Received 24 November 2023, accepted 27 December 2023, date of publication 29 December 2023, date of current version 10 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3348408

RESEARCH ARTICLE

Variable-Length Differential Evolution for Numerical and Discrete Association Rule Mining

UROŠ MLAKAR, (Member, IEEE), IZTOK FISTER JR., (Member, IEEE), AND IZTOK FISTER, (Member, IEEE)
Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

Corresponding author: Uroš Mlakar (uros.mlakar@um.si)

This work was supported by the Slovenian Research and Innovation Agency under Grant P2-0041.

ABSTRACT This paper proposes a variable-length Differential Evolution for Association Rule Mining. The proposed algorithm includes a novel representation of individuals, which can encode both numerical and discrete attributes in their original or absolute complement of the original intervals. The fitness function used is comprised of a weighted sum of Support and Confidence Association Rule Mining metrics. The proposed algorithm was tested on fourteen publicly available, and commonly used datasets from the UC Irvine Machine Learning Repository. It is also compared to the nature inspired algorithms taken from the NiaARM framework, providing superior results. The implementation of the proposed algorithm follows the principles of Green Artificial Intelligence, where a smaller computational load is required for obtaining promising results, and thus lowering the carbon footprint.

INDEX TERMS Association rule mining, differential evolution, data mining, variable-length solution representation, green AI.

I. INTRODUCTION

In today's world, the amount of data stored in real-world databases is growing rapidly and shows no signs of slowing down [1]. These databases encompass a variety of fields, including medical, scientific, financial, and marketing transaction data, with many containing vast amounts of information [2]. As a result, analyzing these datasets and uncovering valuable insights has become a major challenge. Over the past decade, the most effective approach to addressing this issue has been through Data Mining methods focused on knowledge discovering in large data sets. These methods are divided into descriptive and predictive. The former are devoted for searching the properties of data, while the latter are capable of making predictions. In general, the Data Mining is comprised of methods for association rule mining, classification, regression, clustering, deep learning, and outlier analysis [3].

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

Association Rule Mining (ARM) is a widely-used method for discovering relationships between attributes in large databases. The method aims to identify rules that exhibit a high level of interestingness based on various measures. The idea of discovering regularities between products in large-scale transaction databases was first introduced by Agrawal et al. [4], who is widely considered the pioneer of teh ARM. Agrawal proposed the Apriori algorithm, which has become a standard approach in this field since it was born in 1994. Typically, this algorithm selects association rules based on measures of interestingness, such as support and confidence, in order to identify the most significant attribute relationships in transaction databases.

With the increasing size and complexity of datasets, traditional algorithms often encounter challenges in terms of high computational complexity when mining for and generating association rules [5]. To address this issue, several new stochastic population-based nature-inspired algorithms have been proposed, that approach the rule mining process as an optimization problem. Additionally, much attention

© 2023 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

has been devoted to the types of attributes being mined (i.e. categorical and continuous).

The problem of the high computational complexity has firstly been exposed by the usage of AI solutions in practise, where, especially, large and computationally expensive deep learning models produce a huge carbon footprint by increasing the energy consumption [6]. The traditional AI methods have recently been referred to as Red AI. Indeed, it increases the carbon footprint, and is environmentally unfriendly and energy consumptive. The modern sustainable AI, that is referred to as Green AI, is distinguished by reducing the use of hazardous materials, maximizing the energy efficiency, and increasing the factory waste recycling [7].

In the context of stochastic population-based nature-inspired algorithms, Evolutionary Algorithms (EAs), like Genetic Algorithms (GAs), and Swarm Intelligence (SI) based algorithms, like Particle Swarm Optimization (PSO), which were often utilized for rule mining tasks, have been adapted to mine numerical association rules efficiently [8], [9]. These algorithms encode potential solutions as chromosomes, allowing variation operators such as crossover and mutation to refine the rule population iteratively. By evaluating fitness functions based on support and confidence measures, these approaches can identify significant numeric associations effectively, while accommodating numeric attributes' inherent diversities. According to the review papers in this field, GA and PSO algorithms are the most promising nature-inspired algorithms for solving these problems [10], RPSOA [11]. However, approaches are also based on the Ant Colony Optimization (ACO) [12], Bat algorithm (BA) [13], Cuckoo Search (CS) [14] and Differential Evolution (DE) [15], among others. Even though some algorithms work well on specific datasets, while others work on different datasets, much attention is nowadays given to developing new metrics that help obtain the most appropriate association rules [5], [9].

Most of the reviewed literature focuses mainly on either mining discrete or numerical rules. Also, most of the time, the numerical attributes are discretized in the datasets used, which leads to severe information loss and misinterpretation of the produced rules. It also seems that all methods mine just positive intervals of attributes' domains, thus ignoring other views of rules on the underlying hidden data. Additionally, all the reviewed methods use a fixed length individual representation, which means that each attribute can hold just one specific value. This way of encoding means that different intervals of the same attribute cannot be discovered. This leads to not being able to find very strict and specific rules.

There are a handful methods in the literature which use variable-length individual representations, among which only the following are used in Data Mining. In [16] the authors used a Genetic Algorithm (GA) for mining AR in text documents. Their method produces rules of different lengths by sampling selected individuals randomly for crossover. The method in [17] also utilized a GA for association rule mining as a means of feature selection for spam detection. Their idea for producing different length individuals lies in three carefully adapted single-point crossover operators which ensure feasible offspring. Another hybrid GA was utilized in [18], where variable-length individuals were used for optimizing fuzzy rules, with the aim of facilitating a comprehensive quality assurance scheme in the garment industry. During the rule optimization, different parameters can be inserted or removed from a rule, increasing the diversity of the generated fuzzy rules. A multi-objective GA was used is [19] for fuzzy classification rule mining. By representing fuzzy rules as concatenated integer strings of variable length, the method maximizes accuracy and minimizes complexity, enhancing the efficiency in rule set generation significantly without compromising the results.

All the above reviewed methods operate on only discrete attributes or utilize some form of discretization. They generate new variable-length individuals using a variation of a single-point crossover, which, while easy to implement, necessitates changes made through the mutation operation.

With the drawbacks of the described state-of-the-art methods in mind, this paper proposes a new Differential Evolution (DE) algorithm [20] for Association Rule Mining, based on a variable-length encoding of individuals, where each individual encodes a separate association rule. The variable-length encoding ensures that different intervals of the same attribute can be selected, therefore, having the ability to find very specific and interesting association rules. The proposed methods is also able to mine positive and negative intervals of specific attributes' domains, which enhances the quality of the results.

