]. They employed eleven parameters to evaluate generated rules. Some of these parameters are support, confidence, interest, conviction etc. Kaya proposed another approach based on Multi-objective GA for fuzzy ARM [63]. The objectives of this algorithm are Strength, Interestingness, and Comprehensibility. Strong rules have support and confidence above the minimum. Interestingness represents how many of the generated rules are interesting and finally comprehensibility indicates the number of attributes that participate to the rules. In addition, Anandhavalli et al. put forward another ARM approach based on GAs, named ARG [64]. This approach, firstly, generates frequent rules and then, based on GAs and some parameters like Positive Confidence and Interestingness, selects the most interesting rules. Their experimental results indicate that their approach produces less rules compared to previous approaches and is faster.

Martın et al. [87] proposed a multi-objective evolutionary algorithm, called MOPNAR, and their focus was on reducing the computational cost and the size of mined AR (either positive or negative quantitative rules). The three objectives that MOPNAR tries to maximize are comprehensibility, interestingness, and performance "in order to obtain rules that are interesting, easy to understand, and provide good coverage of the dataset" [87]. Furthermore, Palacios et al. [88] proposed another GA based ARM approach, named FARLAT-LQD. They employed a 3-tuples linguistic representation model to decrease their search space. They also proposed a model, named FFP-growth-LQD, based on the Fuzzy Frequent Pattern-growth algorithm for the fuzzy ARM [88].

Padillo et al. [89], proposed a new ARM approach based on GA for Big Data ARMs. Their focus was on avoiding increasing the complexity of the solutions [89]. This approach also employs a grammar to reduce the search space and uses subjective knowledge [89]. Moreover, Martin et al. [90] proposed another GA based ARM approach called NICGAR, which has a low runtime. This approach contains two thresholds that enable the user to balance the quality and diversity of extracted rules [90]. Finally, this approach can determine the similarity degree between the rules. In addition, in another study, Qodmanan et al. [92] proposed a dynamic multi-objective ARM approach that can set the minimum support value automatically.

Approaches based on Bee Swarm Optimization (BSO)

BSO is one of the swarm optimization approaches that is based on the life of bees. It includes different kinds of bees that have different responsibilities. These bees should randomly explore the research problem space to find the possible food sources (solutions). The quality of each food source can be evaluated by its distance from the hive. Research in this area was conducted by Djenouri et al., who proposed a new method for Web ARM [34]. At the beginning, they proposed an improved GA-based ARM algorithm and compared it with previous GA-based ARMs. The paper discussed the drawbacks of previous methods, such as ARMGA, and classified them into two groups, methods that: i) generate rules that do not respect the minimum support and confidence, and ii) generate unacceptable rules. As a result, first, they proposed an ARM method based on bees' behaviours in real world (BSO), named BSO-ARM. The experimental results indicate that from the point of view of the fitness value, BSO-ARM is the best. However, the main drawback of this approach is its computation time, which is higher than others. In other words, although BSO-ARM is a time consuming approach, it produces rules that are more complete compared to previous GA-based approaches.

It is interesting that, in their next paper Djenouri et al. proposed a new ARM algorithm, HBSO-TS, based on two meta-heuristic algorithms: BSO and Tabu Search (TS) [4]. Tabu Search can be applied in optimization problems. In their previous paper, they proposed BSO-ARM [34], but in this paper, they improved their approach. They applied TS for detecting the best neighbour. The experimental results show that HBSO-TS has better fitness value than ARMGA and BSO-ARM. Moreover, the proposed method also has lower computational time than BSO-ARM. Finally, in contrast to ARMGA, their method does not generate any false rules. However, one of its biggest drawbacks is that when tested against a large dataset, it bluntly blocked [34]. In addition, as authors mentioned, since they applied two heuristic algorithms, many parameters are required to be set.

Applying a heuristic approach also was the case in [65] where Djenouri et al. proposed a new ARM policy based on bees' behaviour and Graphics Processing Unit (GPU), named PMES**.** It is a parallel version of BSO-ARM. Their goal was to tackle problems of ARM in large databases. First, it searches for new solutions using the Central Processing Unit (CPU) ~~CPU~~ and then it evaluates each solution by GPU threaded in parallel. Interestingly, PMES is based on a Master/Slave structure. In this paradigm, the algorithm finds the position of all bees by executing the CPU. At that point, with the help of GPU, it calculates the fitness value of all solutions at the same time and in parallel. The authors compared their approach with BSO-ARM, i.e. their previous work. Their experimental results indicate that PMES is faster than BSO-ARM. Moreover, running these two algorithms in large databases like Web Docs had interesting outcomes: BSO-ARM was blocked in 12 days, but PMES only ran for 10 hours and was not blocked.

Approaches based on Particle Swarm Optimization (PSO)

PSO algorithm is based on the behaviour of bird flocking, where each bird is a single solution and has a fitness value. Based on this algorithm, the birds know

that food sources exist, but they do not know where exactly these sources are; each bird tries to find the best food source. Kuoa et al. proposed a new approach based on PSO [15]. Their goal was to determine the minimum support and confidence automatically in a complete way. They changed the type of database to Binary format, and with the help of PSO, they extracted the rules. In the end, the best particle was found. At that point, the support and confidence of the best particle was considered as the minimum support and confidence. To evaluate their approach, they compared their method with GAs. The experimental results indicate that PSO has lower computation time. In addition, since PSO sets the minimum support and confidence automatically, there is no need to set them by a trial and error approach, which is time consuming. They also applied this method on investors' stock purchases behaviour in the real world.

There are more cases of researchers who applied two heuristic algorithms at the same time. Vyas and Chauhan proposed two heuristic approaches for ARM [66]. One of them was based on PSO and the other was based on GAs. Their experimental results indicated that PSO has lower average support and higher average confidence. GA has higher average fitness value compared to PSO [66], but Kuoa et al. claimed that PSO extracts better rules compared to GAs [15]. Moreover, according to Kuoa et al. [15], Vyas, and Chauhan [66], PSO is faster whether it uses a fixed threshold for the minimum support and confidence, or not. Kuoa et al. also claimed that PSO extracts better rules compared to GAs [15].

Another use of PSO was proposed by Nandhini et al. [18]. Their ARM method includes two steps. First, it transforms the data to binary format. Then, it applies PSO to define the minimum support and confidence, automatically. They considered support and confidence of the best particle as the minimum support and confidence. At that point, their approach mines ARs and extracts the rules. At the final stage, it uses a post mining technique, named Domain Ontology to reduce the number of rules. The algorithm also employs a rule schema, which defines user expectation of mined rules. User expectation is a predefinition that the user has extracted rules following these definitions; like the forms of the consequent part or the antecedent part of generated rules. Finally, they compared their proposed method with Apriori. Their approach generates about 1000 rules without having any knowledge about user expectations. Then, by applying a domain ontology and taking user expectations into account, it faced a dramatic decrease in the number of extracted rules. As a result, their approach produces less rules without any negative impact on the interestingness of rules. However, they did not investigate the computation time of their approach.

Alatas and Akin proposed an ARM approach based on PSO, named Rough PSP (RPSOA) [67]. It does not need any predefined parameters like minimum support and confidence and tries to generate rules, automatically. Because of this feature, some of their test results even indicate that if they use a fix minimum support, they could not generate any rule from the database. This contrasts with their approach that sets minimum support automatically and generates complete rules. Their experimental results indicate that their approach has good performance on different databases even with noisy data. They also compared their approach with other approaches like GAs and the results illustrate that their approach generates rules with higher support that contain less items in most of the cases.

Approaches based on Ant Colony Optimization (ACO)

ACO is a metaheuristic optimization algorithm that can be used to find the best path in a graph. This algorithm is based on the real-world ants' lifestyle. Applying Ant Colony Algorithm (ACO) is also one of the popular solutions for ARM. Kau and Shih proposed a new ARM algorithm based on a meta-heuristic

algorithm, named Ant Colony System (ACS) [12]. ACS is based on the behaviour of ants. However, in this algorithm, ants have memory and they are not blind. In addition, they claimed that ACS needs some predefined parameters. As a result, they proposed a method that uses some limitations and defines most of these parameters before running the model and decrease computational time by scanning the database only once. The experimental results indicate that proposed method takes less computational time compared to Apriori. However, many similar rules are generated, which is an important challenge that should be addressed.

ACO was the main solution for another approach by Hong-yun et al., named SRMining [68] for finding longer rule-chains. First, it creates a graph, named PAGraph, which contains directions to most potential ARs. Their algorithm assumes that an edge of the PAGraph could be part of the rule chain. In this step, with the help of some heuristics and feedback from the ants, some paths can be considered as potential longer rule chains. In their experiments, they compared their approach with FP-growth and DLG. Results indicate that SRMining is faster than others, maybe because it only scans the database once.

There is another use of ACO by Zhu et al. [69]. In this paper, they proposed a new AR decision algorithm based on ACO for a Ball Mill Pulverizing system. In this system, a set of variables for controlling the system should be determined. However, these variables depend on the environment. As a result, defining suitable values for them is an open challenge. Previous works indicated that other approaches like Neural Networks, GAs, and Fuzzy optimization algorithms do not satisfy completely user expectations. Experimental results indicate that their approach finds these values quickly and completely.

In [70], Y. He and S. Hui proposed two methods, named Ant-C [70], which is an ACO based clustering approach, and Ant-ARM [70] that is an ACO based ARM approach. Their experimental results indicate that Ant-C has better performance and higher completeness compared to other well-known clustering algorithms like K-means. Moreover, Ant-ARM is a fast approach. Its execution time was reported to be 24.5% compared to that of FP-growth and 0.2% compared to that of Apriori, when executed on the same machine.

Finally, ACO programming was also found to be a good technique for ARM by Olmo et al. [93], who proposed two algorithms guided by a context-free grammar. One is called Grammar-Based Ant Programming for Association Rule Mining (GBAP-ARM) following a single-objective approach using a novel fitness function, which measures the weighted average between support and confidence, to evaluate the individuals mined. The other is called Multi-Objective Grammar-Based Ant Programming for Association Rule Mining (MOGBAP-ARM), considering individual evaluation from a Pareto-based point of view, measuring the confidence and support of the rules mined and assigning them a ranking fitness. They were both tested over 15 data sets from the UCI repository [94], and their results were compared to other ARM algorithms, including Apriori [1] and FP-growth [14], ARMGA [3], as well as other genetic programming algorithms. The results obtained were very promising in terms of reliability and instance coverage, as well as memory, although no details are disclosed w.r.t. to memory usage, execution time, or interestingness.

