IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 May 2024, accepted 28 May 2024, date of publication 20 June 2024, date of current version 3 October 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3417334*

APPLIED RESEARCH

# An Exhaustive Multi-Aspect Analysis of Swarm Intelligence Algorithms in Numerical Association Rule Mining

**MINAKSHI KAUSHIK[1], RAHUL SHARMA[1,2], PILLERIIN KÕIVA[3], IZTOK FISTER JR.[4], AND DIRK DRAHEIM[1]**

[1]Information Systems Group, Tallinn University of Technology, 19086 Tallinn, Estonia
[2]Ajay Kumar Garg Engineering College, Ghaziabad 201015, India
[3]School of Information Technologies, Tallinn University of Technology, 19086 Tallinn, Estonia
[4]Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

Corresponding author: Minakshi Kaushik (minakshi.kaushik@taltech.ee)

This work was supported in part by the European Union through the European Social Fund through the project "ICT Program."

**ABSTRACT** Numerical association rule mining (NARM) is an extended version of association rule mining that determines association rules in numerical data items through distribution, discretization, and optimization methods. In the optimization techniques domain, numerous evolutionary and swarm intelligence-based algorithms have been proposed to extract association rules from numerical datasets. However, there is still a lack of comprehensive understanding regarding the performance of swarm intelligence-based algorithms, particularly for NARM. Presently, in state-of-the-art, various swarm intelligence-based optimization algorithms are claimed to be better based on their arbitrary comparisons with different algorithms in different classes, e.g., swarm intelligence-based algorithms are compared with genetic algorithms. Consequently, it becomes challenging to select the most suitable swarm intelligence-based algorithm for NARM. This article specifically aims to address this gap by conducting an exhaustive multi-aspect analysis of four popular swarm intelligence-based optimization algorithms (MOPAR, MOCANAR, ACO-R, and MOB-ARM) using four real-world datasets and six key metrics: performance time, the number of rules, support, confidence, comprehensibility, and interestingness, aiming to demonstrate the efficiencies of the SI-based algorithms in addressing the NARM problem. The achieved outcomes are also compared with the Apriori algorithm, which is one of the classical algorithms for association rule mining. In our analysis, MOPAR shows low rule count with high confidence, comprehensibility, and interestingness. MOCANAR consistently performs well across all parameters and datasets. ACO-R generates high-quality rules but may need parameter adjustments for large datasets. MOB-ARM is slower compared to others, and Apriori underperforms in support and time but excels in confidence.

**INDEX TERMS** Swarm intelligence optimization, association rule mining, machine learning, numerical association rule mining.

## I. INTRODUCTION

Numerical Association Rule Mining (NARM) represents an evolved form of classical association rule mining (ARM) technique developed explicitly for extracting association rules from datasets that contain continuous values or numerical attributes. Due to this capability, NARM has become increasingly relevant in numerous modern data analysis tasks.

### A. MOTIVATION

Several methods, such as optimization, discretization, and distribution, have been proposed in the literature to extend the capabilities of classical ARM to NARM [1], [2]. Among

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

these methods, optimization is considered a potential solution for dealing with the complexity of NARM, employing Evolutionary-based, Swarm Intelligence (SI)-based, and Physics-based algorithms [3]. Various SI-based optimization algorithms have been proposed, drawing inspiration from animal and insect movements, as well as the biological behavior of natural objects [4], [5], [6].

These optimization algorithms are helpful for mining association rules from numeric datasets without the need for discretization. However, it remains unclear which algorithms perform better for efficient NARM.

### B. LITERATURE GAP

In the existing literatures [7], [8], [9], and [10], comparisons of SI-based optimization algorithms with other algorithms have been conducted randomly across different classes, without specific comparisons within their own classes. Consequently, selecting the most suitable SI-based algorithm for NARM becomes a challenging task. To address this gap, we present an exhaustive multi-aspect analysis of four popular SI-based optimization algorithms, namely MOPAR [7], ACO-R [8], MOB-ARM [9], MOCANAR [10] and Apriori [11].

### C. OBJECTIVE

By conducting this comprehensive analysis, we aim to provide insights into the performance and effectiveness of these algorithms, specifically for NARM.

In our analysis, the MOCANAR, MOB-ARM, MOPAR, and ACO-R algorithms have demonstrated efficiency in solving multi-objective optimization problems across different domains, including NARM and continuous optimization.

Although the MOCANAR, MOB-ARM, MOPAR, and ACO-R algorithms are relatively new, there is a lack of performance comparisons available in the existing literature. Consequently, these algorithms are selected for their performance comparison to find a set of optimal solutions that trade off between multiple objectives simultaneously. The performance evaluation of these algorithms will surely shed light on their potential advantages and limitations.

### D. METHODOLOGY

In this paper, we begin by examining the applicability of the MOCANAR, MOB-ARM, MOPAR, and ACO-R algorithms in the context of NARM. Subsequently, we conduct experiments using four real-world datasets to evaluate the performance of these algorithms. The comparison is based on six metrics and objectives: the average number of mined rules, the average values of confidence, support, comprehensibility, interestingness of the rules, and the average time required to execute the algorithms.

### E. CONTRIBUTIONS

The investigations conducted in this article build upon the data gathered from a preliminary study on the performance of SI-based NARM algorithms [12]. This analysis serves the purpose of bridging the existing gaps between optimization algorithms and developing an advanced framework for generalized association rule mining [13].

The following are the key contributions of this article.

1) Investigating the role of multi-objective optimization algorithms, with a focus on SI-based optimization algorithms, in the context of NARM.
2) Conducting an exhaustive multi-aspect analysis of SI-based algorithms, utilizing four real-world datasets and six key metrics and objectives: performance time, number of rules, support, confidence, comprehensibility, and interestingness.
3) Providing efficient utilization and understanding of four popular SI-based optimization algorithms (MOPAR, MOCANAR, ACO-R, and MOB-ARM) specifically tailored for NARM.
4) Providing a comparative analysis of the following four SI-based optimization algorithms (MOPAR, MOCANAR, ACO-R, and MOB-ARM) in relation to the traditional ARM algorithm (Apriori) with regard to NARM.
5) Contributing towards the advancement of NARM by exploring the potential of multi-objective optimization algorithms and addressing the associated challenges.

### F. RESULTS

In our evaluations, the MOPAR algorithm demonstrates a notable advantage in terms of producing a low number of rules with high confidence, comprehensibility, and interestingness. However, it requires parameter modifications when dealing with datasets that have a large number of dimensions. On the other hand, the MOCANAR algorithm consistently delivers reliable results across all six parameters for all datasets. The ACO-R algorithm generates high-quality rules but necessitates parameter adjustments for datasets with numerous attributes. Conversely, the MOB-ARM algorithm exhibits significantly slower performance compared to the other algorithms. Based on this analysis, it is evident that different SI-based optimization algorithms for NARM cater to different requirements.

The paper is structured as follows. In Sect. II, related work is given. Sect. III highlights the background information to understand the subject. We discuss the SI-based algorithms in Sect. IV. Sect. V outlines the experimental results and multi-aspect analysis of the four SI-based algorithms and one traditional Apriori algorithm. Sect. VI provides the challenges and future directions for the algorithms. The conclusion is given in Sect. VII.

## II. RELATED WORK

In data mining [14], ARM is a well-known technique to find interesting relations among various data items. Agrawal [15] introduced ARM in 1993 to discover the associations between data items in market basket analysis. Later, some essential

algorithms, such as Apriori [16] and FP-Growth [17], were proposed. These algorithms were suitable for binary data but could not deal with numerical data. In 1996, Srikant introduced the concept of quantitative association rule mining (QARM) [11] to deal with numerical data. Further, this technique is also known as NARM [1]. Several methods, such as optimization, discretization, and distribution, are available in the literature to solve the problem of NARM [1] and [2]. The optimization method seems to be a potential solution to deal with such complex problems. Evolutionary-based and SI-based algorithms come under the optimization method [3]. Recent NARM optimization algorithms also cover SI-based algorithms, which are based on animal and insect movements [18] and the biological behavior of natural objects [5]. In recent decades, bio-inspired computation [19] has been one of the most researched subfields of artificial intelligence. SI-based algorithms are the subcategory of nature-inspired algorithms. Particle swarm optimization (PSO) [20], ant colony optimization (ACO) [21], cuckoo search (CS) [22], bat-inspired algorithm (BA) [23], crow search (CSA) [24], wolf search (WSA) [25] are some examples of various SI algorithms. The variants of these algorithms were used for solving NARM problems. Such as, in 2008, Alatas and Akin [26] used the PSO algorithm for mining the association rule with numeric attributes. The PSO was modified to search the numeric attributes' intervals and discover the numeric association rules. Further, Coello et al. [27] extended PSO to handle multi-objective issues. In the same way, Ledmi et al. [28] used the crow search-based algorithm for NARM. A multi-objective PSO technique using an adaptive archive grid for NARM was proposed by Kuo et al. [29]. It is based on the Pareto-optimal technique as well. Recently, Stupan and Fister [30] presented a minimalistic framework NiaARM for NARM, which is the extended version of the ARM-DE [31] algorithm. Users can preprocess their data using the NiaARM framework and use a variety of interest measures. The numerical association rule miner proposed by Fister et al. [32] combines the offline uARMSolver [33], which is part of the Red AI class, with the recently created OnlineNARM miner, which is part of the new Green AI. In the literature, a performance analysis of several NARM algorithms was conducted. Varol Altay and Alatas [1] analyzed the performance of seven evolutionary and fuzzy evolutionary NARM algorithms. The chosen algorithms were also compared against the apriori algorithm. A comparative analysis was done in terms of support, confidence, the number of rules mined, the number of records covered, and time spent using eleven real-world datasets. This research found that the evolutionary algorithms have better results in terms of support, confidence, and time metrics. The authors also performed a performance analysis of multi-objective evolutionary NARM algorithms in [34]. In this research, six multi-objective and four single-objective optimization algorithms were chosen to be compared. The number of rules, coverage percentage, support, confidence, conviction, lift, netconf, ylesQ, and certain factor measures were used for comparative analysis. Ten real-world datasets were used. This research found that multi-objective algorithms outperformed single-objective algorithms in terms of support, lift, certain factors, netconf, and yulesQ metrics. The measures netconf and yulesQ are generally used to quantify the rule's level of interest or relevance to the user.

An example of using NARM for real-world problems was presented in [35], which performed an association analysis of multi-objective NARM algorithms using data about Parkinson's disease. This research used numerical data consisting of speech samples related to Parkinson's disease. This data was used on three multi-objective NARM algorithms to find association rules related to healthy individuals and patients with Parkinson's disease.

Another example of using NARM for real-world problems was presented in [36], which presented an association analysis of multi-objective NARM algorithms using data about liver fibrosis. This data was used on two multi-objective NARM algorithms to find association rules related to liver fibrosis. In this article, a sensitivity analysis was done to find the best parameters for this problem. A recent exhaustive review of more than five hundred nature-inspired metaheuristic algorithms and a performance assessment of fifteen algorithms has been conducted in [37].