The proposed paper introduces the following key novelties:

- the new variable-length representation of individuals for ARM that is Green AI ready,
- the development of the new variable-length Differential Evolution algorithm for ARM (vDE_ARM),
- mining discrete and continuous rules without any form of data preprocessing,
- considering either the mined intervals of continuous attributes' domains or their absolute complements.

Let us mention that the purpose of the study is also to develop an ML method with foundations of Green AI in the sense that the vDE_ARM finds the solutions in a much shorter time, while their qualities are comparable.

The remainder of the paper is organized as follows: Section II introduces the basic information needed for understanding subjects that follows. The design and implementation of the proposed vDE_ARM is presented in Section III. In Section IV, the experiments are illustrated, where the vDE_ARM is applied for mining several transaction databases, while the results encourage us to continue with the development of this algorithm. Section VI summarizes the results of the performed work and outlines potential directions for the future development.

II. MATERIALS AND METHODS

This Section captures the topics needed by potential readers for the understanding subjects that follow. At first, a mathematical formulation of the ARM is presented briefly. The section is concluded with a detailed description of the DE, on which the mining of the Association Rules (AR) is based in the present study.

A. ASSOCIATION RULE MINING

The ARM problem is defined mathematically as follows: Let us suppose a set of attributes $A = \{a_1, \ldots, a_M\}$ and a set of transactions $T = \{t_1, \ldots, t_N\}$ are given, where each transaction is identified uniquely and contains a subset of items $t_i \subset A$ (also itemset). Let us notice that M denotes the number of attributes and N the number of transactions in the transaction database. Then, an association rule is defined as an implication:

$$X \implies Y, \tag{1}$$

where X and Y are two itemsets and it holds that $X \cap Y = \emptyset$.

Several interestingness measures have been defined for identifying and evaluating the more important association rules in the literature. However, the most commonly used are Support and Confidence that are defined as follows:

$$Support(X \implies Y) = \frac{|t_i | t_i \in X \wedge t_i \in Y|}{N}, \tag{2}$$

$$Confidence(X \implies Y) = \frac{Support(X \cup Y)}{Support(X)}, \tag{3}$$

where $Support(X \implies Y) \geq S_{min}$ denotes the support, and $Confidence(X \implies Y) \geq C_{min}$ the confidence of the association rule $X \implies Y$. Additionally, S_{min} denotes minimum support and C_{min} minimum confidence, determining that only those association rules with support and confidence higher than S_{min} and C_{min} are taken into consideration, respectively.

In some literature [21], the following additional metrics are usually used:

$$Coverage(X \implies Y) = \frac{|t_i | t_i \in Y|}{M}, \tag{4}$$

$$Interestingness(X \implies Y) = Confidence(X \implies Y) * \frac{Support(X \implies Y)}{Support(Y)} * (1 - \frac{Support(X \implies Y)}{N}). \tag{5}$$

Let us mention that the coverage and the interestingness ARM metrics are also used in this research.

The mentioned measures estimate the mined association rules differently, i.e., the support is defined as the proportion of transactions in a transaction database containing the item set X. On the other hand, the confidence is a conditional proportion of finding the item set X under the condition that this transaction also contains Y. The coverage metric estimates the ratio between the number of the attributes used in the specific AR, and all the attributes in the transaction database M. The interestingness is used to measure how much the rule is surprising for the user. In association rule mining, the paramount objective revolves around discovering concealed insights from the data. Assessing the interestingness of a rule involves identifying not only the highly frequent rules but also those with comparably lower occurrence in the database [22]. Indeed, the item set X represents the Left Hand Side (LHS), while the item set Y the Right Hand Side (RHS) of the association rule $X \implies Y$.

B. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is a population-based Evolutionary Algorithm for continuous global optimization [20]. In its core, the DE algorithm is a simple, but very effective heuristic for solving various real-life problems. The idea of the DE algorithm is a mathematical model based on scaled vector differences. The DE algorithm evolves a population of vectors, through consecutive generations, denoted as g, where each vector represents a solution of the problem, either directly or indirectly. Normally, the DE population consists of Np real-coded vectors that can be defined mathematically as follows:

$$x_i^{(g)} = (x_{i,1}^{(g)}, \ldots, x_{i,n}^{(g)}), \quad \text{for } i = 1, \ldots, Np, \tag{6}$$

where each element $x_{i,j}^{(g)}$ lies within the interval $[x_j^{(L)}, x_j^{(U)}]$ for $j = 1, \ldots, D$. Vectors $x^{(L)}$ and $x^{(U)}$ denote the lower and upper bounds of the problem, variables while D refers to the problem dimension. In each generation all the vectors in the population go through a set of evolutionary operators, namely, mutation, crossover, and selection. By using these operators a trial vector is produced, which competes with the current vector (i.e. parent) for surviving into the next generation based on the fitness value. The mutation operator is executed for every vector in the population that yields a mutant vector, denoted as:

$$u_i^{(g)} = x_{r_1}^{(g)} + F \cdot (x_{r_2}^{(g)} - x_{r_3}^{(g)}) \tag{7}$$

Thus, indices r_1, r_2, and r_3 are selected randomly from uniform distribution in the interval $[1, Np]$, and are also mutually exclusive integers different from index i, while F is the scaling factor. Typically the scaling factor lies within the interval [0.1, 1.0].

After generating the mutant vector $u_i^{(g)}$, a crossover is performed according to the crossover rate Cr and the corresponding vector $x_i^{(g)}$ from the population as follows:

$$t_{i,j}^{(g)} = \begin{cases} u_{i,j}^{(g)}, & \text{if } rand() \leq Cr or j = j_{rand}, \\ x_{i,j}^{(g)}, & otherwise. \end{cases} \tag{8}$$

The crossover rate Cr, normally defined within the interval [0, 1], controls the probability of modifying an element of the trial vector $u_i^{(g)}$. Additionally, the j_{rand} index ensures that at least one value from the mutant vector $u_i^{(g)}$ is modified in the new trial vector $t_i^{(g)}$.

The last operation is the selection, where the newly created trial vector $t_i^{(g)}$ competes with its parent for surviving into the next generation. In the case of minimization, the selection operator can be defined mathematically as:

$$x_i^{(g+1)} = \begin{cases} t_i^{(g)}, & \text{if } f(t_i^{(g)}) \leq f(x_i^{(g)}), \\ x_i^{(g)}, & otherwise. \end{cases} \tag{9}$$

Let us mention that the selection operator in DE is typically denoted as one-to-one selection.