Other Approaches

In our recent work, we proposed ARMICA [71], focusing on automation, speed and completeness. We employed the Imperialism Competitive Algorithm (ICA), which is a fast heuristic approach. Our objective was to extract rules automatically, without any predefined minimum support and confidence. Early experimental results indicated that ARMICA is faster than Apriori and generates all the rules that Apriori produces, thus, it is complete. Finally, further comparisons illustrate that ARMICA is faster than FP-

growth, so it can be considered a fast approach. However, ARMICA has one drawback: it requires a predefined parameter, named: *Number of Imperialists*. Although it is much easier to set this parameter than the minimum support and confidence, it is desirable that ARMICA defines this parameter automatically, too. ARMICA should be further compared with more recent ARM approaches rather than just Apriori and FP-growth. As well as consider more metrics, such as memory usage, the number of database scans or interestingness.

Another heuristic based ARM approach is MDS-H, which have been presented in Hong and Bian [10]. It is based on the Multi-Dimensional Scaling (MDS) [10] approach and tries to improve Apriori in terms of performance. MDS-H consists of two steps: grouping and joining. During the grouping step if an itemset has some items from the same group, the algorithm considers it as non-frequent. A result of non-frequent items elimination is avoiding large numbers of frequent itemsets. Their approach was tested in an urban transportation network. The experimental results indicate that MDS-H has low computation time and it is complete in long pattern discovery. They compared their method with Apriori and found that MDS-H generates less itemsets and has very lower computation time compared to Apriori.

Jorio et al. proposed a new heuristic approach for mining gradual itemsets, which are sets of gradual items [72]. In this paper, there are two kinds of gradual items: value increases and value decreases. They considered the gradual rule as a higher/lower structure. For instance, they considered this rule as a gradual rule: “the higher the age, the higher the pay” [72]. Moreover, they claim that fuzzy based approaches may have some problems like losing some information or may not be suitable for the gradual pattern mining problem. In addition, gradual pattern mining causes a competition between items. It could be a two-by-two comparison, but they claim that it is a memory consuming process and does not generate gradual patterns. As a result, they make an ordered dataset. Since their method is based on Apriori, they needed to have a pruning process. However, they claim that it is not applicable to their approach. Hence, from level two they order objects and keep this order until the end. The main drawback of this approach is its long execution time. However, they did not compare their approach with other methods. They also mentioned that their approach may not generate all possible rules.

Hu and Li claimed that ARM is not a single objective problem, so they considered it as a multi-objective problem and proposed a new evolutionary algorithm [73]. They considered three parameters as their objectives: Statistical Correlation, Comprehensibility, and Confidence. They defined Statistical Correlation as formula 8:

$$SC\ (X\ U\ Y) = \frac{|D| * S(X\ U\ Y) - |D| * \prod_{(i \in X \cup Y)} S(i)}{\sqrt{|D| * \prod_{(i \in X \cup Y)} S(i) * (1 - \prod_{(i \in X \cup Y)} S(i))}} \quad (8)$$

where S indicates the support value and the rule is X→Y and $|D|$ is the number of records of the database. This parameter eliminates the rules that do not have any relativity. They also claim rules with less items in the consequent or the antecedent are more comprehensive. Their experimental results indicate that execution time for this approach was 54,2% shorter than that of FP-growth. Their approach also generates fewer rules.

Alatas et al. proposed another ARM approach [74]. A multi-objective evolutionary algorithm that is a Pareto based approach, named MODENAR (Multi-Objective Differential Evolution Algorithm). Because this approach is automatic and does not need any minimum support and confidence, they call it, database independent. They also compared their approach with these in Alatas and Akin [59] and GAR [41]. Experimental results indicated that MODENAR generates rules with higher support and confidence even with noisy databases. It also generates less rules

compared to GAR [41]. There are less items in generated rules by MODENAR than by GAR [41]. Finally, there are more database records covered by the generated rules by MODENAR, than these by GAR [41] and by Alatas and Akin [59]. Lastly, Cano et al. [91] proposed a new methodology based on Graphics Processing Units (GPUs) to evaluate ARs. Any evolutionary approach can employ this approach which enables them to evaluate ARs in parallel, resulting in decreasing computation time.

Finally, we could briefly mention here approaches based on Grammar-Guided Genetic Programming (G3P). G3P can restrict the search space and build rules conforming to a given context-free grammar. For instance, an ARM algorithm called Grammar-Guided Genetic Programming for ARM (G3PARM) makes the knowledge extracted more expressive and flexible, by allowing a context-free grammar to be adapted to each problem domain and discretization [95]. It keeps the best individuals and obtains solutions within specified time limits and does not require large amounts of memory. It is compared to exhaustive search (Apriori and FP-Growth) and GAs (QuantMiner and ARMGA) algorithms and rules are analysed whilst its scalability is verified. Further work that deals with ARM under a multi-objective perspective using G3P models, that enable the extraction of both numerical and nominal ARs in a single step was presented in [96]. Experimental results indicate that multi-objective proposals obtain very frequent and reliable rules when attaining the optimal trade-off between support and confidence. Furthermore, for the trade-off between support and lift, the multi-objective proposals also produce very interesting and representative rules. Lastly, a G3P algorithm that does not need many parameters and enables the discovery of quantitative ARs comprising small-size gaps was presented in [97]. The algorithm was tested over a varied set of data, comparing the results to other ARM algorithms, showing that it can reduce gaps in numerical features.

Comparison Performance Metrics

In this section, we discuss the main criteria that are important in ARM algorithms. Based on the reviewed papers, we selected 8 performance metrics to compare all the previous researches. They can be found in Table 3. In this table, the coloured cells indicate the best approaches from the point of view of that performance metric; the cells that contain the term "considered" represent the fact that the approach is not the best approach from the point of view of that performance metric, but it did consider this performance metric in its evaluation. We discuss this further in section 5. Then, based on the frequency of these performance metrics in all the approaches, a priority has been allocated to them, which can represent the importance of each performance metric. Moreover, based on the priority of each performance metric, we gave them a weight value. Since we have 7 priorities (1 to 7), we give to the performance metrics a weight from 1 to 7. For instance, Execution Time and Interestingness have priority 1, so we gave them weight 7. Finally, we propose a formula to calculate the *GT-Rank* for each performance metric. *GT-Rank* can be used to rank each performance metric and based on this rank, the best ARM approach could be selected:

$$GT - R(\mathrm{A}) = \sum_{n=1}^{8} B \times \left(a_{\mathrm{param}} \times \mathrm{W}\right) \quad (9)$$

where A is any given ARM approach and R is its rank, and a_{param} is factor set to 1 if approach A considers the corresponding performance metric n or not, and to 0 otherwise. W is the weight for each performance metric, as shown in Table 3. Finally, if an algorithm is the best approach from the point of view of that performance metric (if it is coloured), then B is set to 2 and if the approach is not the best approach in that performance metric but does consider it, B is set to 1. The main reason why the *GT- Rank* is an important factor is that right now we do not have any efficient ranking system to compare the heuristic ARMs, which can confuse researchers about their future directions, as well as practitioners with regards to selecting the

most suitable ARM solution. Having an efficient rating system that was generated based on more than 30 papers in the literature, can show to researchers which of the previous approaches are better qualified and which need further improvements. *GT- Rank* is based on the previous trends in heuristic ARM approaches and based on the frequency of the evaluation performance metrics in previous approaches. It determines which performance metric is more valuable to real world applications.

However, it is noticeable that, there is not any "best" term for performance metrics like database scans, interestingness, parallel processing, and automatic procedure. For these performance metrics, we just considered all the approaches that apply to them, as the best approaches and colour their cells. As a result, they should not have any extra impact on the *GT-Rank*s. Therefore, their B value would be 1. In most of researches on ARM, there are some common experimental performance metrics.

We divided these 8 performance metrics in two groups based on their *P*riority values. performance metrics that are mandatory and other additional features. On one hand, mandatory performance metrics include Execution Time, Interestingness, Automatic Procedure, and Completeness. On the other hand, additional features are Itemsets/Rules Reduction, Database Scans, Parallel Processing, and Memory Usage.

Mandatory Performance Metric

Some performance metrics are necessary for any ARM algorithm. Without satisfying these performance metrics, ARM approaches could face problems.

Execution Time

One of the most important features of an ARM approach is its execution time. Most approaches focused on improving Apriori-like algorithms and extract the frequent rules faster. Many algorithms have been proposed after Apriori and they have succeeded to decrease execution time, but there should be further improvements in this area.

Table III. Final Comparison: The best algorithm for each performance metric.

Algorithm	Memory Usage	Execution Time	Completeness	Database scans	Parallel processing	Automatic Procedure	Interestingness	Rule/Itemsets Reduction	*GT-Rank*
ARMGA[3]		considers						considers	27
ACS [12]		considers	considers						25
Narvekar [7]	considers	considers							20
Kuoa[15]									20
SRmining[68]									18
Dash[16]			considers						24
Ant-ARM[70]			considers						20
ARMICA[71]		considers	considers						19
BSO-ARM[34]		considers	considers						19
Pathak[19]	considers	considers							20
Nandhini[18]								considers	14
HSBO-TS[4]		considers							19
PMES[65]									16
MASP[5]		considers							17
MDS-H[10]									14
KAYA2[58]									16

MOGAR[17]									16
Fdez [55]		considers	considers						13
Chen [47]		considers	considers						13
FCGFMMS [49]		considers	considers						13
RPSOA[67]			considers						12
Alatas[59]								considers	10
MODENAR [74]								considers	10
QuantMiner [40]									10
Ghosh[43]									10
KAYA[45]									10
Hong [32]									10
Thilagam[51]									10
Wang [46]									10
KAYA3[63]									10
GFMMS[48]			considers						6
DGFMMS [50]			considers						6
MOPNAR [87]									32
FARLAT-LQD[88]		considers						considers	21
Padillo [89]		considers							17
NICGAR [90]									32
Qodmanan [92]									6
Cano [91]		considers						considers	11
GBAP-ARM[93]			considers					considers	10
MOGBAP-ARM [93]			considers					considers	10
G3PARM [95]			considers						28
NSGA-G3PARM [96]			considers					considers	10
SPEA-G3PARM [96]			considers					considers	10
GGGPA [97]			considers					considers	10
Frequency of Parameters	2	22	19	3	2	11	17	16	
Priority	7	1	2	6	7	5	3	4	
Weight	1	7	6	2	1	3	5	4	

Interestingness

Most of users in real world applications do not look for obvious rules; they look for relations which may be hidden. Therefore, they look for patterns that are not already known. In this situation, researchers may propose some new approaches to extract rules that are more interesting. It is noticeable that this performance metric could have different definitions in different approaches and researchers employ varied mechanisms to define interestingness in their ARM algorithms.