## III. BACKGROUND

### A. ASSOCIATION RULE MINING

ARM aims to extract interesting correlations, frequent patterns, or associations among sets of items in mainly transactional databases. One application of ARM is to find out what products are bought together from a store [15]. The discovered association rules can help determine how to boost the sales of a product, what products may be impacted by the discontinuation of another product, and the best locations for the products. Let $I = \{i_1, i_2, i_3, \ldots i_m\}$ be a set of different $m$ data items and $D$ be a set of transactions where each transaction $T$ contains a non-empty set of items, $T \subseteq I$. A transaction $T$ contains $X$, which is a set of some items in $I$ if $X \subseteq T$. An association rule is an if-then relationship and denoted by $X \Rightarrow Y$ that has an antecedent $X$, and a consequent part $Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \phi$ [16]. Support and confidence are the most commonly used measures in ARM. The support of an itemset $X$ determines how frequently the itemset appears in a transactional database. The confidence of an association rule $X \Rightarrow Y$ determines how frequently items in $Y$ appear in transactions that contain $X$.

### B. NUMERICAL ASSOCIATION RULE MINING

NARM came into the scenario to extract association rules from numerical data. Unlike a classical ARM, a numerical ARM allows attributes to be either categorical (e.g., gender, education) or numeric (e.g., salary, age) rather than just Boolean. A Numerical association rule is an implication of the form $X \Rightarrow Y$, in which both antecedent and consequent parts are the set of attributes in the forms $A = \{v_1, v_2, \ldots v_n\}$

if $A$ is a categorical attribute, or $A \in [v_1, v_2]$ if $A$ is numeric attribute.

An example of a numerical association rule is given in (1). Here, Support is 10% and Confidence is 70%. This rule states that "10% of the employee are males aged between 25 and 35, and their salary would be between \$2,000 and \$2,500" while "70% of males aged between 25 and 35 are earning between \$2,000 and \$2,500."

$$\text{Age} \in [25, 35] \wedge \text{Gender: [Male]} \Rightarrow \text{Salary} \in [2,000, 2,500] \tag{1}$$

Here, Age and Salary are numerical attributes, and Gender is a categorical attribute. In ARM, except for support and confidence, more than fifty measures of interestingness are available in the literatures [38], [39]. This article mainly uses support, confidence, comprehensibility, and interestingness measures.

The support of an association rule $X \Rightarrow Y$ is calculated as the percentage of transactions of the total records containing both itemsets $X$ and $Y$, shown in (2).

$$\text{Support}(X \Rightarrow Y) = \frac{|(X \cup Y)|}{|D|} \tag{2}$$

The confidence of a rule, $X \Rightarrow Y$, is described as the percentage of transactions that contain itemset $X$ also contain itemset $Y$, shown in (3).

$$\text{Confidence}(X \Rightarrow Y) = \frac{|(X \cup Y)|}{|X|} \tag{3}$$

According to [40], if the number of conditions involved in the antecedent part is less than the consequent part, the rule is more comprehensible. Eq. (4) is used to calculate the comprehensibility of an association rule. Here, $|Y|$ represents the number of attributes in the consequent part of the rule, and $|X \cup Y|$ shows the number of attributes in both the antecedent and consequent parts of the rule.

$$\text{Comprehensibility}(X \Rightarrow Y) = \frac{\log(1 + |Y|)}{\log(1 + |X \cup Y|)} \tag{4}$$

The interestingness measure is focused on discovering hidden information by extracting interesting rules. Eq. (5) consists of three parts; the first part shows the probability of generating the rule based on the antecedent part, the second part shows the probability based on the consequent part and the third part shows the probability of not generating the rule based on the whole dataset.

$$\text{Interestingness}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)} \times \frac{\text{Support}(X \cup Y)}{\text{Support}(Y)} \times \left(1 - \frac{\text{Support}(X \cup Y)}{|D|}\right) \tag{5}$$

#### 1) MULTI-OBJECTIVE NARM

A single objective optimization problem has just one objective function; however, when many objective functions are used, the process is referred to as multi-objective [41]. Multi-objective optimization aims to balance several conflicting performance measures by using a set of non-dominated solutions [42]. The weighted sum and Pareto dominance are two methods for solving multi-objective optimization problems. The weighted sum method is a classical multi-objective method that summarizes multiple objectives into a single objective by multiplying each objective with a pre-defined weight. Traditional evolutionary algorithms optimize the resulting single-objective function. It is the simplest multi-objective method, but finding suitable multipliers can be challenging. However, all the objectives are evaluated simultaneously in the Pareto dominance method [43]. One solution dominates another if it improves one objective without causing a worse outcome for all the other objectives. Using this dominance criterion, non-dominated solutions can be defined.

```
Step 1: Initialize the population
Step 2: Evaluate solutions
Step 3: For iteration in max iterations
Step 4: Modify solutions
Step 5: Evaluate modified solutions
Step 7: Return the best solutions
```

**LISTING 1. Pseudo code of nature-inspired meta heuristic algorithm.**

## IV. SWARM INTELLIGENCE OPTIMIZATION ALGORITHMS

Optimization methods offer a robust and efficient approach to tackling large search spaces, and they can be classified into biology-inspired and physics-based methods. Among the biology-inspired methods, SI and evolution-based algorithms are particularly prevalent and widely used [1], [2]. It is important to note that SI-based algorithms fall within the category of bio-inspired algorithms, which in turn belong to the broader category of nature-inspired algorithms [44].

As stated in Bonabeau et al. [45], SI-based optimization methods draw inspiration from the collective intelligence and group behavior observed in self-organized groups, such as swarms of birds, fish, honey bees, and ant colonies. These algorithms consist of individuals who navigate the search space through simulated interactions. Various SI-based algorithms have gained popularity for addressing several optimization problems, including recent advancements in solving NARM problems. The most popular SI-based algorithms for NARM are PSO and ACO. The Bat Algorithm and the Cuckoo Search Algorithm are also part of the family.

Below is an example of pseudocode List 1 for a nature-inspired meta-heuristic algorithm, outlining the general steps. In this algorithm, a population of agents is initialized with random solutions. The solutions are evaluated based on the defined objectives. Subsequently, each agent modifies its solution iteratively until a specified stopping criterion is met. Finally, the best-generated solutions are returned.

The following two key elements must be addressed when employing a nature-inspired population-based algorithm to solve the ARM.

1) The representation of solutions in the search space
2) Fitness function assessment

The first element involves describing the encoding of solutions in the search space, while the second element focuses on assessing the quality of solutions.

A solution must be encoded to mine numerical association rules in the search space. There are two well-known approaches to representing individuals: Michigan and Pittsburgh. When using the Michigan approach for representing individuals, each individual encodes a single association rule, while in the Pittsburgh approach, each individual encodes a set of association rules [5]. The Michigan approach is comparatively better than the Pittsburgh approach for finding high-quality rules. In the Michigan approach, different types of individual representation are identified.

The first representation of encoding the association rules in NARM is shown in (6). The rule is encoded as a vector of attributes with $n$ number of triplets, where $n$ is the total number of attributes in the transactional database. Each triplet consists of three elements. Here, ACN determines whether the attribute is present in the antecedent or consequent of the rule or not present in the rule at all. Furthermore, $l$ determines the lower bound of the attribute and $u$ determines the upper bound of the attribute [5].

$$(\text{ACN}_1, l_1, u_1), \ldots, (\text{ACN}_n, l_n, u_n)) \tag{6}$$

Another way to represent a rule as a vector is shown in (7). Here, $s$ represents the value, and $\delta$ represents the standard deviation of the attribute [8].

$$((\text{ACN}_1, s_1, \delta_1), \ldots, (\text{ACN}_n, s_n, \delta_n)) \tag{7}$$

The ACN element can be encoded in two different ways. In the first way, if the $\text{ACN}_j$ (where j = 1 to n) value is less than or equal to $1/3$, then the attribute is in the antecedent part of the rule. If the value is greater than $1/3$ and smaller than or equal to $2/3$, then the attribute is in the consequent part. If the value is greater than $2/3$, then the attribute is not present in the rule.

The second way to encode is used in [10]. Cuckoos in ARM are represented using a 2D array. The array has $n$ columns, indicating the number of features in the dataset, and 3 rows. The first row indicates the feature's location in the current association rule, with 0 meaning the feature is absent, 1 denoting it is in the antecedent, and 2 indicating it is in the consequent. The second and third rows represent the lower and upper bounds of the feature values in the association rule, respectively.

Another way to represent an association rule is used in [9]. Each solution $X$ represents a rule with vector $S$ and comprises $k$ items, resulting in a total of $k + 1$ positions. In this vector, $S[0]$ functions as the separator between the antecedent and the consequent of the rule. If the $i_{th}$ item in the database is included in the rule, then position $k$ contains $i$; otherwise, the position contains zero, where $0 < k \leq n + 1$, with $n$ being the total number of items in the database. For example, with the set of items $I = i_1, i_2, \ldots, i_{10}$, the rule $X_1 = 3, 1, 5, 0, 6, 2, 0, 0, 7, 0, 0$ represents the rule $i_1, i_5 \Rightarrow i_6, i_2, i_7$.

### A. PARTICLE SWARM OPTIMIZATION ALGORITHM (PSO)

PSO is the most popular optimization method for continuous non-linear functions, which simulates the movement of bird flocks or fish schools [20], [46]. Just as bird flocks move around in search of food in the sky and adjust their speed and position based on the group's direction and food availability, PSO artificially replicates this behavior.

A swarm consists of $N$ particles that traverse a $D$-dimensional search space. During the search, each particle continually adjusts its position towards the global optimum by considering two factors: its personal best position (referred to as pbest), which is the best position it has encountered so far, and the global best position (referred to as gbest), which is the best position found by the entire swarm [26]. The velocity and position of the particles are calculated iteratively to find the optimum solution.

#### 1) MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION ALGORITHM FOR ASSOCIATION RULES MINING (MOPAR)

The Multi-Objective Particle swarm optimization algorithm for Association Rules Mining (MOPAR) is a multi-objective PSO (MOPSO) algorithm based on Pareto optimality for extracting numerical association rules in one step. The algorithm utilizes three objectives: confidence, comprehensibility, and interestingness.

The algorithm's steps are based on the research conducted by Beiranvand et al. [7] and are presented in detail in Algorithm 1. In the algorithm, each candidate rule is represented as a particle $X_i(t)$, where $i$ ranges from 1 to $n$, by an $m$-dimensional vector. The $i_{th}$ particle $X_i(t)$ at time $t$ can be expressed as $X_i(t) = [x_{i,1}(t), \ldots, x_{i,m}(t)]$, where $x_{i,k}(t) \epsilon [Lj, Uj]$, with $1 \leq j \leq m$, indicating the position of the $i_{th}$ particle for the $k_{th}$ attribute or dimension.

The velocity of the $i_{th}$ individual at iteration $t$ is defined as an m-dimensional vector $V_i = (-v_{i,1}, v_{i,2}, \ldots, v_{i,m})$, where $v_{i,k}$ denotes the velocity component of the $i_{th}$ particle concerning the $k_{th}$ dimension. The velocity of the $i_{th}$ particle is constrained to a maximum velocity $v_{imax} = (v_{imax,1}, v_{imax,2}, \ldots, v_{imax,m})$, where $i$ ranges from 1 to $n$ [7].