III. VARIABLE-LENGTH DIFFERENTIAL EVOLUTION FOR ARM

A traditional approach to Evolutionary Association Rule Mining uses a fixed length representation of individuals, where the length is proportional to the number of attributes in the used transaction database. The main challenge in the study is to design an algorithm for able to mine multiple intervals of the same attributes' domains without using very complex representations of rules. With this in mind, we propose a variable-length representation of a rule (i.e. individual in the population), where the length is not influenced directly by the size of the transaction database, but rather it is learned throughout the evolutionary process by the algorithm itself. Using this new rule representation, we expect that the discovered rules will be much more precise, more interesting (according to the commonly used metrics), and also cover the whole spectrum of the underlying information of the mined dataset.

The pseudo-code of the general EA is illustrated in Algorithm 1.

Algorithm 1 Pseudocode of a Generic Evolutionary Algorithm

```
1:  function Evolutionary_Algorithm(Np, MAXFEs)
2:      INITIALIZE_population
3:      EVALUATE_each_candidate
4:      while not TERMINATE_CONDITION_satisfied do
5:          SELECT_parents
6:          RECOMBINE_parents
7:          MUTATE_offspring
8:          EVALUATE_mating_pool
9:          SELECT_survivors
10:     end while
11:     return BEST_individual
12: end function
```

from which it can be seen that this consists of the following components [23]:

- generation of initial population (function *INITIALIZE* in line 2),
- evaluation of a solution (function *EVALUATE* in lines 2 and 8),
- parent selection (function *SELECT_parent* in line 5),
- crossover of the parents (function *RECOMBINE* in line 6),
- mutation of the offspring obtained after crossover (function *MUTATE* in line 7),
- survivor selection (function *SELECT_survivor*).

To run correctly, the EA also needs an appropriate representation of individuals representing a solution of the problem to be solved, and the termination condition determining conditions for terminating the evolutionary cycle. Let us mention that the general pseudo-code of the EAs is similar to the pseudo-code of the general SI-based algorithms, but changing of the particular particle (i.e., individuals in the EA) positions in the problem search space is typically performed using some nature-inspired equation (e.g., moving bee colonies, ant colonies, bird flocks, etc.) in place of crossover and mutation as by EAs.

Indeed, the new variable-length Differential Evolution for Association Rule Mining (vDE_ARM) is introduced in this Section. The proposed algorithm is able to extract rules from transaction databases containing both numerical and categorical attributes. The implementation of vDE_ARM consists of modifying the following three key components referring to the pseudo-code of the EAs (Algorithm 1), to which the original DE also belongs:

- the representation of individuals,
- the definition of a fitness function.
- modifications of the other components of the original DE.

The first component refers to modification of the original DE algorithm, necessary to operate with variable-length representation of individuals. Since a DE algorithm is used for rule mining, the fitness function needs to be defined in the second component, that enable us to find the more interesting association rules. Finally, the variation operators in the original DE must be redefined, in order to support the variable-length representation of individuals.

Indeed, the mentioned modifications are universal, and can be applied to any EAs or SI-based algorithms. Let us notice that the proposed variable-length encoding is an integral mechanism of the newly proposed DE algorithm. Obviously, the operators of crossover and mutation are also modified in order to consider the variable-length representation. Readers are directed to the graphical representation of the vDE_ARM in Fig. 1.

A. REPRESENTATION OF INDIVIDUALS

Each individual x_i in a population is represented as a real-valued vector of variable length, where each vector encodes a separate rule in the so-called solution space (also genotype space). The rule supports two kinds of attributes in the problem space (also phenotype space), i.e., either numeric $attr_{i,j}^{(num)}$ or categorical $attr_{i,j}^{(cat)}$. Thus, each attribute is encoded by four real-valued elements. In the genotype-phenotype mapping [23], the vector is represented, in the

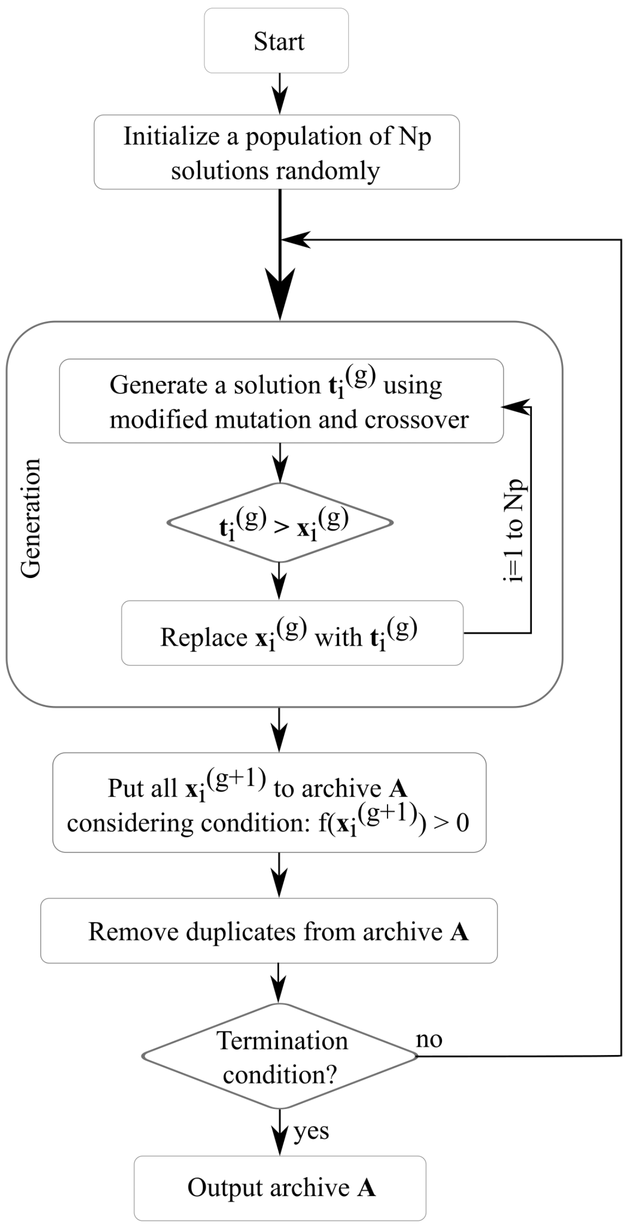

Start
Initialize a population of Np solutions randomly
Generation
Generate a solution $\mathbf{t_i}^{(g)}$ using modified mutation and crossover
$\mathbf{t_i}^{(g)} > \mathbf{x_i}^{(g)}$
i=1 to Np
Replace $\mathbf{x_i}^{(g)}$ with $\mathbf{t_i}^{(g)}$
Put all $\mathbf{x_i}^{(g+1)}$ to archive **A** considering condition: $f(\mathbf{x_i}^{(g+1)}) > 0$
Remove duplicates from archive **A**
Termination condition?
no
yes
Output archive **A**