Automatic Procedure

One performance metric that may have positive impact on efficiency of an ARM algorithm is employing a mechanism to extract frequent rules, automatically. Many researchers considered it to improve efficiency of their approaches. As a result, their approaches do

not require predefined values like minimum support or confidence. Their approaches set these values automatically. In addition, they do not need to have any knowledge about the database in advance and could be applied in any database.

Completeness

Although users are looking for approaches that extract frequent rules and patterns in a short period of time, they also want to ensure that these results are complete. Hence, extracting false frequent rules in a short fraction of time is not acceptable in most of the cases. Consequently, most researchers check their results from the point of view of completeness. To calculate completeness of an algorithm, we could test each generated rule with the transactions in the database and look for the number of matches or mismatches. We also could compare our methods with Apriori or FP-growth from the point of view of generated rules.

Additional Features

There are other performance metrics which may not have been considered necessary by researchers. However, it is important to consider them since they may have an impact even on mandatory performance metrics and affect their performance.

Number of database scans

A factor, which may have a great impact on execution time, is the number of database scans. Based on this number, ARM approaches may have long execution time if they have many database scans and vice versa. This is because these approaches need many I/O operations increasing execution time. As a result, many algorithms have been proposed with only one database scan. After this scan, they store the necessary information in the main memory for future reference, which is faster than having to access the disk, using I/O operations.

Parallel processing

Employing a parallel structure could make any ARM approach faster. According to the parallel structure, all stages of extracting frequent rules could be done at the same time or at least in short intervals. As a result, this could decline execution time. However, like any other mechanism, using a parallel mechanism could have its own weaknesses. It may increase some other performance metrics like memory usage. Peris et al. [75] noted that the memory overhead of parallel processing could have a negative impact on system performance [10]. One of the solutions that they proposed is employing of a memory aware resource allocation policy. There are many other drawbacks when applying a parallel mechanism and all should be considered in ARM.

Number of generated itemsets or rules

Whenever an ARM approach generates many itemsets, it may need long time to process them. The best known example is Apriori. This approach generates itemsets many of which may not even be frequent. As a result, it requires extra effort and time to eliminate non-frequent ones. Moreover, there are many cases that producing all possible rules is not even required. Therefore, researchers prefer the most important rules. As it is clear, this performance metric has influence on execution time. In addition, it may have positive impact on memory usage, too.

Memory Usage

Another important performance metric in ARM algorithms is memory usage. Most of the fast algorithms like FP-growth focused only on improving execution time and did not consider memory usage. This could have negative impact on ARM algorithms.

Comparison

In this section, we compare current heuristic ARM algorithms, based on well-known evaluation performance metrics. Most of these approaches may have strengths regarding some of these performance metrics. Therefore, in order to find out the real advantages of each approach, we do the comparisons in eight sections, separately for each performance metric. We use this process, to establish the best approaches for each performance metric.

Mandatory Performance Metric

We selected four performance metrics as mandatory, based on their frequency on previous ARM approaches. They have higher priority compared to the additional performance metrics.

Execution time

Without doubt, proposing a faster algorithm was the most important goal for many ARM algorithms. Its importance is expected to increase in the future, especially when trying to analyse large datasets. Many heuristic approaches tried to improve previous ARM algorithms like Apriori, with extraction of frequent rules in a shorter period. In this paper, we compare these approaches in two subsections. First, we compare all the heuristic approaches that were compared with Apriori by their authors. Then, we evaluate all the heuristic approaches in the literature which were originally compared with other heuristic approaches, in the respective articles found in the literature.

Many heuristic approaches were compared with Apriori and in some cases even with FP-growth. Fig. 1 illustrates the results of these comparisons. In many of these papers, the authors benchmarked their approaches against Apriori on different databases. For the sake of simplicity, we calculate the average value of these results and mention them in Fig. 1. Moreover, ACS [12] runs in one hour and 45 minutes compared to Apriori that runs in eight hours and 5 minutes.

According to Fig. 1 and the above information, NICGAR [90] is 575 times, MONPNAR [87] is 533 times, G3PARM [95] is 23 times, and MDS-H [10] is about 9 times faster than Apriori. After that, ASC [12] is approximately 4 times faster than Apriori. Then, ARMICA [71], Li and Yang [11], Pathak et al. [19], and Improved GA [13] are in the next steps, respectively.

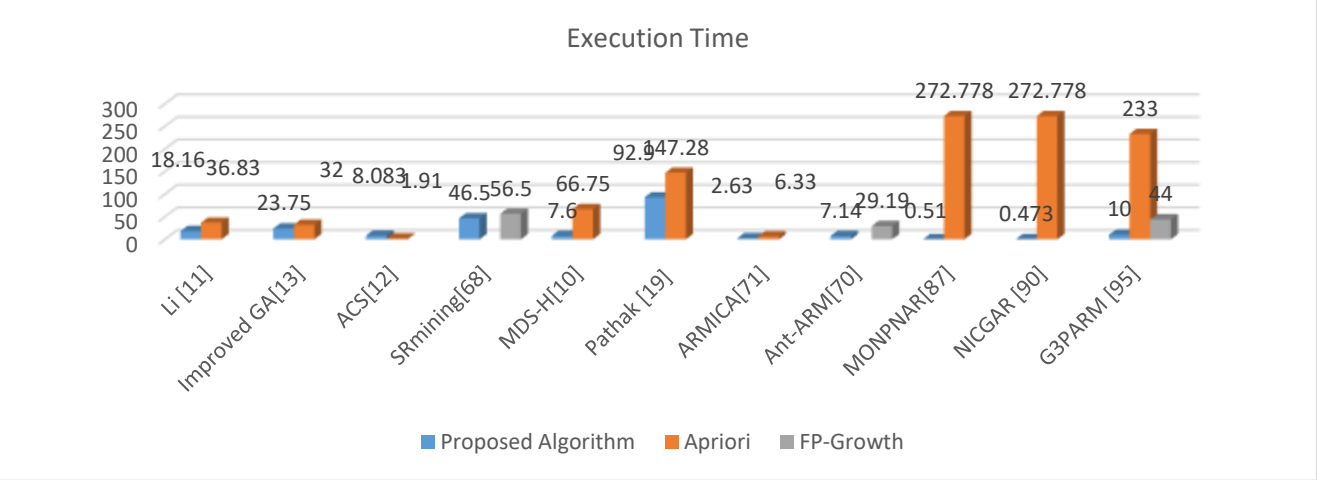


Figure 1. Comparison of Execution time with Apriori and FP-growth

Figure 2. First Execution time Comparison

However, SRmining [68] and Ant-ARM [70] are even faster than FP-growth. Since FP-growth is much faster than Apriori, these algorithms could be among the fastest ones. In conclusion, according to Fig. 1, NICGAR [90], MONPNAR [87], G3PARM [95], MDS-H [10], SRmining [68], and Ant-ARM [70] are the fastest heuristic ARM approaches, which were compared with Apriori and FP-growth.

However, there are other approaches that compared themselves with other heuristic algorithms, instead of Apriori and FP-growth. For the sake of simplicity, we calculated the average value of these results and mentioned them in Fig. 2. According to Fig. 2, HSBO-TS [4] has the highest execution time compared to others. BSO-ARM [34], which is previous version of HSBO-TS, has the second highest execution time. Moreover, ARMGA [3], AGA [33], and IARMGA [8] have the next places. Finally, PMES [65] has the shortest execution time compared to others. This could indicate that heuristic approaches based on the BSO algorithm may be faster than approaches based on GA.

Comparison of classic GA based ARM algorithms and PSO based ARM algorithms was also another field of study. As Fig. 3 indicates, in contrast to BSO based algorithms, PSO based approaches do not defeat GA based approaches in all the cases. In one case GA is faster [66] and in another case PSO [15]. However, since PSO is 40 times faster than the GA proposed by Kuoa et al. [15] and there is no significant difference between the GA and PSO reported by Vyas and Chauhan [66], the approach of Kuoa et al. [15] is faster. It was shown [11] [13] that GA based ARM approaches are faster than Apriori, so Kuoa et al. [15] is potentially faster than Apriori.

After all, according to these comparisons, NICGAR [90], MONPNAR [87], G3PARM [95], MDS-H [10], SRmining [68], PMES [65], Ant-ARM [70], and Kuoa et al. [15] have the potential to be the fastest heuristic ARM algorithms. However, there is a big gap in this area and all the heuristic approaches should compare themselves with each other and instead of comparing themselves with Apriori, they should try to establish comparisons with more up-to-date and faster ARM algorithms. The worst part is that most of them did not compare their approaches even with FP-growth, which is faster than Apriori.

Interestingness

Many definitions could define interestingness. There are many mechanisms, which can find out that a rule is interesting, or not. For instance, using simple performance metrics like minimum support and confidence is a way to extract interesting rules. However, many users prefer rules that are more interesting. As a result, applying minimum support and confidence may not satisfy them completely. ARM algorithms in Yan et al. [3], Dash et al. [16], Nandhini et al. [18], Soysal [5], Bidgoli et al. [17], Aoussi et al. [40], Ghosh and Nath [43], Qodmanan [92], NICGAR [90], MONPNAR [87], [53-56], [58], Kaya [63] in addition or even instead of the minimum support and confidence, considered other factors to determine interestingness of the extracted rules. They may have different mechanisms to evaluate the interestingness of a rule. Nandhini et al. [18] developed a method which is based on user knowledge (in one of the algorithm stages, they ask users to determine the user's expectations about rules) and the domain

ontology process. Another interestingness parameter has been proposed in MASP [5].

According to that, if we have a rule like: A→ C, then in this paper authors defined a performance metric, named Lift. It represents that how much C is dependent to A.

$$L(A \rightarrow C) = \frac{Conf(A \rightarrow C)}{S(C)} \qquad (10)$$

where Conf and S represent the confidence and support values, respectively.

Automatic Procedure

One of the biggest drawbacks of some ARM approaches is that users should set the value of minimum support and confidence, manually. Since these values could affect the quality of generated rules, especially in huge databases, it could be difficult to set them manually. A simple solution is a trial and error approach. It means that one tries many values to get the best result, but this is not easy [3].