The population consists of particles, the external repository holds the mined rules, and the global best (the best particle) is initialized. In each iteration, the particle population is updated. Subsequently, the best solutions from the population are added to the external repository, and the global best solution is updated. Finally, after the iterations, the external repository is returned. To update particles, (8) and (9) are used to update the velocities and positions of a particle. After that, the particle's objectives are evaluated. Finally, the local best solution of each particle is updated using Pareto dominance.

$$\begin{aligned} v_{i,k}(t+1) = {} & w(t) v_{i,k}(t) + c_{\text{local}} R_{\text{local}} \left(\text{lbest}_{i,k}(t) - x_{i,k}(t)\right) \\ & + c_{\text{global}} R_{\text{global}} \left(\text{gbest}_{i,k}(t) - x_{i,k}(t)\right) \end{aligned} \tag{8}$$

$$x_{i,k}(t+1) = x_{i,k}(t) + v_{i,k}(t+1) \tag{9}$$

To determine the global best solution, roulette wheel selection is employed. Initially, the roulette wheel assigns

**Algorithm 1** MOPAR Algorithm

**Input:** Data, population size, maximum iterations, external repository size, $c_{\text{local}}$, $c_{\text{global}}$, inertia weight, xRank
**Output:** External repository

1: Initialize population.
2: Evaluate the objectives of the generated rule.
3: Initialize external repository.
4: Initialize global best.
5: Update the velocities of particles.
6: Update the positions of particles and evaluate the objectives of the new rules.
7: Update the non-dominated local set of each particle. Update local best.
8: Update external repository. If the size of the external repository is larger than the external repository size, remove particles that dominate more rules.
9: Update global best using the roulette wheel selection. If the maximum number of iterations is not reached, go to Step 2.
10: **return** the external repository.

a rank $r_i$ to each particle using (10), with $x_{\text{rank}}$ denoting a user-specified parameter, and $n_{\text{local\_dominated}}$ is the local dominated count representing the number of local best solutions dominated by the current solution. Subsequently, each particle $i$ is assigned a probability $P_i$ according to (11). Based on these probabilities, a particle is selected.

$$r_i(t) = \frac{x_{\text{rank}}}{n_{\text{local\_dominated}}} \tag{10}$$

$$P_i(t) = \frac{r_i(t)}{\sum_{k=1}^{n} r_k(t)} \tag{11}$$

The MOPAR algorithm introduces a modified version of the MOPSO algorithm, which includes redefined lbest (local best) and gbest (global best) particles, as well as a selection procedure designed to address the challenges of NARM. In this algorithm, each particle is represented in a similar manner to the rough particle swarm optimization algorithm (RPSOA) and incorporates lower and upper bounds for intervals.

### B. CUCKOO SEARCH ALGORITHM

The cuckoo search algorithm (CSA), proposed by Yang and Deb in 2009 [22], draws inspiration from the brooding parasitic behavior observed in cuckoo species. Unlike other birds, cuckoos do not construct nests but instead lay their eggs in the nests of other bird species, employing their remarkable ability to mimic the color and pattern of the host birds' eggs. Some host birds may detect the presence of foreign eggs and either discard them or abandon their nests. The CSA follows three key rules. Firstly, cuckoo birds lay only one egg at a time, selecting a nest at random. Secondly, nests containing high-quality eggs have a higher chance of survival and are carried over to the next generation. Lastly, the number of host nests remains fixed, and the probability of a host bird discovering cuckoo eggs is either 0 or 1. If a host bird identifies a cuckoo egg, it can choose to destroy the egg or abandon the nest.

In the context of ARM, Kahvazadeh and Abadeh [10] extended the concept of the multi-objective cuckoo search algorithm to tackle NARM using a Pareto-based approach. In this algorithm, each egg in the nest represents a solution, and a new solution is introduced when a cuckoo lays an egg. The objective is to utilize these new and potentially superior solutions to replace less promising solutions within the nest.

#### 1) MULTI-OBJECTIVE CUCKOO SEARCH ALGORITHM FOR NUMERICAL ASSOCIATION RULE MINING (MOCANAR)

The multi-objective cuckoo search algorithm for numerical association rule mining (MOCANAR) proposed by Kahvazadeh and Abadeh [10] is a powerful method that utilizes Pareto principles to extract high-quality association rules from datasets with numeric attributes. Inspired by the brooding parasitic behavior of cuckoo species, this algorithm mimics the natural process in which cuckoos lay their eggs in other bird species' nests. In MOCANAR, the ARM problem is represented using a $2D$ array, which serves as the "cuckoos" in the algorithm. The columns of the array correspond to the attributes present in the dataset, while the number of rows is fixed at three. The first row of the array indicates the location of each attribute within the association rule. A value of 0 signifies that the attribute is not included in the rule, a value of 1 indicates that the attribute belongs to the antecedent part of the rule, and a value of 2 indicates that the attribute belongs to the consequent part of the rule. The second and third rows of the array represent the lower and upper bounds of the corresponding attribute, respectively. The parameters $w_1$, $w_2$, and $w_3$ determine the length of steps in the three rows of the cuckoo, guiding its movement toward a better position. Higher values for these parameters result in an increase in step length, facilitating faster convergence due to the larger steps taken.

In MOCANAR, multiple objectives are considered to guide the search for high-quality association rules. These objectives include support, confidence, interest, and comprehensibility, which collectively contribute to the evaluation of the rules. The rules are retrieved incrementally during the algorithm's iterations, with a focus on generating a small number of high-quality rules at each step.

The extraction of non-dominated rules is achieved through Pareto optimality. The algorithm's steps, based on the work by Kahvazadeh and Abadeh [10], are presented in Algorithm 2.

The algorithm begins by initializing the population, which consists of cuckoos and the current set of non-dominated rules. Each increment involves the initialization of the population. Within each generation, random cuckoos are generated and directed towards the best solution, employing the levy flight policy proposed by Yang and Deb [22]. The worst cuckoos in the population are replaced with these newly generated cuckoos. Subsequently, each cuckoo generates an

**Algorithm 2** MOCANAR Algorithm Steps

**Input:** Data, population size, number of increments, maximum generations, pa, pmut, number of tournaments, number of random cuckoos, $w_1$, $w_2$, $w_3$

**Output:** Final non-dominated rules

1: Initialize the population and cuckoo eggs. Evaluate the objectives of the generated rules.
2: Generate random cuckoos and direct them towards the best cuckoo in the population using the levy flight policy. Replace the worst cuckoo in the population with the directed cuckoo.
3: Generate cuckoo eggs by directing all cuckoos in the population towards the best cuckoo, which is chosen through a tournament selection process.
4: Eliminate a percentage of the worst eggs based on the support measure. Form a new population by selecting cuckoos with the best objective measures.
5: Merge the population and non-dominated lists, removing any duplicated rules. Assign the non-dominated rules from the merged list to the non-dominated list.
6: If the maximum number of generations is not reached, go to step 2.
7: Add the rules from the non-dominated list to the final non-dominated list.
8: If the maximum number of increments is not reached, go to step 1.
9: Remove duplicated and dominated rules from the final non-dominated list.
10: **return** the final non-dominated list.

egg using levy flight. At the end of each generation, the current set of non-dominated rules is updated. This process continues until the final increment, where the final set of non-dominated rules is updated. Finally, the algorithm returns the final non-dominated rules.

To choose the best solution when generating eggs, a tournament selection process is employed. A certain number of tournament cuckoos are randomly selected from the population, and a random non-dominated solution is chosen from this selection. A levy flight policy is utilized to guide the movement of cuckoos towards the best cuckoo. For each attribute of a source cuckoo's rule, three-step sizes are calculated using the levy distribution and a target cuckoo. These step sizes are then used to modify the rule of the source cuckoo.

To generate a new population, an elimination process is performed on a percentage of eggs with the worst support measure. Afterward, the eggs and the cuckoo population are merged into a temporary population. The temporary population is sorted based on the support measure, and the top 1/4 of the highest-ranking solutions are added to the new population. The same process is repeated for the remaining measures, resulting in the formation of a new population.

### C. ANT COLONY OPTIMIZATION ALGORITHM

Ant colony optimization (ACO) is based on the foraging behaviour of various ant species. Ants begin to investigate the area around the nest at random and eventually find some food sources. Based on the quantity and quality of food, these ants deposit chemical pheromones on the ground to suggest the desired path for colony members to follow on their return trip [21]. In ACO, a group of artificial ants develops solutions to the optimization problem and communicates information about the quality using a communication mechanism similar to real ants.

There are several variations and extensions of the ACO algorithm that have been developed to address different problem domains. The commonly used ACO algorithm for NARM is ACO-R.

#### 1) ANT COLONY OPTIMIZATION FOR CONTINUOUS ATTRIBUTES (ACO-R)

The ACO-R algorithm is a variant of ACO specifically designed for retrieving association rules from numerical attributes without minimum support and minimum confidence thresholds. ACO-R differs from ACO in its approach to probability distributions and pheromone storage.

In ACO-R, a probability density function is used instead of a discrete probability distribution to describe the pheromone distribution over the search space. The solution archive size, denoted as $k$, is used to store the pheromone information. The solution archive is represented as a matrix, where each entry is denoted by $s_j^i$. Here, $i = 1, 2, \ldots, n$ corresponds to the number of dimensions, and $j = 1, 2, \ldots, k$ represents the number of rows in the matrix.

The ants in ACO-R move across the archive by selecting one row based on its associated weight ($\omega$). A new solution is then generated by sampling the Gaussian function ($g$) of the values of each dimension in the selected solution.

The numeric attributes are represented as dimensions of the solution archive, divided into three sections. Each complete solution in the archive represents a numeric association rule. The first part of the solution represents the antecedent or consequent of the rule, the second part represents the value, and the third part indicates the standard deviation, which is used to construct numeric attribute intervals.

ACO-R utilizes Gaussian functions to identify attribute intervals that correspond to interesting rules, with the function determining the frequency and length of the intervals. The objective function in ACO-R consists of four components, as defined in (14). The first component represents the support for the rule, indicating its importance. The second component represents the confidence value. The third component represents the number of attributes in the rule. The last component penalizes the amplitude of the intervals that adhere to the itemset and rules.

The pheromone update technique in ACO-R involves adding a number of new solutions generated by the ants and removing an equal number of poor solutions from the archive.

**Algorithm 3** ACO-R Algorithm Steps

**Input:** Data, archive size, ant colony size, maximum iterations, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, q, e$

**Output:** Archive

1: Initialize and sort the archive.
2: Initialize weights, probabilities, and ants.
3: Each ant chooses a solution and generates a new solution by sampling a Gaussian function. Evaluate the objectives of the new solution.
4: If the number of ants that have generated a new rule has not reached the ant colony size, go to step 3.
5: Add the solutions generated by the ants to the archive, then sort and cut off the archive to the specified size.
6: If the maximum number of iterations has not been reached, go to step 2.
7: Remove duplicated rules from the archive.
8: **return** Non-dominated rules from the archive.

The solutions are ranked to maintain the best solutions at the top of the solution archive. In each execution of ACO-R, the best solution can be considered as a rule.

The algorithm follows the steps outlined in Algorithm 3 based on Qodmanan et al. [8].