FIGURE 1. Graphical representation of the vDE_ARM.

so-called genotype space, as follows:

$$\mathbf{x}_i^{(g)} = \Bigg(\dots, \underbrace{\left\langle x_{i,j,1}^{(g)}, x_{i,j,2}^{(g)}, x_{i,j,3}^{(g)}, x_{i,j,4}^{(g)} \right\rangle}_{attr_{i,j}^{(num)}}, \dots, \underbrace{\left\langle x_{i,j',1}^{(g)}, x_{i,j',2}^{(g)}, x_{i,j',3}^{(g)}, x_{i,j',4}^{(g)} \right\rangle}_{attr_{i,j'}^{(cat)}}, \dots, \underbrace{\left\langle x_{i,|attr|+1,1}^{(g)} \right\rangle}_{CP_i} \Bigg),$$

where the j-th element of the vector $x_{i,j}^{(g)}$ represents the mapping to the numerical, while the j'-th element $x_{i,j'}^{(g)}$ to the categorical attribute. Indeed, the vector $x_i^{(g)} = (\{x_{i,j}^{(g)}\}, x_{i,|attr|+1,1}^{(g)})$ consists of a set of 2-dimensional vectors $x_{i,j}^{(g)}$ for $j = 1, \dots, |attr|$ with elements $x_{i,j,k}^{(g)}$ for $k = 1, \dots, 4$ encoding the particular attribute of the association rule, and the last element of the vector is a cut point that determines which attributes belong to the antecedent and which to the consequent parts of the encoded association rule. The term $|attr|$ denotes the number of attributes in the observed association rule. Consequently, the length of the vector in genotype space is determined by the expression $D_i = 4 \cdot |attr| + 1$.

Obviously, the genotype-phenotype mapping is performed differently, depending on the attribute types. Obviously, both the numerical attributes $attr_{i,j}^{(num)} = \langle type, interval, compl \rangle$ and the categorical attributes $attr_{i,j}^{(cat)} = \langle type, item, reserved \rangle$ are mapped from the 2-dimensional vector $\mathbf{x}_{i,j}^{(g)}$ as illustrated in Table 1.

A type of the attribute is determined simply, according to the following expression:

$$type = \begin{cases} num, & \text{if } x_{i,j,1}^{(g)} \equiv 1, \\ cat, & \text{otherwise}, \end{cases} \tag{10}$$

where the attribute type is determined by analyzing the values of the feature in the transaction database.

Interval $[lb, ub]$ of the corresponding numerical attribute $attr_{i,j}^{(num)}.interval$ is calculated according to the following equation:

$$lb = \begin{cases} \lfloor (Ub_j - Lb_j) \cdot x_{i,j,2}^{(g)} \rfloor, & \text{if } x_{i,j,2}^{(g)} < x_{i,j,3}^{(g)}, \\ \lfloor (Ub_j - Lb_j) \cdot x_{i,j,3}^{(g)} \rfloor, & \text{otherwise}. \end{cases} \tag{11}$$

and

$$ub = \begin{cases} \lfloor (Ub_j - Lb_j) \cdot x_{i,j,3}^{(g)} \rfloor, & \text{if } x_{i,j,2}^{(g)} < x_{i,j,3}^{(g)}, \\ \lfloor (Ub_j - Lb_j) \cdot x_{i,j,2}^{(g)} \rfloor, & \text{otherwise}. \end{cases} \tag{12}$$

where the variables Lb_j and Ub_j denote the lower and upper bounds of the feasible range for the particular numerical attribute, respectively.

The absolute complement of the interval $attr_{i,j}^{(num)}.interval$ represents a set of those values that are not in the $[lb, ub]$, in other words, the absolute complement $attr_{i,j}^{(num)}.interval$ is the interval $[Lb_j, lb] \cup [ub, Ub_j]$. The decision about taking the interval or its absolute complement is made according to the following equation:

$$interval = \begin{cases} [Lb_j, lb] \cup [ub, Ub_j], & \text{if } rand(0,1) > x_{i,j,4}^{(g)}, \\ [lb, ub], & \text{otherwise}. \end{cases} \tag{13}$$

As can be seen from the Eq. (13), the complement of the original interval is taken into account, when the random number drawn from uniform distribution in the range [0, 1] is higher than the value of element $x_{i,j,4}^{(g)}$.

TABLE 1. Genotype-phenotype mapping by vDE_ARM.

Element	Map to	Numerical attribute	Categorical attribute
$\mathrm{x}_{\mathrm{i,j,1}}^{(\mathrm{g})}$	$\mapsto$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{num})}.\mathrm{type} = \mathrm{num}$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{cat})}.\mathrm{type} = \mathrm{cat}$
$\mathrm{x}_{\mathrm{i,j,2}}^{(\mathrm{g})}, \mathrm{x}_{\mathrm{i,j,3}}^{(\mathrm{g})}$	$\mapsto$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{num})}.\mathrm{interval} = [\mathrm{lb}, \mathrm{ub}]$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{cat})}.\mathrm{item} = \mathrm{map\_cat}(\mathrm{x}_{\mathrm{i,j,2}}^{(\mathrm{g})}, \mathrm{x}_{\mathrm{i,j,3}}^{(\mathrm{g})})$
$\mathrm{x}_{\mathrm{i,j,4}}^{(\mathrm{g})}$	$\mapsto$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{num})}.\mathrm{compl} = [\mathrm{true}\vert\mathrm{false}]$	$\mathrm{attr}_{\mathrm{i,j}}^{(\mathrm{cat})}.\mathrm{reserved} = 0$

Similar as for a numerical attribute, the value of the categorical variable *item* is decoded from two elements of the 2-dimensional vector $x_{i,j}^{(g)}$ using a function $map\_cat(x_{i,j,2}^{(g)}, x_{i,j,3}^{(g)})$ that is defined as follows:

$$item = \left\lfloor \frac{x_{i,j,2}^{(g)} + x_{i,j,3}^{(g)}}{2} \cdot \left(Ub_j - Lb_j\right) \right\rfloor + 1, \tag{14}$$

where the average of two elements are scaled with the total range of feasible values for the categorical attribute j. Interestingly, the fourth elements of the vector are not included into the decoding process, and, therefore, the variable *reserved* is normally set to zero.

Finally, the cutoff point cp_i, is calculated as:

$$cp_i = \lfloor x_{i,D+1} \cdot (|attr| - 1) \rfloor + 1, \tag{15}$$

where $|attr|$ designates the number of attributes in $x_i^{(g)}$. This means that all identified attributes whose indexes are smaller than cp_i belongs to the antecedent, on the one hand, and all the others to the consequent, on the other.