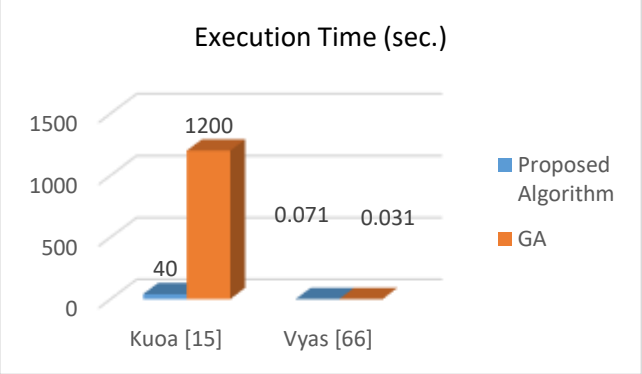


Figure 3. Comparison of GA based and PSO based ARM algorithms

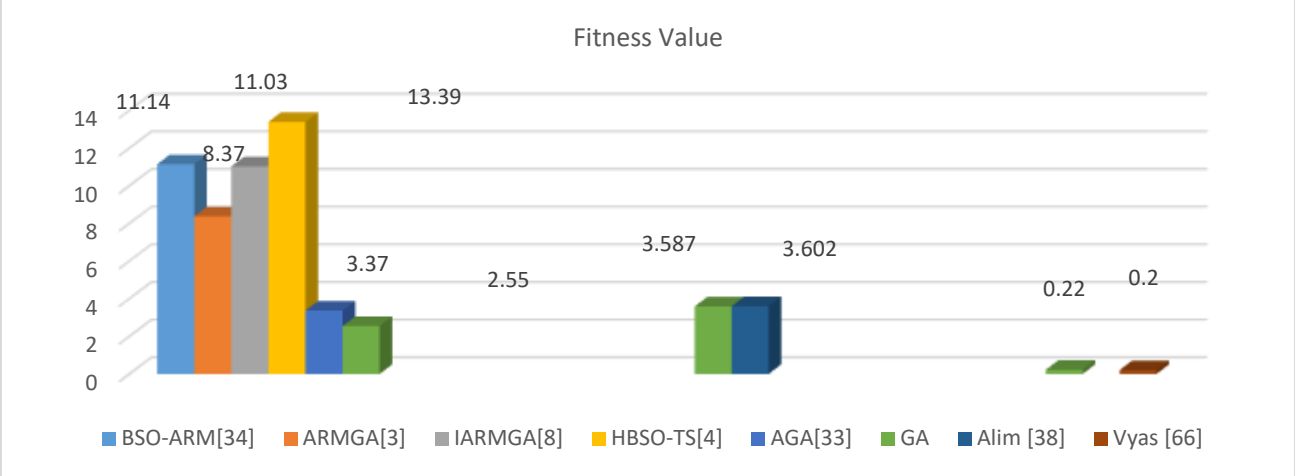


Figure 4. Comparison based on the Fitness Values

Some researchers detected this drawback and proposed some semi-automatic or automatic ARM algorithms to find the minimum support and confidence, such as Yan et al. [3], [34], Kuoa et al. [15], Dash et al. [16], Nandhini et al. [18], Bidgoli et al. [17], Puig et al. [61], Khabzaoui et al. [62], Kaya [63], Alatas et al. [74], Ghafari and Tjortjis [71].

This could have many benefits: 1) there is no need to have prior knowledge about the database, 2) when working with huge databases, it may be difficult to set these values manually 3) this process could avoid faults that may be caused by setting wrong values for minimum support and confidence. However, this area needs more investigation.

Completeness

It is worth taking into consideration that, in most of the heuristic approaches, completeness has been considered as the fitness value, and measured based on this fitness value.

Experimental results of ACS [12] indicate that many similar rules are generated by this method. Although it may not result in decreasing completeness, it could be wasting time. They also prepared an expert questionnaire and interviewed expert graduate students of the medical university of Taiwan. Based on this, they claim that their approach produces more complete rules like Apriori.

Moreover, in Pathak et al. [19] they considered completeness as an important performance metric. Completeness of generated rules was also another important performance metric in IARMGA and IARMMA [8]. IARMMA has better solution quality. The authors claimed that their approach solved the problem of generating false rules. Their experimental results indicate that with increasing in the size of data, their approaches have higher fitness values.

Djenouri et al. [4] approach's experimental results indicate that it has better fitness value compared to approaches in Djenouri et al. [34] and Yan et al. [3]. Moreover, BSO-ARM produces more complete rules than ARMGA [3]. According to Djenouri et al. [34], ARMGA generates rules that do not respect the minimum support and confidence and may even generate false rules. Finally, in Sagar and Argraval [36] they considered all types of rules (TP, TN, FP and FN) in their approach and their proposed a method generating rules with 100 % completeness.

We compared heuristic approaches based on their fitness values in Fig. 4. Since these approaches have been tested on different datasets and for having a fair comparison, heuristic approaches are compared in three categories. In the first category, the comparison is between heuristic approaches and GA algorithms from the point of view of fitness values.

It is clear that HSBO-TS [4] with 13.39 has the highest fitness value among these approaches. After that, BSO-ARM [34], IARMGA [8], ARMGA [3], and AGA [33] are in the next places, respectively. Moreover, GA approach with 2.55 has the least fitness value compared to other methods. It is interesting that the fitness value of HSBO-TS is approximately 5 times higher than GA.

In the next category, comparison indicates that Alim and Ivanov [38] has a fitness value, which is slightly higher than GA. Moreover, in the third category, Vyas and Chauhan [66] has even lower fitness value compared to GA. In conclusion, according to Fig.4, HSBO-TS has the highest fitness value among these approaches. In addition, Pathak et al. [19] has fitness value of 74.33 which is higher than 70.452 of Apriori, so it could also be considered as complete.

Additional Features

There are also other parameters that were not frequently applied in the literature, but they may have great impact on ARM algorithms' performance.

No. of database scans

One of the biggest challenges in ARM approaches is the number of database scans. It is obvious that if an ARM algorithm needs several database scans, it requires many I/O operations, which results in increasing execution time and performance degradation. There is a competition between ARM algorithms to have the least number of database scans.

Researchers in Kau and Shih [12], Goyal et al. [6], Van et al. [68], [7], considered this parameter in their experimental process. Since Apriori scans the database many times and this number could vary in different databases, hence, we do not include it in Fig. 5. FP-growth solved this problem and generates frequent rules with two scans. Until now, many ARM algorithms have been proposed to decrease this number even more. ACS [12], SRMining [68], and Narvekar [7] are such algorithms. They all require to scan the database just once. Moreover, Skyline [6] needs the same number of database scans as FP-growth. However, it is clear that researchers who are

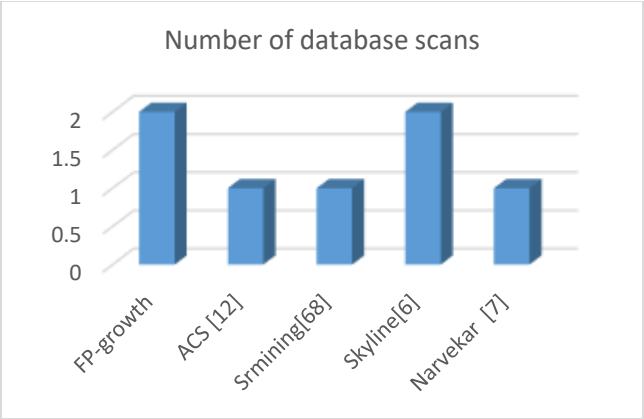


Figure 5. Number of database scans

working with heuristic approaches did not investigate this performance metric enough and it requires further study.

Parallel processing

With many improvements in parallel computing, we have an opportunity to employ this concept in ARM. This could bring many benefits, especially in decreasing the execution time. However, this is another gap in heuristic ARM approaches. Among these algorithms, only Dash et al. [16] and Djenouri et al. [65] implemented their approaches based on a parallel structure. The results indicate that it had positive impact on their execution time. Therefore, researchers should have further investigation on the potential of parallel computing.

Number of generated itemsets or rules

Other important performance metric in ARM algorithms is the number of generated rules and itemsets. If an ARM algorithm generates many itemsets, then some of them may be non-frequent; hence, it may take too long to process and eliminate them. It could also cause production of wrong rules. Therefore, some researchers proposed methods to decrease the number of generated itemsets. In addition, many of the generated rules are not interesting. Hence, researchers tried to decrease this number and generate only the most interesting rules. This could help to decrease the time and memory consumption. MONPNAR [87] applies a Pareto- based algorithm and produces about 4000 times less rules compared to Apriori. MDS-H [10] employs a grouping technique and prevents joining between none-frequent itemsets. If an itemset has two items, which belong to the same group, it is not a frequent itemset.

This would decrease the number of generated rules. Nandhini et al. [18] use an interesting procedure. After the ARM process, their approach generates many rules. At that point, they ask the user to define his/her

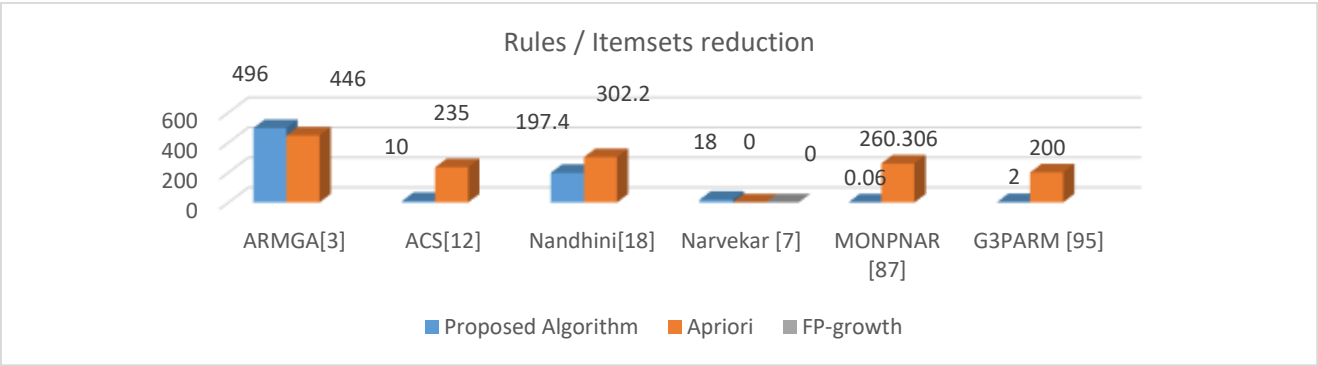


Figure 6. Comparison from the point of view of number of Rules / Itemsets

expectations about the generated rules. After this process, they remove rules that do not match those expectations. As a result, the number of rules would dramatically decrease. Moreover, the proposed algorithm in Goyal et al. [6] calculates the utility of single and pair itemsets. This algorithm pre-processes the data and eliminates some nodes from UP-tree. They also apply another heuristic approach that uses node utility of support to increase the maximum utility of an itemset. This approach helps them to eliminate non-frequent itemsets. As a result, it decreases the number of generated itemsets.