Initially, the archive is initialized with a set of solutions, and these solutions are ranked based on their objective values. In each iteration of the algorithm, the weights and probabilities of the solutions in the archive are calculated. Each ant selects a solution from the archive based on the assigned probabilities and generates a new solution by sampling a Gaussian function. At the end of each iteration, the solutions in the archive are ranked again, and the worst solutions are removed from the archive. This process helps maintain a diverse and high-quality set of solutions in the archive. After completing the specified number of iterations, the final archive, which consists of the remaining solutions, is returned as the output of the ACO-R algorithm.

The interval objective, shown in (12), favours rules with smaller intervals. Here, $n$ is the number of attributes, maxbound and minbound are the maximum and minimum values for the attribute in the database. $\mathrm{UB}_i$ and $\mathrm{LB}_i$ are the upper and lower bounds of an attribute in the rule. The upper and lower bound of the intervals can be calculated by adding a coefficient of a standard deviation to the value of solution $s_j^i$ using (13).

$$\text{Interval} = \sum_{i=0}^{n} \frac{(\mathrm{UB}_i - \mathrm{LB}_i)}{\text{maxbound}_i - \text{minbound}_i} \tag{12}$$

$$\mathrm{UB}_i = s_j^i + \alpha_5 \cdot \sigma \quad \text{and} \quad \mathrm{LB}_i = s_j^i - \alpha_5 \cdot \sigma \tag{13}$$

All the mentioned objectives are put together into a single objective function, shown in (14). Here $\alpha_1, \alpha_2, \alpha_3$, and $\alpha_4$ are user-specified input parameters and one might increase or decrease the effects of parts of function.

$$\begin{aligned}\text{objective} = \alpha_1 \cdot \text{Support} + \alpha_2 \cdot \text{Confidence} - \alpha_3 \\ \cdot \text{Interestingness} - \alpha_4 \cdot \text{Interval}\end{aligned} \tag{14}$$

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(j-1)^2}{2q^2k^2}} \tag{15}$$

To calculate the weight $\omega_j$ of a solution $S_j$, (15) is used: In (15), $k$ represents the number of solutions in the archive, $j$ is the rank of the solution, and $q$ is a user-specified parameter. The weight $\omega_j$ is determined based on the rank of the solution. A smaller value of $q$ leads to a higher preference for the best-ranked solutions, while a larger value of $q$ results in a more uniform probability distribution.

To calculate the probability $p_j$ of choosing solution $S_j$, (16) is used: In (16), the weight $\omega_j$ of the solution $S_j$ is divided by the sum of the weights of all the solutions in the archive. This calculation yields the probability $p_j$, which represents the likelihood of choosing solution $S_j$ during the solution selection process.

$$p_j = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r} \tag{16}$$

After an ant chooses a solution based on the probabilities, a new solution is sampled using (17): In (17), $\mu_{\text{new}}$ represents the new value of the chosen solution, $\mu$ represents the value of the chosen solution and $\delta$ is calculated using (18).

$$P(x) = g(x, \mu, \delta) = \frac{1}{\delta\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\delta^2}} \tag{17}$$

In (18), $\delta$ represents the perturbation value added to the chosen solution, $\xi$ is a user-specified parameter that controls the magnitude of the perturbation, $k$ is the number of solutions in the archive, and $s_r^i$ and $s_j^i$ are the values of the solutions $S_r$ and $S_j$ respectively.

$$\delta = \xi \sum_{r=1}^{k} \frac{s_r^i - s_j^i}{k-1} \tag{18}$$

### D. BAT ALGORITHM

The BAT algorithm (BA) was introduced by Yang in 2010 as a method for solving continuous constrained optimization problems inspired by the echolocation behavior of microbats [23]. Microbats possess the remarkable ability to use echolocation to sense distances. By emitting high-frequency sound pulses and listening to the echoes reflected from objects in their environment, they can navigate, locate prey, avoid obstacles, and find shelter even in darkness.

The BAT algorithm mimics this behavior by employing a population of virtual bats. Each bat represents a potential solution to the optimization problem. Similar to microbats, virtual bats have attributes such as position, velocity, frequency, wavelength, and loudness. During the optimization process, the bats explore the search space by adjusting their positions and velocities based on their current locations and the locations of the best solutions discovered so far.

**Algorithm 4** MOB-ARM Algorithm Steps

**Input:** Data, population size, iterations, Pareto points, $\alpha$, $\beta$, $\gamma$, $\delta$, minimum support, minimum confidence **Output:** Non-dominated solutions

1: Initialize the population and sort it. Initialize the global best solution. Initialize the list of non-dominated solutions.
2: Initialize the weights.
3: Update the frequency and velocity of each bat in the population and generate a new rule.
4: If a randomly generated number is greater than the bat's rate, modify one attribute in the new rule.
5: Check and fix the rule. Evaluate its fitness.
6: If the new objective value is better than the old one, accept the new rule, increase the bat's rate, and decrease its loudness.
7: Sort the population and update the global best solution.
8: If the maximum number of iterations is not reached, go to step 3.
9: Add the best solutions to the list of non-dominated solutions.
10: If the desired number of Pareto points is not reached, go to step 2.
11: **return** Non-duplicated rules from the list of non-dominated solutions.

The frequency of a bat remains fixed, while its wavelength and loudness change dynamically. The wavelength affects the step size of the bat's movements, while the loudness influences the rate at which the bat emits sound pulses.

BAT algorithm has also been applied to ARM tasks involving categorical attributes [47]. By adapting the principles of echolocation and the bat's behavior, the BAT algorithm offers a unique approach to discovering association rules in datasets containing categorical attributes.

#### 1) MULTI-OBJECTIVE BAT ALGORITHM FOR NARM (MOB-ARM)

The multi-objective bat algorithm for NARM (MOB-ARM) proposed by Heraguemi et al. in [9] is inspired by the behavior of microbats. The algorithm aims to optimize two global objective functions to extract interesting rules using four quality measures: support, confidence, comprehensibility, and interestingness. The first objective function combines support and confidence measures, as shown in (21). It represents the importance of the association rule based on the rule's support and confidence values. The second objective function combines comprehensibility and interestingness measures, as shown in (22). It captures the trade-off between generating comprehensible rules and maximizing their interestingness. The algorithm follows three main steps: initialization, searching for non-dominance solutions for the Pareto points, and searching for the best solution for each bat at the Pareto point. The rule encoding in this algorithm follows the Michigan approach. Bats are initialized with random frequencies and velocities.

Prior to applying the algorithm, the input data is discretized into intervals. The algorithm utilizes a weighted sum approach to determine the best solutions. The steps of the algorithm are outlined in Algorithm 4, as given by Heraguemi et al. [9]. The algorithm begins by initializing a population consisting of bats. In each iteration, objective weights are generated, and the frequency, velocity, and rules of each bat are updated. At the end of each iteration, the bats are ranked, and a new global best solution is selected. After each iteration, the best solution of each bat is recorded as a non-dominated solution. Finally, the non-dominated solutions are returned as the output of the algorithm. To generate the objective weights, (19) is utilized. In this equation, the value of $k$ represents the number of objectives used, which in the case of MOB-ARM is two. These weights are employed in the calculation of an objective measure, as depicted in (19), which incorporates the two objectives.

$$\sum_{k=1}^{k} w_k = 1 \tag{19}$$

$$\text{Obj(R)} = w_1 \cdot \text{Obj}_1\text{(R)} + w_2 \cdot \text{Obj}_2\text{(R)} \tag{20}$$

The objective measure in the MOB-ARM algorithm considers two distinct objectives, each computed using (21) and (22). These equations incorporate user-defined parameters $\alpha$, $\beta$, $\gamma$, and $\delta$ as weights assigned to the support, confidence, comprehensibility, and interestingness measures, respectively and $R$ is a rule $X \Rightarrow Y$. By adjusting these weights, the algorithm can prioritize different aspects of rule quality during the optimization process.

$$Obj_1(R) = \alpha \cdot \text{Confidence}(R) + \frac{\beta \cdot \text{Support}(R)}{\alpha} + \beta \tag{21}$$

$$\text{Obj}_2(R) = \gamma \cdot \text{Comprehensibility}(R) + \frac{\delta \cdot \text{Interestingness}(R)}{\gamma} + \delta \tag{22}$$

To update a bat's frequency and velocity in the MOB-ARM algorithm, (23) and (24) are employed. The new frequency is determined by computing the maximum frequency, which corresponds to the number of attributes in the dataset. Subsequently, the new velocity is computed using the maximum frequency, the new frequency, and the previous velocity. These update equations enable the bats to dynamically adjust their exploration and exploitation capabilities during the optimization process (here, $i$ is the index for individual bats within the population).

$$f_i^t = 1 + (f_{max})\beta \tag{23}$$

$$v_i^t = f_{max} - f_i^t - v_i^{t-1} \tag{24}$$

$$A_i^{t+1} = \alpha A_i^t \tag{25}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{26}$$

**TABLE 1. Datasets used in the experiments.**

| Dataset | #records | #attributes | Description |
|---|---|---|---|
| Basketball | 96 | 5 | This dataset includes a variety of numerical attributes related to the performance of basketball teams and players to identify patterns and relationships. |
| Quake | 2178 | 4 | This dataset is used to demonstrate the use of various smoothing techniques in statistics. The dataset contains a time series of the number of earthquakes that occurred in California between the years 1980 and 1984. |
| Fat | 225 | 18 | This dataset contains the percentage of body fat, age, weight, height, and 10 measurements of body circumference (such as the abdomen) for a total of 252 men. |
| Longley | 16 | 7 | The Longley dataset comprises a number of strongly collinear US macroeconomic indicators. It has been used to assess the precision of least squares methods. |

**TABLE 2. Algorithmic parameters used in the experiment.**

| Algorithms | Parameters |
|---|---|
| MOPAR [7] | Population size: 50, iterations: 200, external repository size: 50, inertia weight: 0.63, velocity: 3.83, xRank: 13.3, c1:2, c2: 2 |
| MOCANAR [10] | Population size: 50, generations: 200, increments: 1, randomcuckoo: 1, tournament: 30, Pa: 0.3, P_mut: 0.05, w1: 0.2, w2: 0.5, w3: 0.3 |
| MOB-ARM [9] | Population size: 50, iterations: 40, Pareto points: 5, alpha: 0.4, beta: 0.3, gamma: 0.2, delta: 0.1, minsupp: 0.2, minconf: 0.5 |
| ACO-R [8] | Ant colony size: 50, iterations: 200, archive size: 50, alpha1, alpha2: 4, alpha3, alpha5: 1, alpha4: 0.001, Q: 0.1, E: 0.85 |

In the MOB-ARM algorithm, a new rule is generated using a method proposed in [47]. The generation process takes into account the velocity, frequency, and loudness of the bat. The velocity determines the starting position in the rule where changes will be made, while the frequency determines the number of attributes that will be modified. The algorithm introduces variability and exploration by randomly modifying rule attributes, ensuring a diverse search in the solution space.

If the new rule's objective is superior to the previous objective, the rule is accepted, and the bat's loudness and rate are updated. The loudness is decreased according to (25), while the rate is increased using (26). In (26), $r_i^0$ represents the initial rate of the bat, and $t$ denotes the current iteration, as described by Yang [23].