B. FITNESS FUNCTION EVALUATION

Since the proposed vDE_ARM belongs to the EA family of algorithms, a metric needs to be determined for measuring the quality of a rule (i.e. a fitness function). Several different fitness functions can be found in the literature, varying from very simple to complex. In this work, each mined association rule $X \implies Y$ obtained by genotype-phenotype mapping $\mathbf{x}_i^{(g)} \mapsto (X \implies Y)$ is evaluated using the function:

$$f(X \implies Y) = \frac{\alpha * Confidence(X \implies Y) + \beta * Support(X \implies Y)}{\alpha + \beta}. \tag{16}$$

Factors α and β are used to assign weights to both the confidence and support measures, which determine their relative importance in the fitness function calculation. When α is increased, the algorithm will prefer finding rules with higher confidence levels. Conversely, increasing β will favor rules with greater support. Depending on the particular use case for data rule mining, the user has the ability to adjust the weightings of both measures accordingly. In this study, the weights of support and confidence were kept the same, in other words $\alpha = \beta = 1$.

C. MODIFICATIONS OF THE OTHER COMPONENTS OF THE ORIGINAL DE ALGORITHM

The DE algorithm described in Section II-B is designed to work with optimization problems where the solution is of fixed length. Since the proposed method uses a variable-length representation of individuals, several modifications of the original DE algorithm need to be made beside introducing the variable-length representation and adapting the fitness function. These modifications are summarized in the remainder of this Section.

The first modification is made by the initialization of the initial population. Each individual in the initial population is constructed according to the following mathematical expression:

$$x_i^{(0)} = (x_{i,0}^{(0)}, \ldots, x_{i,D_i}^{(0)}), \tag{17}$$

where the length of each individual $D_i = \mathcal{N}(M, 1)$ is drawn from the normal distribution with mean M and Standard Deviation one. Normally, the variable M denotes the number of features in the mined transaction database. In this way, it is possible that different intervals of a specific attribute are selected, thus, making it possible for mining very specific rules.

Since each population individual can be of different length, the mutation and crossover operators also need to be updated. The mutation operation is performed as follows: The dimension of the mutant vector D_{Mut} needs to be calculated first. This is done with the help of the mutually exclusive selected set of population individuals, denoted as $S = \{x_{r_1}, x_{r_2}, x_{r_3}\}$. The length of the mutant vector D_{Mut} is selected randomly from one of the individuals from the set $x_{r_i} \in S$. This process is known under the name the mutant dimension/length selection strategy.

Three different mutant dimension/length selection strategies are experimented with, namely, the best fitness strategy, the shortest length strategy, and the longest length strategy. Using the best fitness selection strategy, the best individual from the set **S** is selected according to the corresponding fitness value, whereas using the longest/shortest strategy, the longest/shortest individual according to the number of attributes is used from the set **S**. This process can be expressed mathematically as:

$$D_{Mut} = \begin{cases} length\left(\max\limits_{i=1,\ldots,3} f(x_{r_i})\right), & \text{if } strategy = \text{'fitness'}, \\ \min\limits_{i=1,\ldots,3} length(x_{r_i}), & \text{if } strategy = \text{'shortest'}, \\ \max\limits_{i=1,\ldots,3} length(x_{r_i}), & \text{if } strategy = \text{'longest'}, \end{cases} \tag{18}$$

where the function $f(x_{r_i})$ denotes the fitness value, and the function $length(x_{r_i})$ of three randomly selected individuals x_{r_i}.

The index of the vector, which is used for determining the length of the mutant vector, is also used as the basis vector in the mutation operation (i.e., $x_{r_1}^{(g)}$ in Eq. 7). Thus, the used mutating strategy can be regarded as a 'DE/local-best/1' mutation strategy. The individuals, entering in the scaled difference and adding to the basis individual (i.e., $x_{r_2}^{(g)}$ and $x_{r_3}^{(g)}$ in Eq. 7), need to be adjusted in length to support the mutation operation. Both individuals are temporarily padded randomly to match the length of the mutant vector if both are shorter than D_{Mut}. On the other hand, if both individuals are longer than D_{Mut}, they are cropped accordingly.

A new crossover operation also has to be developed in order to operate on variable-length individuals, and to produce different length trial individuals at the same time. Firstly, the trial length has to be determined as $D_{Tr} = \max(length(x_i^{(g)}), length(u_i^{(g)}))$, where the dimension is referred to the longer individual among the target $x_i^{(g)}$ and mutant $u_i^{(g)}$ vectors. The new target vector $t_i^{(g)}$ is then generated using the new crossover operator, which is defined using the equation (19), as shown at the bottom of the page, where j denotes the current attribute index. The variable-length crossover defined in Eq. 19 ensures that different length individuals are produced, and, at the same time, tries to find the optimal size of the solution. Through generations, the size of generated trial individuals is decreased slowly, which, in turn, produces more optimal and interesting individuals (i.e., association rules).

IV. RESULTS

This Section describes the experimental work which was conducted during this study. The primary goal of the experimental work was to show that the results of vDE_ARM are comparable, if not better than, several other well established EA and SI algorithms. Additionally, we wanted to prove that comparable results can be obtained while using a small number of function evaluations, and, thus, the algorithm complies with the principles of Green AI. In line with this, five experiments were conducted, in which we were focused to the following objectives: (1) To discover the characteristics of the vDE_ARM, (2) To study the impact of the starting population size on the quality of the results, (3) To analyze the influence of the number of fitness function evaluations, (4) To compare the vDE_ARM with some selected SI-based and EAs from the Niapy library using the Niaarm framework [24], and (5) To investigate how the variable-length encoding of the individuals impacts the results, and how does it affect the search space in the process of rule optimization. In other words, the last experiment was designed to observe how the rule lengths are changed through the typical run.

TABLE 2. Parameters of vDE_ARM for association rule mining.

Parameter	Value
Np	15
D	dataset dependent*
$MAXFEs$	2,000
β	1
γ	1

All the experiments were evaluated according to the following five metrics: support, confidence, coverage, interestingness, and fitness value of the rule. The setting of the algorithms' parameters is reviewed in the remainder of the Section. Then, the experimental environment is described in detail. The section concludes with illustrating the results of the conducted experiments, to which also the discussion is included, where these are analyzed from the critical point of view.