Finally, ACS [12] succeeded to have great reduction in the number of generated rules. Compared to Apriori, their approach generates approximately 24 times less rules. Fig. 6 illustrates the comparison results of heuristic approaches from the point of view of the number of rule / itemset reduction.

It is noticeable that the actual number of generated rules by ARMGA and Apriori in the first category has been divided by 1000 to have numbers, which are at the same range of other results.

As it is clear in Fig. 6, G3PARM [95] has the first, MONPNAR [87] has the second and ASC [12] has the third most rules reduction among other approaches. However, since according to NICGAR [90], NICGAR generates around 4 times less number of rules compared to MONPNAR [87], it would be the second approach from the point of view of number of generated items or rules. ASC generates approximately 23 times less rules compared to Apriori. After that, Narvekar [7] shows a significant feature, similarly to FP-growth. This algorithm does not need to generate any frequent candidate itemset, which results in generating frequent itemsets in a short period of time. ARMGA [3] and Nandhini et al. [18] also generate respectively 1.11 and 1.53 times less rules than Apriori.

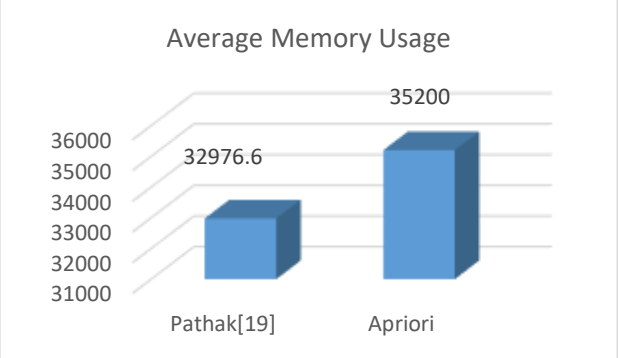


Figure 7. Average Memory Consumption

Memory Usage

Memory consumption is one of the main performance metrics in ARM. It is interesting that only one heuristic approach in the literature, considered memory consumption as an evaluation parameter. However, this performance metric is important for system performance. Pathak et al. [19], have tested 5 datasets. For simplicity, we considered the average memory consumption of the proposed method and compared it with the average memory usage of Apriori (Fig. 7). It is clear that Pathak et al. [19] has less memory consumption compared to Apriori. However, there is a big gap in this area and more research is needed to decrease memory usage in heuristic ARM approaches. It is noticeable that Narvekar [7] employed some mechanism to decrease the memory usage. In contrast to Apriori, it does not generate any candidate itemset and in contrast to FP-growth, it does not generate conditional FP trees, which results in having low memory consumption. However, they did not test their approach and compare it with other approaches, like Apriori or FP-growth.

Applications of Heuristic Algorithms

As mentioned earlier, heuristic approaches are fast and do not need any prior knowledge about the problem domain. Heuristic algorithms have many real

world applications. Zhu et al. [69] applied ACO to propose a new AR decision algorithm for a Ball Mill Pulverizing system which determines a set of variables that control the system. The authors believe that the AR decision problem is a combinational problem that it is hard to be solved and ACO is a good approach to solve very hard combinatorial optimization [69]. Their simulation results illustrate that applying this heuristic approach can result in finding the best association rules. Moreover, Liang et al. [76] also proposed a new fuzzy based hybrid PSO approach for forecasting wind speed and workload demand in wind energy systems. Their experimental results indicate that their heuristic based approach have a better performance in finding the best setting compared to the current approaches [76]. Their results indicate that their approach has a high satiability to find the final solution. For instance, the fuel cost difference between the maximum and minimum fuel costs, the emission difference of the maximum and minimum total emission, and the power loss difference of the maximum and minimum total real power loss are only 0.0004 M$/h, 0.4 ton/h, and 0.05 MW, respectively, which shows the efficiency of this approach.

Moreover, Schoonderwoerd et al. [60] employed ACO to propose a new routing policy in telecommunication networks and to establish load balancing in such networks. They mentioned that a single controller in distributed networks can have several drawbacks like the fact that a single point of the network (controller) should have the information of each part of the network, there should be a link between each node in the network and the controller, and more importantly, this network suffers from the single point of failure problem [60]. Hence, they believed their heuristic approach can be used to design a decentralized control mechanism for networks. Based on their experimental results, their heuristic based approach has less call failures compared to state of the art approaches.

Furthermore, there was an application of GAs to find the optimal solution for reactive power planning in [77]. They compared their approach with current approaches and their experimental results indicate that it has satisfactory convergent characteristics and sufficiently short execution time. The only limitation of this approach based on the authors' opinion is its computational time. However, back in 1994 the authors predicted that with future improvements in computational power, their approach could have much better computational time compared to the current approaches. In addition, there are some applications of the Artificial Bee Colony Algorithm in real world problems like the constrained Weber problem [78]. Stojanovi et al. [78] proposed new approaches for solving the constrained Weber problem based on four heuristic algorithms, of which the Artificial Bee Colony has the best performance with respect to solution quality, robustness and computational time [78]. The authors believe that the mentioned problem has a nonconvex feasible set that is hard for deterministic algorithms to find a global optimum; in such occasions, heuristic algorithms can have complete results.

M. Balokovic and M. Kürster [79] proposed a new approach for "radial velocity data from multi-planet extrasolar systems" based on GA. Their experimental results indicate that their heuristic based approach increases the quality of solutions and can decrease execution time, compared to current approaches. Moreover, in the cloud computing field, there are many applications of heuristic algorithms. Wang, et al. [80] proposed a new GA based approach, called load balancing genetic algorithm (JLGA) for "task circulation categorization" in cloud computing environments. Their experimental results indicate that although JLGA can establish a great load balancing in the cloud computing environment, it requires 30 more generations for coverage compared to the original heuristic-based approach, called AGA.

Furthermore, in our previous works [81-83], we also proposed three different approaches based on heuristic algorithms, like ABC, Cuckoo Optimization Algorithm, and Imperialism Competitive Algorithm to establish load balancing in cloud computing datacenters and decrease the energy consumption. Our experimental results indicate that the proposed heuristic approaches have better performance compared to state-of-the-art algorithms like Local Regression [81], Dynamic Voltage Frequency Scaling [81], Interquartile Range [81], and Median Absolute Deviation [81]. For instance, Bee-MMT (based on ABC) has 26.46% less energy consumption compared to the Local Regression based approach; or it has around nine times fewer virtual machine migration compared to the Local Regression based approach. However,

these approaches have a limitation. In all of them the number of hours that the hosts are in 100 percent of their utilizations are much more that of the current approaches, which can result in Service Level Agreement violation.

Finally, in [84] the application of heuristic and metaheuristic algorithms in the Artificial Neural Network training stage was discussed. In addition, the possibility of employing GA and PSO in the training stage of Deep Learning was checked. The authors claim that although heuristic algorithms can speed up the training stage of deep learning approaches, there were not enough studies in this area. They also asserted that even with high computational power available, it is still necessary to apply heuristic algorithms. They noted that finding every possible solution is a time consuming process, hence, finding near-optimal results with heuristic algorithms could be a possible solution for many current real world applications.

Discussion

In this paper, we focused on current ARM solutions, which are based on heuristic approaches. Applying heuristic algorithms is one of the most popular solutions in many current technological problems, especially because of their interesting structure, which does not require having any knowledge of the problem in advance. We focused on ARM approaches that directly apply heuristic algorithms and classified them based on the type of heuristic: GA, PSO, BSO, ACO, and others.

In section 5, we compared these approaches from many points of view, using performance metrics grouped into two main categories. In the first category, we considered mandatory performance metrics and in the second, we used additional performance metrics. We compared these methods based on memory usage. Surprisingly, although this is a very important metric, only one heuristic approach investigated this performance metric. However, there were some initial attempts in some researches, but their approaches were not implemented. It is one of the biggest gaps that should be addressed by researchers in the future.

We also discussed one of the most popular performance metrics in ARM algorithms, which is execution time. First, we evaluated algorithms that were originally compared with Apriori. The results illustrate that NICGAR [90], MONPNAR [87], G3PARM [95], MDS-H [10], Ant-ARM [70], and SRmining [68] are the fastest heuristic ARM algorithms compared to Apriori. At that point, we compared approaches that compared themselves with other heuristic approaches. The results indicate that PMES [65] and Kuoa et al. [15] are among the fastest ARM approaches. After all, these five heuristic algorithms have the least execution time compared to other ARM algorithms.

Although many researchers have investigated execution time of ARM approaches, there are still many gaps in this area. For instance, we should have many more approaches that are at least faster than FP-growth. Researchers should focus on proposing approaches, which are faster than new ARM algorithms. Moreover, there is another big gap in comparing all heuristic approaches with each other and find out which approach could be faster in extracting the frequent rules.

After that, we considered another mandatory performance metric, completeness, and compared heuristic approaches according to this. Completeness of generated rules can be important for users. Hence, many researchers tried to generate rules with high completeness. Kau and Shih [12], Pathak et al. [19], and Drias [8] are among these approaches focusing on completeness. According to Djenouri et al. [34], the approach that has been proposed in Yan et al. [3] is incomplete and may produce some false rules. As a result, in Djenouri et al. [34] and Djenouri et al. [4] they managed to propose approaches, which are more complete. In addition, Sagar and Argraval [36] proposed a method that has 100% completeness. Finally, we compared some of the heuristic

approaches from the point of view of the fitness value, which represents completeness of generated rules. The results indicate that HSBO-TS [4] has the highest fitness value compared to other approaches. After all, it seems that completeness is greatly taken seriously by researchers. However, we believe that in any new ARM algorithms, completeness of generated rules should have high priority.