## V. EXPERIMENTAL RESULTS

To assess the performance of the MOPAR, MOCANAR, ACO-R, MOB-ARM and Apriori algorithms, four real-world datasets from Guvenir et al. [48] are chosen. Table 1 provides a detailed description of these datasets, which vary in the number of records and attributes. This diversity allows for a comprehensive evaluation of the algorithms' effectiveness in handling different dataset characteristics. The experiments are conducted on a machine with an Intel Core i7-10510U processor, 16 GB of memory, and running Windows 10.

Table 2 presents the parameter settings for the four SI-based algorithms used in this experiment. To ensure a fair comparison, the population size is fixed at 50, and the number of iterations is set to 200 for all algorithms.

The MOPAR algorithm adopts an external repository size of 50, and the remaining parameters are taken from Beiranvand et al. [7]. As for ACO-R, the parameter values are determined through testing, as the original authors, Qodmanan et al. [8], did not specify the optimal parameters. The MOB-ARM algorithm employs 40 iterations and aims to obtain 5 Pareto points. Each algorithm is tested five times on each dataset. The programming code for these algorithms is available in the GitHub repository.[1]

Table 3 displays the average support values obtained by the four SI algorithms and the Apriori algorithm across the four datasets. Fig. 1 visually represents the average support values of the mined rules generated by these algorithms. Among the algorithms, MOCANAR consistently yielded rules with high support across most of the datasets. ACO-R also produced rules with high support in the Basketball, Quake, and Longley datasets, but it performed relatively poorer in the Fat dataset. MOB-ARM achieved average support values across all datasets. The Apriori algorithm, on the other hand, achieved a lower average support value compared to the other algorithms. Meanwhile, MOPAR exhibited the lowest support values overall, indicating that it discovered relatively fewer itemsets with high occurrence rates.

Table 4 provides the average confidence values derived from the SI and the Apriori algorithms for the four datasets. Fig. 2 visually presents the average confidence values of the rules generated by these algorithms. MOCANAR and ACO-R consistently achieved the highest confidence results across all datasets. On the other hand, MOPAR and MOB-ARM yielded average confidence values, but they exhibited the lowest confidence in the Fat and Basketball datasets, respectively, when compared to the other algorithms. However, the Apriori algorithm has achieved a commendable average confidence value.

[1]https://github.com/rahul-sharmaa/Performance-Analysis-of-SI-based-Algorithms.git

**TABLE 3.** Average support of the MOPAR, MOCANAR, ACO-R, MOB-ARM and apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] | Apriori [16] |
|---|---|---|---|---|---|
| Basketball | 0.13 | 0.49 | 0.41 | 0.28 | 0.16 |
| Fat | 0.08 | 0.63 | 0.01 | 0.34 | 0.16 |
| Quake | 0.22 | 0.51 | 0.57 | 0.45 | 0.24 |
| Longley | 0.10 | 0.29 | 0.35 | 0.28 | 0.19 |

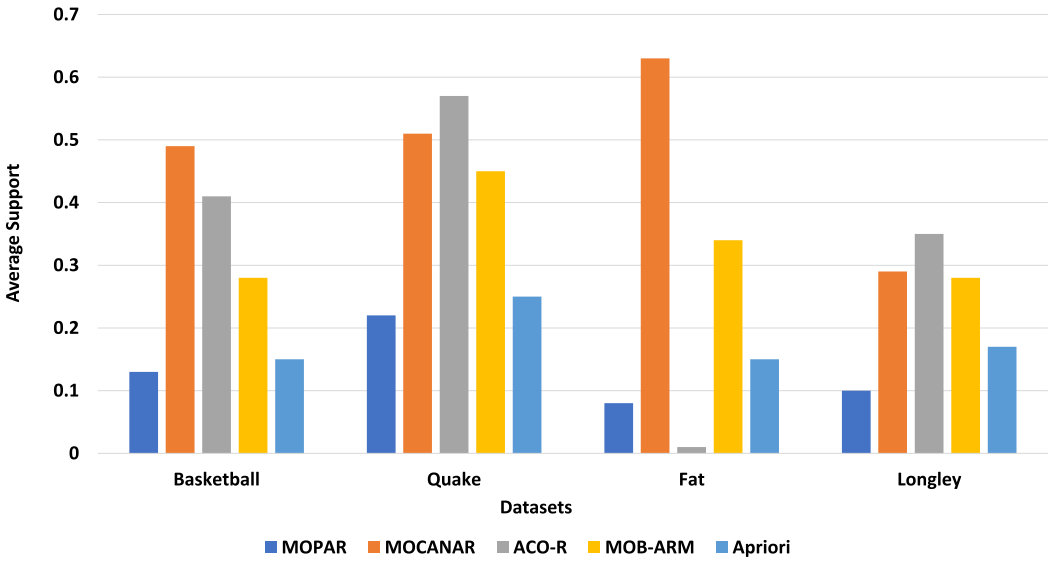


**FIGURE 1.** Average support of MOPAR, MOCANAR, ACO-R, MOB-ARM and Apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

**TABLE 4.** Average confidence of the MOPAR, MOCANAR, ACO-R, MOB-ARM and apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] | Apriori [16] |
|---|---|---|---|---|---|
| Basketball | 0.78 | 0.80 | 0.80 | 0.63 | 0.78 |
| Fat | 0.48 | 0.87 | 0.86 | 0.72 | 0.82 |
| Quake | 0.71 | 0.84 | 0.87 | 0.72 | 0.82 |
| Longley | 0.94 | 0.93 | 0.99 | 0.92 | 0.90 |

**TABLE 5.** Average generated rules by the MOPAR, MOCANAR, ACO-R, MOB-ARM and apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] | Apriori [16] |
|---|---|---|---|---|---|
| Basketball | 11.2 | 32.8 | 40.4 | 8.4 | 4 |
| Fat | 10.4 | 54.6 | 13.8 | 7.8 | 237.40 |
| Quake | 18.6 | 22.2 | 47 | 8.4 | 16.2 |
| Longley | 16.2 | 8.8 | 8.4 | 20.6 | 136.8 |

Table 5 displays the average number of association rules generated by each algorithm for each dataset. It is evident from the table that MOCANAR and ACO-R mined the highest number of rules across all datasets. Conversely, MOPAR and MOB-ARM yielded the fewest rules across all datasets, with the exception of the Longley dataset. However, the Apriori algorithm produced numerous rules for all datasets except for Basketball. Fig. 3 provides a visual representation of the average number of rules mined by the algorithms.

Table 6 provides the average time taken by the algorithms, Table 7 presents the average comprehensibility values of the mined rules, and Table 8 displays the average interestingness values of the mined rules. Additionally, Table 9 offers a comparative analysis of all five algorithms across the four datasets.

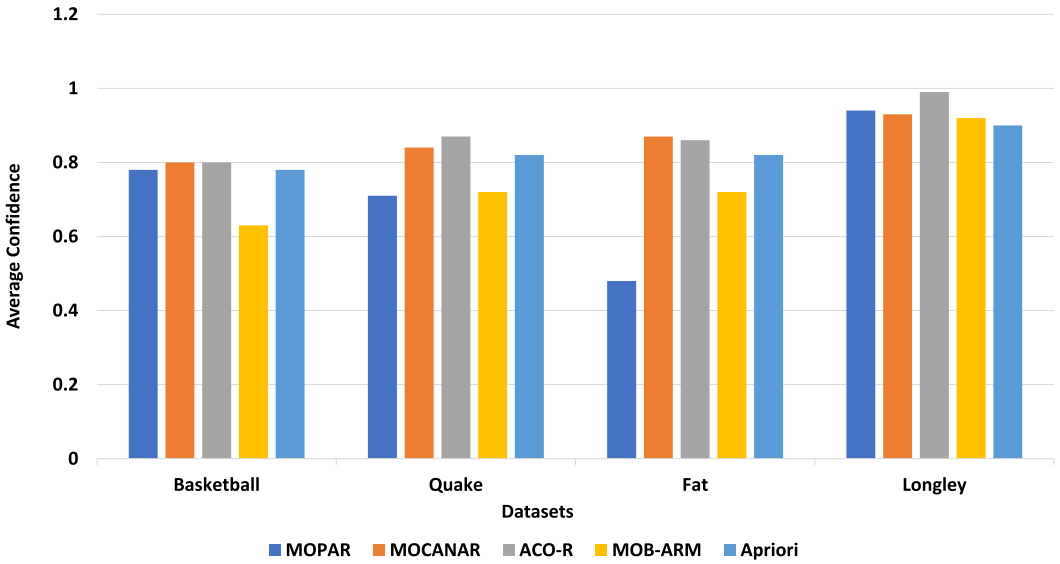


**FIGURE 2.** Average confidence of the MOPAR, MOCANAR, ACO-R, MOB-ARM and Apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

**TABLE 6.** Average time (in seconds) spent by the MOPAR, MOCANAR, ACO-R, MOB-ARM and apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] | Apriori [16] |
|---|---|---|---|---|---|
| Basketball | 455.4 | 404.9 | 442 | 1181.92 | 750.78 |
| Fat | 1259.28 | 1469.22 | 1173.26 | 3345.4 | 1891 |
| Quake | 361 | 424.04 | 402.16 | 1253.42 | 1463 |
| Longley | 500.28 | 545.08 | 604.52 | 1539.3 | 1614 |

**TABLE 7.** Average comprehensibility of the MOPAR, MOCANAR, ACO-R and MOB-ARM algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] |
|---|---|---|---|---|
| Basketball | 0.82 | 0.67 | 0.62 | 0.62 |
| Fat | 0.83 | 0.69 | 0.75 | 0.62 |
| Quake | 0.71 | 0.66 | 0.63 | 0.64 |
| Longley | 0.90 | 0.75 | 0.55 | 0.70 |

**TABLE 8.** Average interestingness of the MOPAR, MOCANAR, ACO-R and MOB-ARM algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | MOPAR [7] | MOCANAR [10] | ACO-R [8] | MOB-ARM [9] |
|---|---|---|---|---|
| Basketball | 0.43 | 0.24 | 0.25 | 0.24 |
| Fat | 0.15 | 0.21 | 0.54 | 0.27 |
| Quake | 0.16 | 0.24 | 0.24 | 0.22 |
| Longley | 0.84 | 0.65 | 0.56 | 0.59 |

Fig. 4 depicts the comparative runtime performance of the algorithms. It is evident that MOPAR, MOCANAR, and ACO-R exhibit similar runtime performance across all datasets, while MOB-ARM and Apriori are consistently slower. Fig. 5 showcases the average comprehensibility values of the mined rules. MOPAR consistently yields the highest comprehensibility measures for all datasets. On the other hand, MOCANAR, ACO-R, and MOB-ARM exhibit similar average comprehensibility results across all datasets. Fig. 6 illustrates the average interestingness values of the mined rules. MOPAR yields the highest interestingness values for the Basketball and Longley datasets but the lowest values for the Quake and Fat datasets. ACO-R obtains the highest interestingness measure for the