A. ALGORITHM PARAMETER SETTINGS

It is important to emphasize that two distinct variants of vDE_ARM were developed for the study: The original variant, named as vDE_ARM^1, focuses solely on the considering original intervals of the numerical attributes. On the other hand, the modified variant, named as vDE_ARM^2, encompasses the capability to extract both intervals of numerical attributes, i.e., the original and their absolute complements, in particular association rules. With the two defined vDE_ARM variants, the purpose of the first experiment was to determine which variation of the observed vDE_ARM achieved the better results. The better performing variant was then considered in the next experiments.

The parameters for vDE_ARM, which were used during the experimental work, are displayed in Table 2.

NiaARM [25], which is a pure Python framework for mining Numerical Association Rules, was chosen in the comparative study, since this software ensures reproducibility due to its open source nature. NiaARM implements the ARM-DE algorithm, based on the original DE proposed in [15], but can run with any of the arbitrary nature-inspired algorithms from the NiaPy library [24].

B. EXPERIMENTAL SETUP

All the experiments were performed on a desktop computer, with the following configuration:

- Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz,

$$t_{i,j}^{(g)} = \begin{cases} u_{i,j}^{(g)}, & \text{if}(\ rand() \leq Cr or j = j_{rand})\ and\ j \leq length(u_i^{(g)}), \\ x_{i,j}^{(g)}, & \text{if}\ rand() > Cr\ and\ j \neq j_{rand},\ and\ j \leq length(x_i^{(g)}), \\ u_{i,j}^{(g)}, & \text{if}\ j \geq length(x_i^{(g)}) or j < length(u_i^{(g)}), \\ x_{i,j}^{(g)}, & \text{if}\ j \geq length(u_i^{(g)}) or j \leq length(x_i^{(g)}), \end{cases} \tag{19}$$

- RAM: 65 GB,
- Operating system: Linux Ubuntu 22.04 Jellyfish.

The vDE_ARM algorithm was implemented using the Python programming language.

The UCI ML datasets [26], listed in Table 3, were used for evaluating the performance of the vDE_ARM. The characteristics of each database are presented in terms of the number of transactions, attributes and attribute types. The attributes can be either categorical (discrete) or numerical (real). Since the vDE_ARM is able to process discrete and numerical attributes, no additional preprocessing needs to be performed in order to obtain transaction databases.

We can also point out that the databases are very colorful, as they contain a different number of transactions and attributes. There are databases that have a large number of transactions and databases containing a large number of attributes. Indeed, those databases are included into the study that include attributes of different types.

C. RESULTS

In this Section, we present a comprehensive analysis of the results obtained by the application of vDE_ARM. It is important to note that two distinct variants of vDE_ARM were considered during the experimental work, referred to as vDE_ARM^1 and vDE_ARM^2, where the former algorithm considered solely the original intervals of numerical attributes, while the latter also the absolute complements of their original intervals.

The following experiments were performed:

- comparison of the results between the vDE_ARM^2 and vDE_ARM^1 (the better variant will be used in all successive experiments),
- the influence of the population size on the obtained results of the better performing vDE_ARM variant,
- the influence of the number of fitness function evaluations on the obtained results of the better performing vDE_ARM variant,
- comparison of the results of the better vDE_ARM variant with the results of the Evolutionary Algorithms from the NiaArm framework,
- analysis of changing the rule length during the evolutionary run.

In the remainder of the Section, the results of the conducted experiments are illustrated in detail.

The best overall results were obtained empirically, when using the best fitness selection strategy as indicated from our extensive experimental work.

1) COMPARISON OF VDE_ARM^1 AND VDE_ARM^2

The goal of the experiment was to determine, which of the developed two variants of vDE_ARM produced the better results. Both variants were compared over 30 independent runs by mining the 14 transaction databases preprocessed from UCI ML datasets according to the following ARM metrics obtained by averaging the corresponding values over 30 runs for each dataset:

- average number of generated rules (#avgRules),
- average support (avgSupport),
- average confidence (avgConfidence),
- average coverage (avgCoverage),
- average interestingness (avgInterest),
- average fitness (avgFitness).

The results of mining using both variants of the vDE_ARM algorithm at $Np = 15$ are presented in Table 4, where the best results for each transaction database are emboldened. The population size $Np = 15$ was chosen, since we wanted to show that the proposed vDE_ARM was capable of finding meaningful and interesting rules, even with the lower values of the population size Np and the maximum number of fitness function evaluations *MAXFEs*. The Table illustrates the average metrics obtained by the specific algorithm on the particular dataset. The last two rows in the Tables (column 'Total') indicate the average values of the corresponding metrics.

The following conclusions can be summarized from Table 4: At first, both vDE_ARM variations generated roughly the same number of association rules, although over 30 runs, on average, the vDE_ARM^2 was able to produce more rules on three datasets, whereas vDE_ARM^1 on twelve. It is objectively hard to assess if more rules dictate the better performance directly. Furthermore, these rules also tend to be longer in the sense of the number of attributes. It is well-known that the longer association rules are often more complex and difficult to interpret than their shorter counterparts.

Additionally, it is evident that the rules produced by vDE_ARM^1 had lower support than those generated by vDE_ARM^2, in general. This suggests that the former may yield rules that are less interesting from a practical standpoint. Moreover, the values of the other statistical metrics (i.e., 'avgSupport', 'avgConfidence', 'avgCoverage', 'avgInterest', 'avgFitness') in the Table are considerably lower for vDE_ARM^1 in most cases. Considering these findings, it is apparent that the vDE_ARM^2 outperformed the results of the vDE_ARM^1 in terms of generating more concise and meaningful association rules with higher support and confidence, and, therefore, this was observed in the experiments that followed.

In summary, the frequencies of the best ranks obtained by both variants of the vDE_ARM with the population size $Np = 15$ and small number of fitness function evaluations $MAXFEs = 2000$ are illustrated in Table 5. As can be seen from Table 5, the vDE_ARM^2 yielded the better results by observing almost all the average metrics, except the metric 'avgRules'.

2) INFLUENCE OF THE POPULATION SIZE

The purpose of the study was to analyze the effect of the population size Np on the quality of mined rules measured

TABLE 3. Datasets used in the study. The abbreviations for attribute types are: D (discrete), N (numerical).

Dataset	No. of instances	No. of attributes	Attribute type (D/N)
Abalone	4177	9	DN
Autompg	392	8	DN
Balance scale	625	5	DN
Basketball	96	5	N
Bolts	40	8	N
Buying	100	40	N
Car	1728	6	D
Color histogram	68040	33	N
Color moments	68042	10	N
German	1000	20	DN
House	22784	17	N
Ionosphere	351	35	DN
Quake	2178	4	N
Wine	178	14	N

TABLE 4. Comparison of vDE_ARM^2 and vDE_ARM^1 at *Np*=15 and *MAXFEs*=2000. The reported results are averages of 30 individuals runs.