Then, we compared current approaches from the point of view of the number of required database scans. It is noticeable that since Apriori needs several database scans to extract frequent rules, many approaches tried to improve Apriori, and focused on decreasing the number of database scans, which resulted in decreasing execution time. However, again, in the heuristic approaches this idea was not investigated enough. Only four of them considered this parameter and based on that ACS [12], SRMining [68] and Narvekar [7] scan the database only once. This parameter could have a great impact on execution time of ARM; ACS [12] and SRMining [68] can be a proof of this. They both are fast approaches and one of the main reasons is scanning the database only once. Because of the lack of implementation, we cannot discuss execution time in the case of Narvekar [7]. However, authors claimed that their approach is fast.

Next, we considered parallel processing as another comparison performance metric. If an ARM algorithm uses parallel processing, it would have many benefits like requiring less computational time. However, only Dash et al. [16] and PMES [65] employed such mechanism. Researchers should investigate applying parallel mechanism in their ARM algorithms in the future.

Next, automatically extracting frequent rules was one of the main goals of many researchers in ARM. This could have many benefits, like decreasing the possibility of using wrong values for performance metrics such as minimum support and confidence. In addition, in such structure, there is no need to have any knowledge about the characteristics of the dataset. Among heuristic ARM algorithms, the ones proposed by Yan et al. [3], Djenouri et al. [34], Kuoa et al. [15], Dash et al. [16], Nandhini et al. [18] and Bidgoli et al. [17] were investigated and they all were benefitted from this automatic procedure.

Another comparison performance metric was Interestingness. It is true that users are looking for complete rules with the minimum possible cost, but ARM approaches should generate rules, which are more interesting for users. Especially for current applications of ARM algorithms, users are looking for the rules that may not make any sense in the first instance, but with further investigation, it is found out that they are complete and interesting. In such cases, using only the minimum support and confidence is not enough. For instance, Yan et al. [3], Dash et al. [16], Nandhini et al. [18], Soysal [5], and Bidgoli et al. [17] have investigated some other performance metrics and approaches to extract more interesting rules aiming to prevent any completeness reduction. This area also needs further investigation.

Finally, this paper considered another performance metric, the number of generated itemsets or rules, and compared heuristic rules from the point of view of this performance metric. As it is clear, if an approach decreases the number of generated itemsets and eliminates non-frequent itemsets, this has a big impact on other performance metrics like time or memory consumption. On the other hand, for users, it is more convenient to have less but highly quality rules. Algorithms like these proposed by Yan et al. [3], Hong and Bian [10], Nandhini et al. [18], Goyal et al. [6], [7], and Kau and Shih [12] investigated this area and proposed some solutions to decrease this performance metric. However, among heuristic approaches, G3PARM [95], NICGAR [90], MONPNAR [87], ASC [12] and Narvekar [7] have the most rules and itemsets reduction, respectively. This reduction had positive impact on their execution time. There could be more improvement in this area in the future.

Now it is time to finalize the comparison process and compare the best approaches, which have been found so far. Table 3 shows this comparison results. According to Table 3 and all the previous discussion, five heuristic approaches are the most accomplished heuristic ARM algorithms. These approaches have the highest *GT-Ranks* compared to other heuristic ARM approaches. MOPNAR [87] and NICGAR [90] are the best heuristic approaches regarding their *GT-Rank* which is 32. They are fast, they generate less number of rules, but the generated rules are more interesting. Next, G3PARM [95] has the highest *GT-Rank* (28). While it tries to generate less number of rules, it has a low execution time. ARMGA [3] is another best heuristic approach. Its *GT-Rank* is 27 and it is a fast approach, which also generates less itemsets/rules compared to Apriori. It is an automatic ARM approach trying to produce interesting rules. Finally, ACS [12] also has 25 *GT-Rank* and it is a fast approach, which scans the database only once. It generates complete rules and produces very high itemsets/rules reduction compared to other approaches. Besides that, these results include some facts, as follows:

- Because there is not enough evidence about memory usage of heuristic approaches, we cannot select Pathak [19] as the best approach from the point of view of memory usage. As a result, we need more comparisons in the future to find out which approach has the least memory usage.
- One of the main reasons that SRmining [68] is among the fastest algorithms is that it scans the database only once.
- Employing a parallel mechanism is one of the main reasons that made PMES [65] is one of the fastest heuristic ARM approaches.
- Dash et al. employed a parallel mechanism, it is an automatic algorithm, and it returns interesting patterns [16]. Although this approach did not investigate performance metrics, it may have the potential to have good outcomes on other performance metrics, too.
- The only reason that Narvekar [7] have not been selected as one of the best ARM approaches is that authors did not produce some actual results to prove the efficiency of this approach. However, this approach scans the database only once, it does not produce any candidate itemset, and it is one of the few approaches that considers memory usage. Moreover, because of the mentioned features, it has the potential to be a fast algorithm. As a result, more investigation on this algorithm could show if it is one of the best heuristic ARM approaches.

Finally, since this paper is about ARM, it would be interesting if we consider Table 3 as a transactional database and analyse it with ARM. Extracted frequent rules from Table 3 are reported in Table 4. In this table, Confidence X for a rule A->B means that the rule is correct in X% of all the occasions. In other words, B co-occurs in X% of the occurrences of A. It can be calculated by formula 3.

For instance, according to the first rule in Table 4, if an approach has low memory consumption then, with possibility of 100%, it also considers decreasing execution time. To calculate the confidence value of this formula, we check table 3 and look for the occasions that an approach employs the memory consumption performance metric. You may notice that only Narvekar [7] and Pathak [19] consider this metric, so T is 2 and both also have the second part of the rule, hence, E=2. In other words, whenever, an approach considered the memory usage metric, it also considered execution time.

Moreover, if an ARM approach is complete, so with 69% probability it may have low execution time. There is also a strong relationship (100% probability) between a parallel mechanism or scanning the database once or twice and having low execution time. In addition, with probability of approximately 40%, if an approach employs an automatic mechanism, it also generates rules that are interesting.

According to Table 4, if an ARM approach generates few rules / itemsets and the generated rules are interesting (with a probability of 50%) this approach tends to be automatic. Moreover, if an ARM algorithm has few database scans and generates complete rules, one of its main goals is to decrease the execution time.

It is also interesting that if an ARM approach generates few rules / itemsets and it scans the database only few times, then it intends to have low execution time (probability 100%). As it was mentioned before, these 2 items are really important to decrease the execution time. In addition, with probability of 50 %, if an approach focuses on being complete and automatic, it also focuses on being fast. There is also another interesting rule. According to that, if an ARM algorithm scans the database once or twice and has low memory usage, it would try to be fast.

Furthermore, there are only two interesting rules left. First, if an ARM approach scans the database once or twice and reduces the number of generated itemsets or rules, and generates complete rules, it definitely tries to be a fast approach. Secondly, if an approach employs a parallel mechanism, follows an automatic procedure, and generates interesting rules, it also tends to generate complete rules in a short period of time.

According to the extracted rules in Table 4, since parallel processing and the number of database scans have great impact on execution time, so researchers tend to apply them in their approaches to decrease execution time. Combination of Rule / Itemset Reduction and Automatic Procedure or Database Scans is also another popular approach to decrease execution time; which means that only reducing the number of itemsets or rules may only be employed in 50% of cases. Finally, it seems that if an approach is intended to be fast, it considers performance metrics like the number of database scans, the number of rules/itemsets, and parallel processing.

Table 4. Extracted rules from final Comparison results

Frequent Rules	Confidence (%)
Memory Usage → Execution Time	100
Completeness → Execute Time	69
Parallel processing → Execute Time	100
Database Scans → Execute Time	100
Automatic Procedure → Interestingness	40
Rule / Itemset Reduction → Execute Time	50
Rule / Itemset Reduction + Interestingness → Automatic Procedure	50
Completeness + Database Scans → Execute Time	100
Rule / Itemset Reduction + Database Scans → Execute Time	100
Automatic Procedure + Completeness → Execute Time	50
Database scans + Memory Usage → Execute Time	100
Rule / Itemset Reduction + Completeness + Database Scans → Execute Time	100
Parallel processing + Automatic Procedure +Interestingness → Execute Time	100

Open challenges and suggested directions

There are many open challenges in this area that should be addressed in the future. The most important challenge for future heuristic ARM approaches is that researchers should compare them against other ARM approaches and algorithms, which do not employ heuristics. Currently, most of heuristic approaches have been compared with Apriori, but, as the Ventura and Luna [86] pointed out, Apriori needs many steps to compute all frequencies of patterns, which can result in requiring a lot of computational power and memory space. Such approaches may not be the feature of a state-of-art approach and

researchers should compare their methods with more recent and efficient ARM approaches.

For instance, from the point of view of execution time, most of the heuristic approaches were compared with Apriori. It is not enough to defeat Apriori, which is not considered to be a fast approach, nowadays. Therefore, this could be a hot topic for further research. In addition, another area that has not been explored enough is that of performance metrics which have not been addressed by most of the existing heuristic algorithms.

A comprehensive solution for ARM should consider most of these performance metrics at the same time. Especially some of these performance metrics like memory usage, completeness, and parallel processing should be investigated in more depth. Finally, it seems that the majority of recent papers focus on execution time. Many organizations and companies own large databases, which require fast approaches to be analysed. This trend for fast ARM may increase in the future. Nevertheless, as mentioned in many papers, extracting interesting rules is also desirable for users. In conclusion, future researches should focus on decreasing execution time and generating more complete and interesting rules with low memory usage. It may also be better to test approaches on larger databases to make them suitable for real world usage.

Conclusion

With the dramatic increase in trends to extract knowledge from data, many data mining techniques have been explored. One of the best known data mining techniques is association rules. It extracts the most frequent rules and patterns from a database. There are many ARM algorithms, which use many different approaches to extract frequent rules. One family of such approaches applies heuristic algorithms or just heuristic characteristics. Many papers in this field have been published until now; although there have been some attempts, to the best of our knowledge, there is no comprehensive review on this area, except from that of Ventura and Luna [86] and M.J. del Jesus et al. [98], who reviewed Pattern Mining with Evolutionary Algorithms. As a result, this paper attempts to provide a comprehensive study on heuristic approaches, rather than just evolutionary ones, and discusses their advantages and drawbacks. We also considered all the necessary evaluation performance metrics and then compared all reviewed algorithms based on these performance metrics. We proposed an evaluation metric, named GT-Rank to rank heuristic approaches and select the best ones. We selected MOPNAR, NICGAR, G3PARM, ARMGA, and ASC as the best approaches with the highest GT-Ranks. Finally, we considered Table 3 as a transactional database and extracted frequent rules from this. They are interesting rules that represent trends of heuristic approach research. Given the recent trend for Big Data we plan soon to review how ARM algorithms cope with it.