**TABLE 9.** Comparative experimental results for the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

| Datasets | Algorithms | Time(sec) | Avg. rules | Avg. Supp. | Avg. Conf. | Avg. Comp. | Avg. Int. |
|---|---|---|---|---|---|---|---|
| Basketball | MOPAR | 455.4 | 11.2 | 0.13 | 0.78 | 0.82 | 0.43 |
| | MOCANAR | 404.9 | 32.8 | 0.49 | 0.80 | 0.67 | 0.24 |
| | ACO-R | 442 | 40.4 | 0.41 | 0.80 | 0.62 | 0.25 |
| | MOB-ARM | 1181.92 | 8.4 | 0.28 | 0.63 | 0.62 | 0.24 |
| | Apriori | 750.78 | 4 | 0.16 | 0.78 | NA | NA |
| Quake | MOPAR | 361 | 18.6 | 0.22 | 0.71 | 0.71 | 0.16 |
| | MOCANAR | 424.04 | 22.2 | 0.51 | 0.84 | 0.66 | 0.24 |
| | ACO-R | 402.16 | 47 | 0.57 | 0.87 | 0.63 | 0.24 |
| | MOB-ARM | 1253.42 | 8.4 | 0.45 | 0.72 | 0.64 | 0.22 |
| | Apriori | 1463 | 16.2 | 0.24 | 0.82 | NA | NA |
| Fat | MOPAR | 1259.28 | 10.4 | 0.08 | 0.48 | 0.83 | 0.15 |
| | MOCANAR | 1469.22 | 54.6 | 0.63 | 0.87 | 0.69 | 0.21 |
| | ACO-R | 1173.26 | 13.8 | 0.01 | 0.86 | 0.75 | 0.54 |
| | MOB-ARM | 3345.4 | 7.8 | 0.34 | 0.72 | 0.62 | 0.27 |
| | Apriori | 1891 | 237.40 | 0.16 | 0.82 | NA | NA |
| Longley | MOPAR | 500.28 | 16.2 | 0.10 | 0.94 | 0.90 | 0.84 |
| | MOCANAR | 545.08 | 8.8 | 0.29 | 0.93 | 0.75 | 0.65 |
| | ACO-R | 604.52 | 8.4 | 0.35 | 0.99 | 0.55 | 0.56 |
| | MOB-ARM | 1539.3 | 20.6 | 0.28 | 0.92 | 0.70 | 0.59 |
| | Apriori | 1614 | 136.8 | 0.19 | 0.90 | NA | NA |

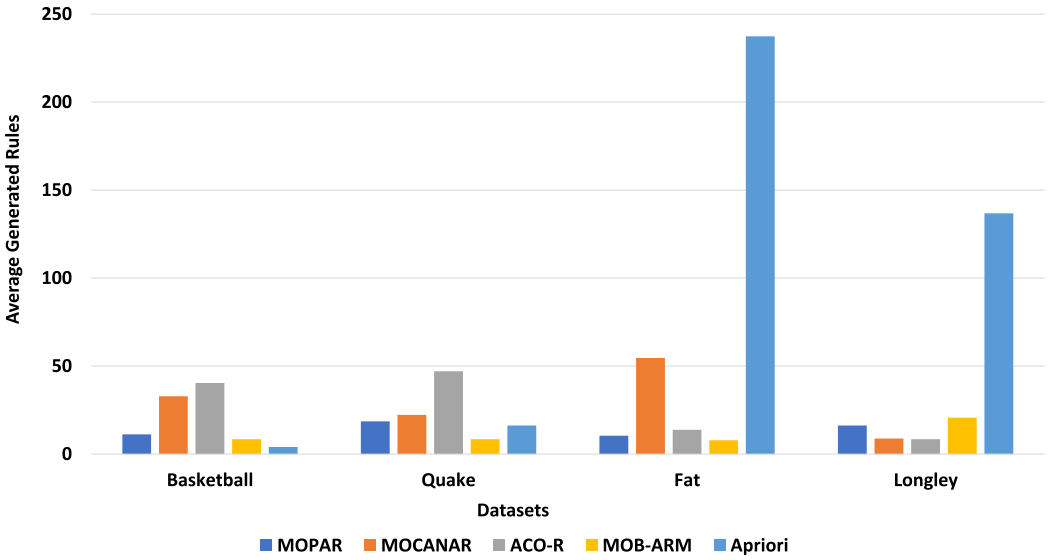


**FIGURE 3.** Average generated rules by the MOPAR, MOCANAR, ACO-R, MOB-ARM and Apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

**TABLE 10.** The average values of six measures across all datasets.

| Algorithms | Time(sec) | Avg. rules | Avg. Supp. | Avg. Conf. | Avg. Comp. | Avg. Int. |
|---|---|---|---|---|---|---|
| MOPAR | 644 | 14 | 0.13 | 0.72 | 0.81 | 0.39 |
| MOCANAR | 710 | 29 | 0.48 | 0.86 | 0.69 | 0.33 |
| ACO-R | 655 | 27 | 0.33 | 0.88 | 0.64 | 0.40 |
| MOB-ARM | 1830 | 11 | 0.34 | 0.75 | 0.64 | 0.33 |
| Apriori | 1429 | 98.6 | 0.18 | 0.83 | NA | NA |

Fat dataset. MOCANAR and MOB-ARM produce average interestingness results across all datasets.

Fig. 7 and Fig. 8 display the boxplots for confidence, support, interestingness, and comprehensibility measures for the Longley and Quake datasets, respectively. These datasets were selected based on their record count, with Longley having the lowest number of records and Quake having the highest number of records. For the Longley dataset, MOPAR demonstrates the best results in terms of comprehensibility and interestingness measures, as evident from the boxplots.

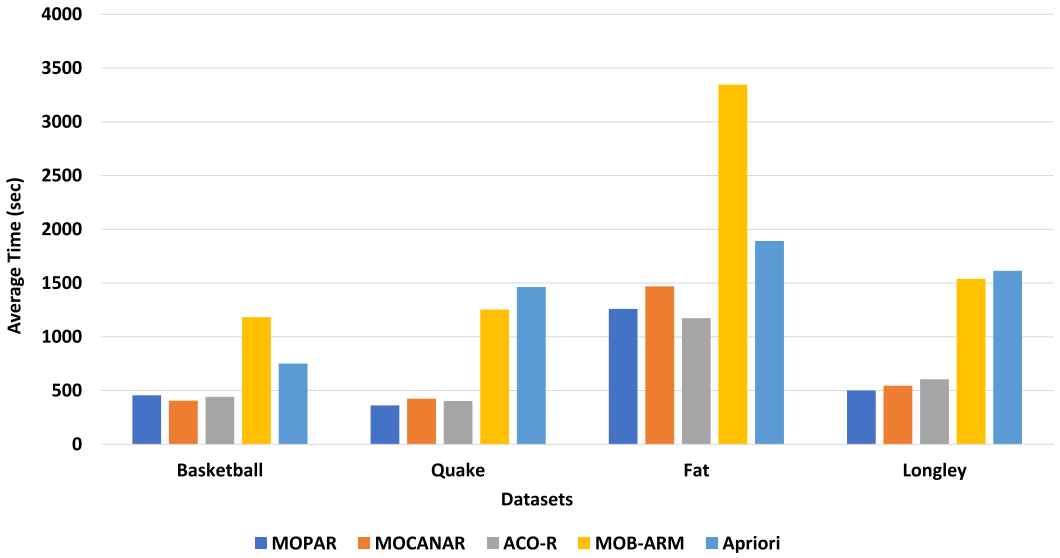


**FIGURE 4.** Average time (in seconds) spent by the MOPAR, MOCANAR, ACO-R, MOB-ARM and apriori algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

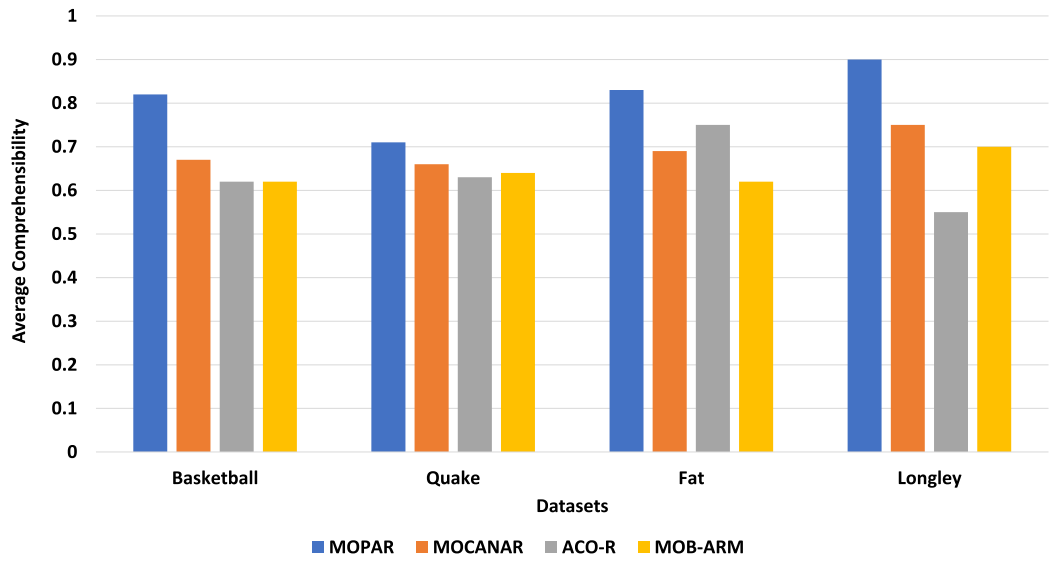


**FIGURE 5.** Average comprehensibility of the MOPAR, MOCANAR, ACO-R and MOB-ARM algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

However, the algorithms perform similarly in terms of confidence, with ACO-R yielding the best result.

It is worth noting that the algorithms do not perform well in terms of support, as indicated by the boxplots. Although ACO-R achieves a higher average support value compared to other algorithms, overall support values are relatively low for all algorithms.

In the case of the Quake dataset, when comparing the algorithms for the interestingness measure, MOCANAR, ACO-R, and MOB-ARM demonstrate similar values, as shown in the boxplots. However, MOPAR yields significantly different results for interestingness, achieving better performance in terms of comprehensibility. ACO-R stands out by providing the best results in terms of confidence and support values, as observed from the boxplots.

Table 10 provides the average results of the four SI-based NARM algorithms and the Apriori algorithm across the four datasets, considering six different evaluation measures. Upon analyzing the table, it is evident that none of the algorithms performed the best across all six measures.