Dataset	Algorithm	#avgRules	avgSupport	avgConfidence	avgCoverage	avgInterest	avgFitness
abalone	vDE_ARM^2	101.70 ± 12.30	0.74 ± 0.04	0.88 ± 0.03	0.81 ± 0.04	0.75 ± 0.04	0.81 ± 0.03
	vDE_ARM^1	134.97 ± 26.95	0.70 ± 0.07	0.87 ± 0.04	0.76 ± 0.06	0.72 ± 0.07	0.78 ± 0.05
autompg	vDE_ARM^2	117.70 ± 17.62	0.68 ± 0.03	0.86 ± 0.03	0.75 ± 0.03	0.69 ± 0.03	0.77 ± 0.03
	vDE_ARM^1	135.00 ± 18.00	0.63 ± 0.05	0.83 ± 0.02	0.70 ± 0.06	0.65 ± 0.05	0.73 ± 0.03
balance scale	vDE_ARM^2	60.63 ± 8.43	0.59 ± 0.04	0.80 ± 0.04	0.72 ± 0.05	0.58 ± 0.04	0.70 ± 0.04
	vDE_ARM^1	65.33 ± 13.98	0.20 ± 0.02	0.50 ± 0.05	0.39 ± 0.04	0.21 ± 0.03	0.35 ± 0.03
basketball	vDE_ARM^2	84.10 ± 10.83	0.69 ± 0.04	0.85 ± 0.04	0.78 ± 0.04	0.67 ± 0.05	0.77 ± 0.04
	vDE_ARM^1	103.33 ± 18.10	0.61 ± 0.04	0.82 ± 0.04	0.71 ± 0.04	0.61 ± 0.04	0.72 ± 0.04
bolts	vDE_ARM^2	48.77 ± 4.95	0.71 ± 0.03	0.86 ± 0.03	0.82 ± 0.03	0.70 ± 0.04	0.78 ± 0.03
	vDE_ARM^1	30.47 ± 13.04	0.19 ± 0.10	0.78 ± 0.06	0.28 ± 0.10	0.61 ± 0.09	0.48 ± 0.07
buying	vDE_ARM^2	72.57 ± 8.45	0.61 ± 0.05	0.77 ± 0.06	0.74 ± 0.04	0.60 ± 0.05	0.69 ± 0.05
	vDE_ARM^1	103.03 ± 17.17	0.58 ± 0.06	0.79 ± 0.04	0.69 ± 0.06	0.59 ± 0.06	0.68 ± 0.05
car	vDE_ARM^2	50.73 ± 9.26	0.11 ± 0.02	0.46 ± 0.05	0.25 ± 0.03	0.12 ± 0.02	0.28 ± 0.03
	vDE_ARM^1	54.73 ± 7.55	0.11 ± 0.02	0.45 ± 0.06	0.25 ± 0.03	0.12 ± 0.02	0.28 ± 0.04
color histogram	vDE_ARM^2	136.63 ± 21.31	0.85 ± 0.03	0.92 ± 0.03	0.91 ± 0.03	0.85 ± 0.03	0.89 ± 0.02
	vDE_ARM^1	121.80 ± 35.00	0.20 ± 0.17	0.61 ± 0.11	0.25 ± 0.17	0.23 ± 0.18	0.41 ± 0.13
color moments	vDE_ARM^2	183.60 ± 32.78	0.80 ± 0.04	0.90 ± 0.03	0.84 ± 0.03	0.80 ± 0.04	0.85 ± 0.03
	vDE_ARM^1	223.57 ± 56.85	0.81 ± 0.04	0.90 ± 0.03	0.85 ± 0.04	0.81 ± 0.04	0.86 ± 0.03
german	vDE_ARM^2	86.90 ± 8.47	0.49 ± 0.03	0.76 ± 0.05	0.59 ± 0.04	0.49 ± 0.04	0.62 ± 0.04
	vDE_ARM^1	90.77 ± 30.80	0.34 ± 0.15	0.64 ± 0.10	0.47 ± 0.14	0.34 ± 0.15	0.49 ± 0.12
house16	vDE_ARM^2	126.40 ± 15.81	0.83 ± 0.04	0.91 ± 0.03	0.88 ± 0.03	0.82 ± 0.04	0.87 ± 0.03
	vDE_ARM^1	151.90 ± 70.23	0.54 ± 0.22	0.78 ± 0.11	0.59 ± 0.19	0.54 ± 0.22	0.66 ± 0.16
ionosphere	vDE_ARM^2	123.27 ± 14.20	0.73 ± 0.04	0.85 ± 0.04	0.82 ± 0.04	0.71 ± 0.06	0.79 ± 0.03
	vDE_ARM^1	117.43 ± 25.24	0.48 ± 0.06	0.77 ± 0.03	0.58 ± 0.07	0.54 ± 0.07	0.62 ± 0.04
quake	vDE_ARM^2	113.60 ± 13.83	0.79 ± 0.03	0.88 ± 0.03	0.87 ± 0.03	0.77 ± 0.05	0.84 ± 0.03
	vDE_ARM^1	147.63 ± 46.36	0.52 ± 0.13	0.79 ± 0.08	0.60 ± 0.11	0.53 ± 0.13	0.66 ± 0.10
wine	vDE_ARM^2	104.50 ± 11.85	0.68 ± 0.05	0.84 ± 0.04	0.77 ± 0.04	0.68 ± 0.04	0.76 ± 0.04
	vDE_ARM^1	121.10 ± 15.85	0.63 ± 0.03	0.83 ± 0.04	0.71 ± 0.04	0.63 ± 0.03	0.73 ± 0.03
Total	vDE_ARM^2	99.29 ± 37.88	0.67 ± 0.19	0.83 ± 0.12	0.76 ± 0.17	0.67 ± 0.18	0.75 ± 0.15
	vDE_ARM^1	114.36 ± 56.23	0.47 ± 0.23	0.74 ± 0.15	0.56 ± 0.21	0.51 ± 0.22	0.60 ± 0.18

by six statistical metrics. This study was performed using the better performing vDE_ARM variant (i.e., vDE_ARM^2), where the parameter population size was varied in the interval $Np = \{5, 10, 15, 20, 30, 50, 100\}$. Thus, six different instances of the algorithm were observed. The results of this experiment are presented in Table 6, where the frequencies, denoting the best ranks of the corresponding ARM metrics as obtained by mining each of the observed 14 datasets, are attached according to various population sizes.