REFERENCES

[1] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In: *VLDB '94 Proc. of the 20th Int'l Conf. on Very Large Data Bases*, J.B. Bocca, M. Jarke, C. Zaniolo (Eds.). USA: Morgan Kaufmann Publishers; 1994.

[2] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. In: *ACM SIGMOD Conf. on Management of Data*. New York: ACM; 1993.

[3] X. Yan, C. Zhang, S. Zhang. Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems with Applications* 2009, 36: 3066–3076.

[4] Y. Djenouri, H. Drias, A. Chemchem. A Hybrid Bees Swarm Optimization and Tabu Search Algorithm for Association Rule Mining. *World Congress on Nature and Biologically Inspired Computing (NaBIC)*. Fargo, ND: IEEE; 2013.

[5] Ö. M. Soysal. Association rule mining with mostly associated sequential patterns. *Expert Systems with Applications* 2015, 42: 2582–2592.

[6] V. Goyal, A. Sureka, D. Patel. Efficient Skyline Itemsets Mining. In: *C3S2E '15 Proc. of the Eighth Int'l Conf. on Computer Science & Software Engineering*. Japan: ACM; 2015.

[7] M. Narvekar, S. F. Syed. An optimized algorithm for association rule mining using FP tree. In: *International Conf. on Advanced Computing Technologies and Applications (ICACTA)*. India: Elsevier; 2015.

[8] H. Drias. Genetic algorithm versus memetic algorithm for association rules mining. In: *Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC)*. Porto: IEEE; 2014.

[9] J. Yuan, S. Ding. Research and Improvement on Association Rule Algorithm Based on FP-Growth. In: *International Conf. on Web Information Systems and Mining, WISM*. China: Springer; 2012.

[10] Z. Hong, F. Bian. A Heuristic Approach for Fast Mining Association Rules in Transportation System. In: *Fifth Int'l Conf. on Fuzzy Systems and Knowledge Discovery*. Shandong: IEEE; 2008.

[11] C. Li, M. Yang. Association rules data mining in manufacturing information system based on genetic algorithms. In: *3rd Int'l Conf. on Computational Electromagnetics and Its Applications* ICCEA. 2004.

[12] R.J. Kuo, C.W. Shih. Association rule mining through the ant colony system for National Health Insurance Research Database in Taiwan. *Computers and Mathematics with Applications* 2007, 54: 1303–1318.

[13] H. Guo, Y. ZHOU. An Algorithm for Mining Association Rules Based on Improved Genetic Algorithm and its Application. In: *Third International Conf. on Genetic and Evolutionary Computing*. Guilin: IEEE; 2009.

[14] J. Han, J. Pei, Y. Yin, R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery* 2004, 8: 53-87.

[15] R.J. Kuoa, C.M. Chaob, Y.T. Chiu. Application of particle swarm optimization to association rule mining. *Applied Soft Computing* 2011, 11: 326–336.

[16] S. R. Dash, S. Dehuri, S. Rayaguru. Discovering Interesting Rules from Biological Data Using Parallel Genetic Algorithm. In: *IEEE 3rd Int'l Advance Computing Conf. (IACC).* Ghaziabad: IEEE; 2013.

[17] B. Minaei-Bidgoli, R. Barmaki, M. Nasiri. Mining numerical association rules via multi-objective genetic algorithms. *(Elsevier) Information Sciences* 2013, 233: 15–24.

[18] M. Nandhini, M. Janani, S.N. Sivanandham. Association Rule Mining Using Swarm Intelligence and Domain Ontology. In: *Int'l Conf. on Recent Trends In Information Technology (ICRTIT)*. Chennai, Tamil Nadu: IEEE; 2012.

[19] N. Pathak, V. Shah, C. Ajmeera. A Memory Efficient Algorithm with Enhance Preprocessing Technique for Web Usage Mining. In: *ICTCS '14 Proc. of the International Conf. on Information and Communication Technology for Competitive Strategies*. Gujarat: ACM; 2014

[20] C. Borgelt. Frequent item set mining. *Data Mining and Knowledge Discovery* 2012, 2: 437–456.

[21] B. Nath, D. K. Bhattacharyya, A. Ghosh. Incremental association rule mining: a survey. *Data Mining and Knowledge Discovery* 2013, 3: 157–169.

[22] T. Le, B. Vo, G. Nguyen, "A survey of erasable itemset mining algorithms. *Data Mining and Knowledge Discovery* 2014, 4: 356–379.

[23] M. Zhang, C. He. Survey on Association Rules Mining Algorithms. *Advancing Computing, Communication, Control and Management* 2010, 56: 111-118.

[24] M. Elloumi, A. Y. Zomaya. *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*. 1, ed. John Wiley & Sons, Inc. Hoboken, New Jersey; 2014.

[25] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. A. Coello. Survey of Multi objective Evolutionary Algorithms for Data Mining: Part II. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* 2014. 18: 4-19.

[26] M.J. del Jesus, J.A. Gámez, P. González, J.M Puerta, On the discovery of association rules by means of evolutionary algorithms. *WIREs Data Mining Knowl. Discov.* 2011, 1: 397-415

[27] C. Wang, C. Tjortjis. PRICES: An Efficient Algorithm for Mining Association Rules. *Intelligent Data Engineering and Automated Learning* 2004, 3177: 352-358.

[28] B. Liao. An Improved Algorithm of Apriori. In: *4th International Symp., ISICA*. China: Springer; 2009.

[29] X. Liang, C. Xue, M. Huang. Improved Apriori Algorithm for Mining Association Rules of Many Diseases. In: *5th International Symp., ISICA*. China: Springer; 2010.

[30] Z. Chen, S. Cai, Q. Song, C. Zhu. An Improved Apriori Algorithm Based on Pruning Optimization and Transaction Reduction. In: *2nd International Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*. Deng Leng: IEEE; 2011.

[31] H. Yu, J. Wen, H. Wang, L. Jun. An Improved Apriori Algorithm Based On the Boolean Matrix and Hadoop. *(Elsevier) Advanced in Control Engineering and Information Science* 2011, 15: 1827 – 1831.

[32] T. Hong, C. Chen, Y. Wu, Yeong-Chyi Lee, "A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions. *Soft Computing* 2006, 10: 1091-1101.

[33] M. Wang, Q. Zou, C. Liu. Multi-dimension Association Rule Mining Based on Adaptive Genetic Algorithm. In: *Int'l Conf. on Uncertainty Reasoning and Knowledge Engineering*. Bali: IEEE; 2011.

[34] Y. Djenouri, H. Drias, Z. Habbas, H. Mosteghanemi. Bees Swarm Optimization for Web Association Rule Mining. In: *IEEE/WIC/ACM Int'l Conf. on Web Intelligence and Intelligent Agent Technology*. Macau: IEEE; 2012.

[35] A H.L. Lim, C.S. Lee, M. Raman. Hybrid genetic algorithm and association rules for mining workflow best practices. *Elsevier Expert Systems with Applications* 2012, 39: 10544–10551.

[36] M. Saggar, A.K. Agrawal, A. Lad. Optimization of Association Rule Mining using Improved Genetic Algorithms. In: *IEEE Int'l Conf. on Systems Man and Cybernetics*. IEEE; 2004.

[37] A, Dehuri S. Ghosh S. *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*. Vol. 98. Springer Berlin Heidelberg: Springer; 2008.

[38] F. Alim, K. Ivanov. Heuristic Rules Embedded Genetic Algorithm to Solve In-Core Fuel Management Optimization Problem. In: *GECCO '05 Proc. of the 7th annual Conf. on Genetic and evolutionary computation*. Washington DC: ACM; 2005.

[39] F. Du, N. Rao, J. Guo, Z. Yuan and R. Wang. Mining Gene Network by Combined Association Rules and Genetic Algorithm. In: *Int'l Conf. on Communications, Circuits and Systems (ICCCAS)*. Milpitas: IEEE; 2009.

[40] A. Salleb-Aouissi, C. Vrain, C. Nortet. QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules. In:

IJCAI'07 Proc. of the 20th international joint Conf. on Artificial intelligence, San Francisco: Morgan Kaufmann Publishers; 2007.

[41] J. Mata, J. L. Alvarez, J. C. Riquelme. An Evolutionary Algorithm to Discover Numeric Association Rules. In: *SAC '02 Proc. of the ACM Symp. on Applied computing*. USA: ACM; 2002.

[42] J. Mata, J. L. Alvarez, J. C. Riquelme. Mining Numeric Association Rules with Genetic Algorithms. In: *Artificial Neural Nets and Genetic Algorithms* 2001, 264-267.

[43] A. Ghosh, B. Nath. Multi-objective rule mining using Genetic algorithms. *Information Sciences* 2004, 163: 123–133.

[44] A. Orriols-Puig, J. Casillas, E. Bernadó-Mansilla. First approach toward on-line evolution of association rules with learning classifier systems. In: *GECCO '08 Proc. of the 10th annual Conf. companion on Genetic and evolutionary computation*. USA: ACM; 2008.

[45] M. Kaya, R. Alhajj. Utilizing Genetic Algorithms to Optimize Membership Functions for Fuzzy Weighted Association Rules Mining. *Applied Intelligence* 2006, 24: 7-15.

[46] W. Wang, S. M. Bridges. Genetic Algorithm Optimization of Membership Functions for Mining Fuzzy Association Rules. In: *International Joint Conf. on Information Systems, Fuzzy Theory and Technology Conf.* Atlantic City: Association for Intelligent Machinery; 2000.

[47] C. Chen, V. S. Tseng, T. Hong. Cluster-Based Evaluation in Fuzzy-Genetic Data Mining. *IEEE Transactions on Fuzzy System* 2008. 16: 249 – 262.

[48] C. Chen, T. Hong, V. S. Tseng, C. Lee. A Genetic-Fuzzy Mining Approach for Items with Multiple Minimum Supports. In: *IEEE International Fuzzy Systems Conf.* London: IEEE; 2007.

[49] C. Chen, T. Hong, V. S. Tseng. Speeding up genetic-fuzzy mining by fuzzy clustering. In: *IEEE International Conf. on Fuzzy Systems (FUZZ-IEEE)*. Jeju Island: IEEE; 2009.