MOPAR demonstrated superior performance in terms of average time, average comprehensibility, and average interestingness. On the other hand, ACO-R achieved the best results in terms of average confidence and average interestingness measures. MOCANAR exhibited the highest average support value. The Apriori algorithm generated the

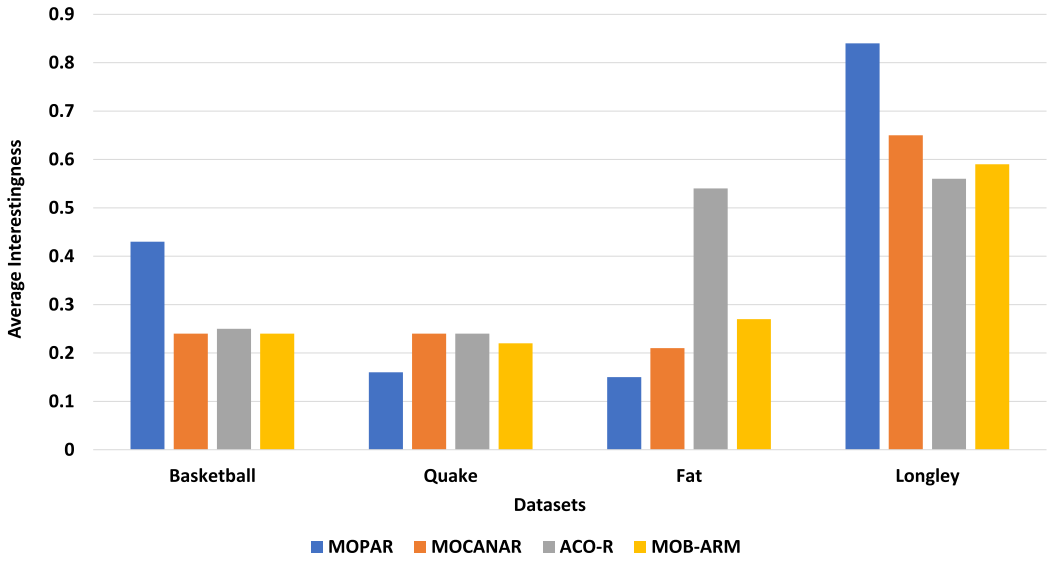


**FIGURE 6.** Average interestingness of the MOPAR, MOCANAR, ACO-R and MOB-ARM algorithms with respect to the 'Basketball', 'Quake', 'Fat' and 'Longley' datasets.

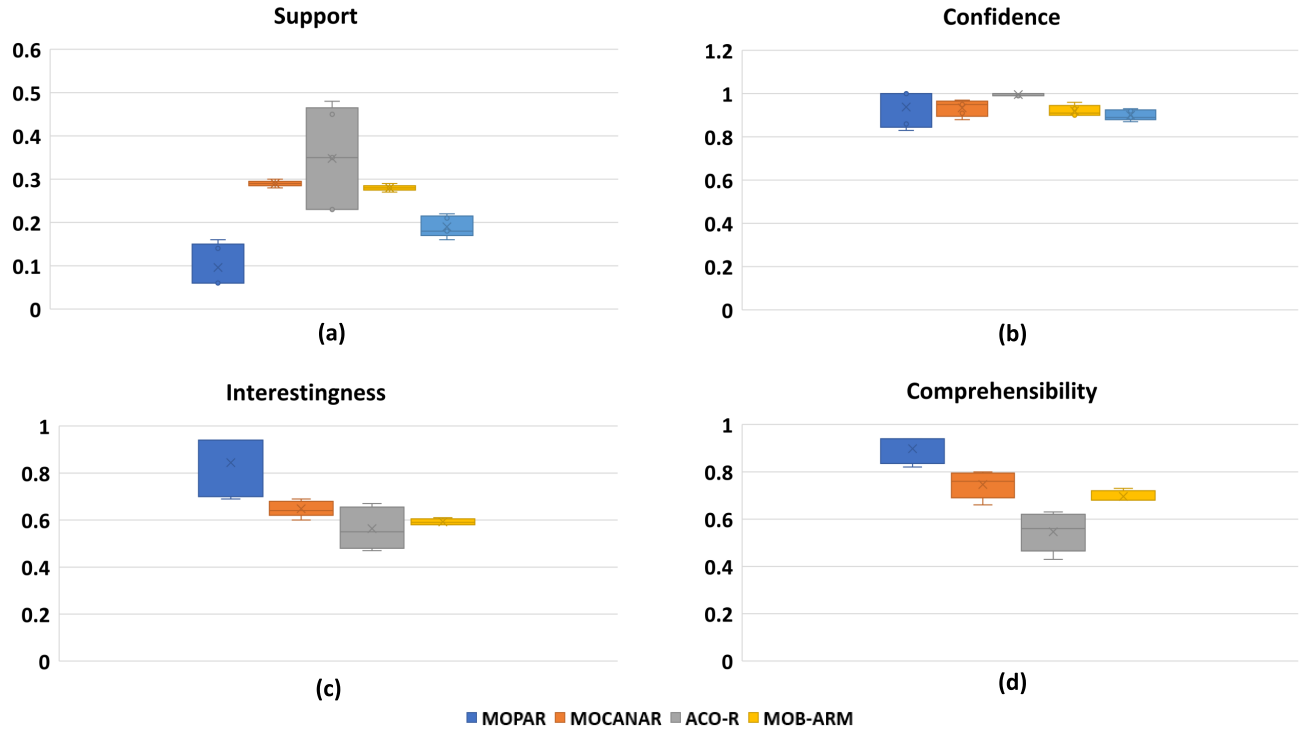


**FIGURE 7.** Boxplots illustrating the Support, Confidence, Comprehensibility, and Interestingness values for the 'Longley' dataset.

most average rules, but there is a chance that some of them may be redundant or not very interesting. Overall, each algorithm has its strengths and weaknesses, and the choice of the algorithm would depend on the specific requirements and priorities of the task at hand.

## VI. FUTURE DIRECTIONS

SI-based algorithms have indeed been employed in NARM to enhance the performance of traditional data mining algorithms. However, there are certain challenges and issues that need to be addressed in order to make SI-based algorithms more effective.

One crucial factor to consider for a fair comparison among different algorithms is the selection of appropriate stopping criteria. Ravber et al. [49] have highlighted this issue and concluded that using the maximum number of generations as a stopping criterion can be detrimental and is not recommended for ensuring a fair comparison of

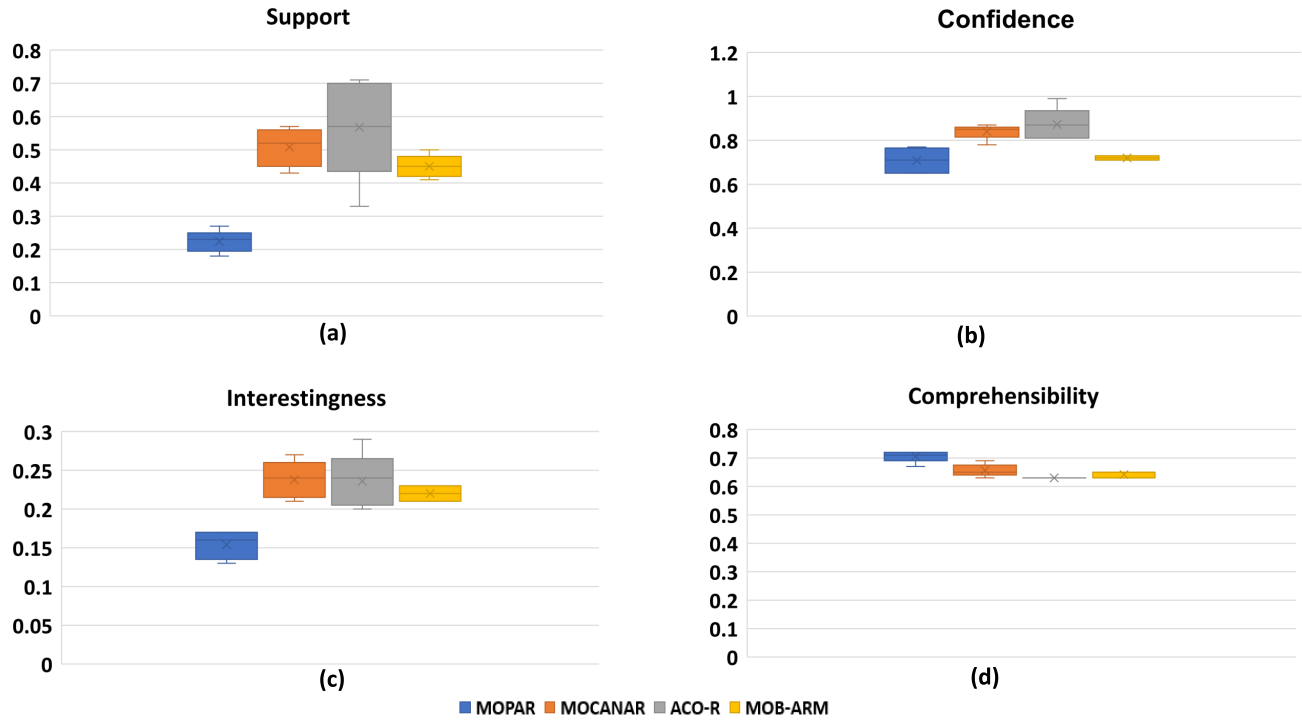


**FIGURE 8.** Boxplots illustrating the support, confidence, comprehensibility and interestingness values for the 'Quake' dataset.

optimization algorithms. However, in this paper, we have followed the original settings proposed for the algorithms, which include using the maximum generation as the stopping criterion. For future research directions, it is important to give due consideration to the selection of stopping criteria to ensure fair and meaningful comparisons among different algorithms. Alternative stopping criteria that take into account convergence behavior or other relevant factors should be explored and employed to evaluate the performance of SI-based algorithms effectively.

Next, scalability and premature convergence are critical issues that need to be addressed in optimization algorithms, including SI-based algorithms in NARM. These challenges can hinder the practical applicability and effectiveness of these algorithms, particularly when dealing with larger datasets.

Scalability: As datasets grow in size, the computational and memory requirements of optimization algorithms increase exponentially. This scalability issue poses a challenge for SI-based algorithms, as they may struggle to handle large datasets efficiently. To address this, future research should focus on developing scalable algorithms that can effectively handle big data. This may involve techniques such as parallel computing, distributed processing, or sampling methods to reduce the computational burden and memory requirements.

Premature Convergence: Premature convergence occurs when an algorithm converges to a sub-optimal solution prematurely without exploring the full search space. This issue can arise due to inappropriate parameter settings or when the search space is too small, limiting the algorithm's exploration capabilities. To mitigate premature convergence, researchers should investigate techniques such as adaptive parameter tuning, dynamic population sizing, or diversity maintenance mechanisms. These approaches can help prevent premature convergence by promoting exploration and preventing the algorithm from getting trapped in local optima.

By addressing the scalability and premature convergence challenges, SI-based algorithms in NARM can become more applicable and effective for real-world applications, especially in scenarios involving large datasets.

Further, parameter tuning is a crucial challenge in SI-based algorithms and can significantly impact their performance. Finding the optimal set of parameter values for an algorithm can be time-consuming and requires domain expertise. It often involves conducting multiple experiments and performing a thorough analysis to determine the best parameter configuration. Moreover, while SI-based algorithms are generally robust to noise and incomplete data, they can still be sensitive to certain types of noise or outliers. Noise and outliers can disrupt the optimization process and lead to suboptimal or misleading results. Therefore, it is important to develop strategies to handle noise and outliers effectively, such as preprocessing techniques, outlier detection, or robust optimization methods.

In the future, addressing these issues and challenges is crucial for the future development and advancement of SI-based algorithms in the field of NARM. By tackling these challenges, researchers can enhance the effectiveness,

efficiency, and applicability of SI-based algorithms, making them more reliable and practical for solving real-world problems.

## VII. CONCLUSION

This paper presented a comprehensive analysis of five algorithms, four of which are SI-based algorithms for NARM and the fifth being the traditional Apriori algorithm. The analysis is conducted using four real-world datasets and evaluated based on six key parameters: time efficiency, average support, average confidence, the average number of rules generated, average comprehensibility, and average interestingness. The experimental results unveiled interesting findings, with the MOB-ARM algorithm showcasing the highest average processing time across all datasets. On the other hand, the MOPAR, MOCANAR, and ACO-R algorithms exhibited superior performance compared to the Apriori algorithm. The analysis of average comprehensibility highlighted MOPAR's consistent superiority over the other algorithms across all datasets. Despite generating a lower number of rules, MOPAR's rules exhibited impressive confidence and interestingness measures. However, for datasets with a larger number of attributes or instances, some parameter adjustments may be necessary. On the other hand, the MOCANAR algorithm consistently generated rules with reliable outcomes across all metrics and datasets. Meanwhile, ACO-R produced high-quality rules overall, but it demonstrated a relative underperformance in terms of support for the Fat dataset. Hence, ACO-R might benefit from parameter adjustments when dealing with datasets featuring more attributes. As for MOB-ARM, it generated a moderate number of rules with average results across all datasets. However, it stood out as the slowest among the algorithms analyzed. To enhance MOB-ARM's performance, one potential improvement could involve eliminating the discretization step, which would lead to more rules and reduced time complexity. This adjustment might significantly enhance the algorithm's efficiency while maintaining its rule-generation capabilities. The Apriori algorithm undoubtedly stands out by generating the highest number of average rules. However, this abundance of rules warrants careful consideration, as it may include redundant or less interesting rules. While Apriori's efficiency in rule generation cannot be denied, the quality and relevance of these rules must be thoroughly assessed to ensure their usefulness in NARM tasks. Despite demonstrating good performance in terms of average confidence, the Apriori algorithm underperformed in terms of average support and average time. In contrast, the four SI-based algorithms for NARM showcased superior performance in various aspects. Based on this analysis, the study concludes that no single SI-based algorithm is universally optimal for efficient NARM. Instead, a more effective approach involves employing a combination of algorithms, each selected based on specific metrics and objectives. By leveraging the strengths of multiple algorithms, researchers can enhance the efficiency and effectiveness of NARM, making it better suited for tackling diverse data mining challenges.

## REFERENCES


[1] E. Varol Altay and B. Alatas, "Intelligent optimization algorithms for the problem of mining numerical association rules," *Phys. A, Stat. Mech. Appl.*, vol. 540, Feb. 2020, Art. no. 123142.

[2] M. Kaushik, R. Sharma, S. A. Peious, M. Shahin, S. B. Yahia, and D. Draheim, "A systematic assessment of numerical association rule mining methods," *Social Netw. Comput. Sci.*, vol. 2, no. 5, pp. 1–13, Sep. 2021.

[3] M. Kaushik, R. Sharma, S. A. Peious, M. Shahin, S. B. Yahia, and D. Draheim, "On the potential of numerical association rule mining," in *Proc. Int. Conf. Future Data Secur. Eng.* Cham, Switzerland: Springer, 2020, pp. 3–20.

[4] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence*. Cham, Switzerland: Springer, 2008, pp. 43–85.

[5] I. Fister and I. Fister, *A Brief Overview of Swarm Intelligence-Based Algorithms for Numerical Association Rule Mining*. Singapore: Springer, 2021, ch. 3, pp. 47–59.

[6] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.

[7] V. Beiranvand, M. Mobasher-Kashani, and A. A. Bakar, "Multi-objective PSO algorithm for mining numerical association rules without a priori discretization," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4259–4273, Jul. 2014.

[8] H. R. Qodmanan, M. Nasiri, and B. Minaei-Bidgoli, "Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 288–298, Jan. 2011.

[9] K. E. Heraguemi, N. Kamel, and H. Drias, "Multi-objective bat algorithm for mining numerical association rules," *Int. J. Bio-Inspired Comput.*, vol. 11, no. 4, p. 239, 2018.

[10] I. Kahvazadeh and M. S. Abadeh, "MOCANAR: A multi-objective Cuckoo search algorithm for numeric association rule discovery," *Comput. Sci. Inf. Technol.*, vol. 99, pp. 99–113, Nov. 2015.

[11] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 1996, pp. 1–12.

[12] P. Kõiva. (Jun. 2022). *Implementation and Performance Assessment of Swarm Intelligence Based Numerical Association Rule Mining Algorithms*. [Online]. Available: https://digikogu.taltech.ee/en/Item/1bbeda96-9036-43be-8618-e83d811f3080

[13] R. Sharma, M. Kaushik, S. A. Peious, A. Bazin, S. A. Shah, I. Fister, S. B. Yahia, and D. Draheim, "A novel framework for unification of association rule mining, online analytical processing and statistical reasoning," *IEEE Access*, vol. 10, pp. 12792–12813, 2022.

[14] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 6, pp. 914–925, Dec. 1993.

[15] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. VLDB 20th Int. Conf. Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann, 1994, pp. 487–499.

[17] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining Knowl. Discovery*, vol. 8, no. 1, pp. 53–87, Jan. 2004.

[18] K. Hu, L. Qiu, S. Zhang, Z. Wang, and N. Fang, "An animal dynamic migration optimization method for directional association rule mining," *Expert Syst. Appl.*, vol. 211, Jan. 2023, Art. no. 118617. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422016669

[19] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm Evol. Comput.*, vol. 48, pp. 220–250, Aug. 2019.

[20] R. E. J. Kennedy, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, Aug. 1995, pp. 1942–1948.

[21] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[22] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 210–214.
[23] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*. Berlin, Germany: Springer, 2010, pp. 65–74.
[24] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016.
[25] R. Tang, S. Fong, X.-S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," in *Proc. 7th Int. Conf. Digit. Inf. Manage. (ICDIM)*, Aug. 2012, pp. 165–172.
[26] B. Alatas and E. Akin, "Rough particle swarm optimization and its applications in data mining," *Soft Comput.*, vol. 12, no. 12, pp. 1205–1218, Oct. 2008.
[27] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.
[28] M. Ledmi, H. Moumen, A. Siam, H. Haouassi, and N. Azizi, "A discrete crow search algorithm for mining quantitative association rules," *Int. J. Swarm Intell. Res.*, vol. 12, no. 4, pp. 101–124, Oct. 2021.
[29] R. J. Kuo, M. Gosumolo, and F. E. Zulvia, "Multi-objective particle swarm optimization algorithm using adaptive archive grid for numerical association rule mining," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3559–3572, Aug. 2019.
[30] Ž. Stupan and I. Fister, "NiaARM: A minimalistic framework for numerical association rule mining," *J. Open Source Softw.*, vol. 7, no. 77, p. 4448, Sep. 2022.
[31] I. Fister, A. Iglesias, A. Galvez, J. Del Ser, E. Osaba, and I. Fister, "Differential evolution for association rule mining using categorical and numerical attributes," in *Intelligent Data Engineering and Automated Learning—-IDEAL 2018*, H. Yin, D. Camacho, P. Novais, and A. J. Tallón-Ballesteros, Eds. Cham, Switzerland: Springer, 2018, pp. 79–88.
[32] I. Fister, A. Iglesias, A. Galvez, and I. Fister Jr., "Online numerical association rule miner," *Neurocomputing*, vol. 523, pp. 33–43, Feb. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222014990
[33] I. Fister and I. Fister, "uARMSolver: A framework for association rule mining," 2020, *arXiv:2010.10884*.
[34] E. Varol Altay and B. Alatas, "Performance analysis of multi-objective artificial intelligence optimization algorithms in numerical association rule mining," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 8, pp. 3449–3469, Aug. 2020.
[35] E. V. Altay and B. Alatas, "Association analysis of Parkinson disease with vocal change characteristics using multi-objective metaheuristic optimization," *Med. Hypotheses*, vol. 141, Aug. 2020, Art. no. 109722.
[36] E. V. Altay and B. Alatas, "A novel clinical decision support system for liver fibrosis using evolutionary multi-objective method based numerical association analysis," *Med. Hypotheses*, vol. 144, Nov. 2020, Art. no. 110028.
[37] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms," *Swarm Evol. Comput.*, vol. 77, Mar. 2023, Art. no. 101248.
[38] R. Sharma, M. Kaushik, S. A. Peious, S. B. Yahia, and D. Draheim, "Expected vs. unexpected: Selecting right measures of interestingness," in *Big Data Analytics and Knowledge Discovery*, vol. 12393. Cham, Switzerland: Springer, 2020, pp. 38–47.
[39] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Comput. Surveys*, vol. 38, no. 3, p. 9, Sep. 2006.
[40] A. Ghosh, "Multi-objective rule mining using genetic algorithms," *Inf. Sci.*, vol. 163, nos. 1–3, pp. 123–133, Jun. 2004.
[41] K. Deb, *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*. London, U.K.: Springer, 2011, ch. 3, pp. 3–34.
[42] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 4–19, Feb. 2014.
[43] V. Palakonda and R. Mallipeddi, "Pareto dominance-based algorithms with ranking methods for many-objective optimization," *IEEE Access*, vol. 5, pp. 11043–11053, 2017.
[44] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," 2013, *arXiv:1307.4186*.
[45] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. London, U.K.: Oxford Univ. Press, Oct. 1999.
[46] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.
[47] K. E. Heraguemi, N. Kamel, and H. Drias, "Association rule mining based on bat algorithm," *J. Comput. Theor. Nanosci.*, vol. 12, no. 7, pp. 1195–1200, Jul. 2015.
[48] H. A. Guvenir, I. Uysal, and F. A. Repositor, *Function Approximation Repository*. Ankara, Türkiye: Bilkent Univ., 2000. [Online]. Available: http://funapp.cs.bilkent.edu.tr/DataSets
[49] M. Ravber, S.-H. Liu, M. Mernik, and M. Črepinšek, "Maximum number of generations as a stopping criterion considered harmful," *Appl. Soft Comput.*, vol. 128, Oct. 2022, Art. no. 109478. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494622005804

**MINAKSHI KAUSHIK** received the B.Tech. and M.Tech. degrees in computer science engineering from Uttar Pradesh Technical University, India, and the Ph.D. degree in computer science engineering from the Information Systems Group, Tallinn University of Technology, Estonia. Her research interests include association rule mining, data science, machine learning, and deep learning.

**RAHUL SHARMA** received the Ph.D. degree from Tallinn University of Technology, Estonia. He is currently the Head of the Information Technology Department, Ajay Kumar Garg Engineering College, Ghaziabad. He also contributes as a Researcher with Tallinn University of Technology. He is a Distinguished Scholar. He is renowned for his passionate dedication to research and his dynamic leadership in the field.

**PILLERIIN KÕIVA,** photograph and biography not available at the time of publication.

**IZTOK FISTER JR.** received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Maribor, Slovenia. He is currently an Assistant Professor with the University of Maribor. He has published more than 120 research papers in refereed journals, conferences, and book chapters. His research interests include data mining, pervasive computing, optimization, and sport science. He is a member of the editorial boards of five different international journals. He has acted as a program committee member of more than 30 international conferences.

**DIRK DRAHEIM** received the Ph.D. degree from Freie Universität Berlin and the Habilitation degree from Universität Mannheim, Germany. He is currently a Full Professor of information systems and the Head of the Information Systems Group, Tallinn University of Technology, Estonia. The Information Systems Group conducts research in large and ultra-large-scale IT systems. He is also an Initiator and a Leader of numerous digital transformation initiatives.