[50] C. Chen, T. Hong, V. S. Tseng. An improved approach to find membership functions and multiple minimum supports in fuzzy data mining. *Expert Systems with Applications* 2009, 36: 10016–10024.

[51] P. S. Thilagam, V. S. Ananthanarayana. Extraction and optimization of fuzzy association rules using multi-objective genetic algorithm. *Pattern Analysis and Application* 2008, 11: 159–168.

[52] B. Wang, N. Zhou, D. Liu, L. Zhou, P. Wang. Genetic Algorithm-Based Rules Discovery for Networked Manufacturing Resources Management. In: *4th International Conf. on Wireless Communications, Networking and Mobile Computing*. Dalian: IEEE; 2008.

[53] Y. Hu. Determining membership functions and minimum fuzzy support in finding fuzzy association rules for classification problems. *Knowledge-Based Systems* 2006, 19: 57–66.

[54] H. Ishibuchi, T. Yamamoto. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems* 2004, 141: 59–88.

[55] J. Alcalá-Fdez, R. Alcalá, M. J. Gacto, Francisco Herrera. Learning the membership function contexts for mining fuzzy association rules by using genetic algorithms. *Fuzzy Sets and Systems* 2009, 160: 905–921.

[56] P. Deepa Shenoy, K.G. Srinivasa, K.R. Venugopal, L.M. Patnaik. Evolutionary Approach for Mining Association Rules on Dynamic Databases. *Advances in Knowledge Discovery and Data Mining* 2003, 2637: 325-336.

[57] F. Guillet, P. Kuntz, R. Lehn. A Genetic Algorithm for Visualizing Networks of Association Rules. *Multiple Approaches to Intelligent Systems* 1999, 1611: 145-154.

[58] M. Kaya, R. Alhajj, Genetic algorithm based framework for mining fuzzy association rules. *Fuzzy Sets and Systems* 2005, 152: 587–601.

[59] B. Alatas¸ E. Akin. An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules. *Soft Computing* 2006, 10: 230-237.

[60] Ruud Schoonderwoerd, Janet L. Bruten, Owen E. Holland, and Leon J. M. Rothkrantz. Ant-based load balancing in telecommunications networks, *Adaptive Behavior* 1996, 5: 169-207.

[61] A. Orriols-Puig, J. Casillas, F. Mart´. Unsupervised learning of fuzzy association rules for consumer behavior modeling. *Math ware and Soft Computing* 2009, 16: 29–43.

[62] M. Khabzaoui, C. Dhaenens, E. Ghazali Talbi. A Multicriteria Genetic Algorithm to analyze DNA microarray data. In: *Congress on Evolutionary Computation CEC*. USA: IEEE; 2004.

[63] M.t Kaya, "Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules. *Soft Computing* 2006, 10: 578-586.

[64] M. Anandhavalli, SMIT, M. K. Ghose, K. Gauthaman, M. Boosha. Global Search Analysis of Spatial Gene Expression Data Using Genetic Algorithm. *Recent Trends in Network Security and Applications* 2010, 89: 593-602.

[65] Y. Djenouri, A. Bendjoudi, M. Mehdi and N. Nouali-Taboudjemat. Parallel association rules mining using GPUS and bees behaviors. In: *IEEE 6th International Conf. of Soft Computing and Pattern Recognition (SoCPaR)*. Tunis: IEEE; 2014.

[66] P. Vyas, A. Chauhan. Comparative Optimization of Efficient Association Rule Mining through PSO and GA. In: *Int'l Conf. on Machine Intelligence Research and Advancement*. Katra: IEEE; 2013.

[67] B. Alatas, E. Akin. Rough particle swarm optimization and its applications in data mining. In: Soft Computing, 2008, 12: 1205-1218.

[68] N. Hong-yun, L. Jin-lan, Z. De-gan. Self-optimization Rule-chain Mining Based on Potential Association Rule Directed Graph. In: *International Symp. on Computational Intelligence and Design*. Wuhan: IEEE; 2008.

[69] W. Zhu, J. Wang, H. Cao, Y. Zhang. A Novel Association Rule Decision Algorithm Based on Ant Colony Optimization Algorithm for Ball Mill Pulverizing System. In: *Int'l Conf. on Computer Science and Software Engineering*. Hubei: IEEE; 2008.

[70] Y. He, S. C. Hui. Exploring ant-based algorithms for gene expression data analysis. *Artificial Intelligence in Medicine* 2009, 47: 105—119.

[71] S.M Ghafari., C. Tjortjis. Association Rules Mining by improving the Imperialism Competitive Algorithm (ARMICA). In: *IFIP AICT Proc. 12th Int'l Conf. on Artificial Intelligence Applications and Innovations (AIAI)*. Greece: Springer; 2016.

[72] L. D. Jorio, A. Laurent, M. Teisseire. Fast Extraction of Gradual Association Rules: A Heuristic Based Method. In: *CSTST'08: International Conf. on Soft Computing as Transdisciplinary Science and Technology*. France: ACM; 2008.

[73] J. Hu, X. Yang-Li. Association Rules Mining Using Multi-objective Coevolutionary Algorithm. *International Conf. on Computational Intelligence and Security Workshop*. Harbin: IEEE; 2007.

[74] B. Alatas, E. Akin, A. Karci. MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing* 2008, 8: 646–656.

[75] K. Prasanna, M. Seetha, A.P. S. Kumar. CApriori: Conviction Based Apriori Algorithm for Discovering Frequent Determinant Patterns from High Dimensional Datasets. In: *Int'l Conf. on Science Engineering and Management Research (ICSEMR)*. Chennai: IEEE; 2014.

[76] Ruey-Hsun Liang, Sheng-Ren Tsai, Yie-Tone Chen, Wan-Tsun Tseng. Optimal power flow by a fuzzy based hybrid particle swarm optimization approach. *Electric Power Systems Research* 2011, 81: 1466-1474.

[77] Iba K. Reactive power optimization by genetic algorithm. IEEE Trans Power Syst. 1994, 9:685–692.

[78] I. Stojanović, I. Brajević, Predrag S. Stanimirović, L. A. Kazakovtsev, and Z. Zdravev. Application of Heuristic and Metaheuristic Algorithms in Solving Constrained Weber Problem with Feasible Region Bounded by Arcs. Mathematical Problems in Engineering. 2017.

[79] Baloković, M.; Kürster, M. An Application of Heuristic Algorithms to Radial Velocity Data from Multiple-Planet Extrasolar Systems. YSC'15 Proceedings of Contributed Papers, pp. 44-47, Kyivskyi Universyte, 2008.

[80] Wang T, Liu Z, Chen Y, Xu Y, Dai X. Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing. Dependable, Autonomic and Secure Computing (DASC) IEEE 12th International Conference on, IEEE, 2014.

[81] S. M. Ghafari, M. Fazeli, A. Patooghy and L. Rikhtechi. Bee-MMT: A load balancing method for power consumption management in cloud computing. In: *Sixth International Conference on Contemporary Computing (IC3),* Noida, pp. 76-80, IEEE, 2013.

[82] M. Yakhchi, S. M. Ghafari, S. Yakhchi, M. Fazeli and A. Patooghi. Proposing a load balancing method based on Cuckoo Optimization Algorithm for energy management in cloud computing infrastructures. In: *6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, Istanbul, pp. 1-5, IEEE, 2015.

[83] S. Yakhchi, S. M. Ghafari, M. Yakhchi, M. Fazeli and A. Patooghy. ICA-MMT: A load balancing method in cloud computing environment. In: *2nd World Symposium on Web Applications and Networking (WSWAN)*, Sousse, pp. 1-7, IEEE, 2015.

[84] Tian, Zhonghuan, and S. Fong. Survey of Meta-Heuristic Algorithms for Deep Learning Training. Optimization Algorithms-Methods and Applications. Ozgur Baskan, InTech, 2016.

[85] V. Calderaro, V. Galdi, G. Graber and A. Piccolo, "Deterministic vs heuristic algorithms for eco-driving application in metro network," In: *International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS)*, Aachen, pp. 1-6, IEEE, 2015.

[86] S. Ventura and J. M. Luna. Pattern Mining with Evolutionary Algorithms. Cham, Switzerland, Springer, 2016.

[87] D. Martín, A. Rosete, J. Alcalá-Fdez and F. Herrera. New Multi-objective Evolutionary Algorithm for Mining a Reduced Set of Interesting Positive and Negative Quantitative Association Rules. *IEEE Transactions on Evolutionary Computation* 2014, 18: 54-69.

[88] A. M. Palacios, J. L. Palacios, L. Sanchez, J. Alcala-Fdez. Genetic learning of the membership functions for mining fuzzy association rules from low quality data. *Information Sciences* 2015, 295: 358-378.

[89] F. Padillo, J. M. Luna, F. Herrera, and S. Ventura. Mining association rules on Big Data through MapReduce genetic programming. *Integrated Computer-Aided Engineering* 2018, 25: 31—48.

[90] D. Martín, J. Alcalá-Fdez, A. Rosete, F. Herrera. NICGAR: A Niching Genetic Algorithm to mine a diverse set of interesting quantitative association rules. *Information Sciences* 2016, 356: 208-228,

[91] A. Cano, J. M. Luna, and S. Ventura. High performance evaluation of evolutionary-mined association rules on GPUs. *The Journal of Supercomputing* 2013, 66: 1438—1461.

[92] H. R. Qodmanan, M. Nasiri, B. Minaei-Bidgoli. Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. *Expert Systems with Applications* 2011, 38: 288-298.

[93] J. L. Olmo, J. M. Luna, J. R. Romero, S. Ventura. Mining association rules with single and multi-objective grammar guided ant programming. Integrated Computer-Aided Engineering, 2013, 20(3): 217-234.

[94] http://archive.ics.uci.edu/ml/ UCI Machine Learning Repository data sets. Accessed 27/6/18.

[95] J. M. Luna, J. R. Romero, S. Ventura. Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules. Knowledge and Information Systems. 32(1): 53-76 (2012).

[96] J. M. Luna, J. R. Romero, S. Ventura. Grammar-based multi-objective algorithms for mining association rules. Data Knowledge and Engineering. 86: 19-37 (2013).

[97] J. M. Luna, J. R. Romero, C. Romero, S. Ventura. Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm. Integrated Computer-Aided Engineering 21(4): 321-337 (2014).

[98] M.J. del Jesus, J.A. Gámez, P. González, J.M. Puerta. On the discovery of association rules by means of evolutionary algorithms. WIREs DMKD 2011, 1, 397-415