As can be seen from Table 6, the bigger the population size *Np*, the higher the number of discovered rules. Indeed, these results are expected since the bigger initial population sizes allow examining the huge fitness landscape, and, consequently, discovering the better results [23]. The ARM metric *Confidence* has more impact on the fitness function value than the *Support*, as can be seen from calculating the Spearman correlation coefficient, because $r(\text{avgConfidence,avgFitness}) = 0.889 > r(\text{avgSupport,avgFitness}) = 0.853$.

Friedman tests [27] were employed in order to assess the statistical significance of the obtained results. The Friedman test is a two-way analysis of variances by ranks. In the first step, test statistics are calculated and converted to a rank. In the second step, post-hoc tests are conducted using these

TABLE 5. The frequencies of the best ranks obtained by both vDE_ARM variants. The reported results present the averages for 30 individual runs.

Algorithm	#avgRules	avgSupport	avgConfidence	avgCoverage	avgInterest	avgFitness
vDE_ARM^2	3	13	12	13	13	13
vDE_ARM^1	11	1	2	1	1	1

ranks. It is worth noting that lower rank values indicate better algorithm performance [28]. The second step of the analysis is performed only if the null hypothesis of the Friedman test is rejected, which assumes equal medians between the ranks of all algorithms [29].

According to Demšar [30], the Friedman test remains a reliable and robust non-parametric method for comparing multiple algorithms across various datasets. When combined with the corresponding Nemenyi post-hoc test, it facilitates a clear presentation of the statistical outcomes [31]. However, one limitation of the Friedman test is its inability to establish proper comparisons between some of the considered algorithms, due to the large number of multiple comparisons made over datasets [28].

To address this issue, a Wilcoxon two-paired non-parametric test [32] was applied as a post-hoc test after determining the control method (i.e., the algorithm with the lowest rank) using the Friedman test. The Nemenyi test, while conservative, may not detect differences in most of the experimental scenarios [28]. In contrast, the Wilcoxon test, preferred by Benavoli et al. [33] over post-hoc tests based on mean-ranks, offers greater statistical power. Consequently, in this study, the Nemenyi test is used for graphical presentation of the results, while the Wilcoxon test wass deemed more effective. Both tests were conducted with a significance level of $\alpha = 0.05$.

The conducted statistical tests encompassed the outcomes of discovering/mining association rules for all of 14 considered UCI ML datasets. In essence, the comparison involved three classifiers (i.e., the results obtained with different population sizes) based on 210 elements. The classifier's size was derived from the product $3 \times 5 \times 14$, where the first number represents the number of metrics considered (i.e., support, confidence, and fitness value), the second denotes the number of statistical measures taken into account (i.e., mean, Standard Deviation, minimum, maximum, and median), and the third indicates the total number of the UCI ML datasets used in the study.

The results of the Friedman non-parametric tests are displayed graphically in Fig. 2 and numerically in Table 7. The results of the Nemenyi post-hoc test are represented as critical difference intervals, where the results of two algorithms are statistically significant if their critical difference intervals do not overlap. To identify the best algorithm, the Friedman test's outcome was used as the control method. Subsequently, all the other algorithms were compared to this control method using the Wilcoxon two-paired non-parametric test. The results of the Wilcoxon test are depicted through corresponding p-values, where a

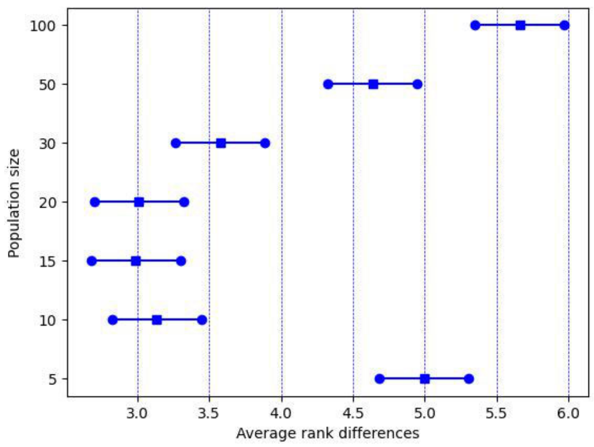


FIGURE 2. Graphical representation of the Friedman critical distances for the population size study on vDE_ARM^2.

significant difference between two algorithms is indicated when $p < 0.05$. In Table 7 the best algorithm identified by the Nemenyi post-hoc test and the control method from the Wilcoxon test are denoted with the ‡ symbol. Moreover, the presence of a significant difference between the control method and the corresponding algorithm is represented by the † symbol.

The Nemenyi post-hoc test results are presented visually through corresponding diagrams. Each diagram displays the average ranks represented by squares, while lines indicate the confidence intervals (critical differences) for the algorithms being compared. Lower rank values signify better-performing algorithms.

The Friedman test determined that the vDE_ARM^2 with population size $Np = 15$ was the best choice at $MAXFEs = 2000$, since the lowest rank was obtained by this parameter setting. This same fact was also confirmed with the Wilcoxon test, where significantly better results were obtained when compared to the results of the same algorithm using the other population sizes, except for $Np = 10$ and $Np = 20$.

3) INFLUENCE OF THE FITNESS FUNCTION EVALUATIONS

The purpose of this study was to identify the effect of the maximum fitness function evaluations $MAXFEs$ on the quality of the obtained rules yielded by vDE_ARM^2 by a fixed population size Np. Two different population sizes were employed in the experiments, i.e., $Np = 15$, and $Np = 100$. The former population size was determined to be the best by the selected small number of fitness function evaluation $MAXFEs = 2000$ in the last experiment, and allowed vDE_ARM^2 to converge fast. The latter represents the

TABLE 6. The frequencies of the best ranks obtained by vDE_ARM^2 with the various population sizes. The reported results present averages for 30 individual runs on 14 selected UCI ML datasets.

Np	#avgRules	avgSupport	avgConfidence	avgCoverage	avgInterest	avgFitness
5	0	0	1	0	0	0
10	0	6	3	6	5	4
15	0	3	4	5	4	3
20	0	4	4	2	4	6
30	0	0	0	1	0	1
50	0	1	0	0	1	0
100	14	0	0	0	0	0

TABLE 7. Friedman and Wilcoxon statistical tests for the population size study on vDE_ARM^2.

Np	Friedman	Nemenyi		Wilcox	
		CD	Sig.	p-value	Sig.
5	5.00	[4.68, 5.31]	†	$\ll 0.05$	†
10	3.13	[2.82, 3.44]		0.41	
15	2.99	[2.67, 3.30]	‡	∞	‡
20	3.01	[2.70, 3.32]		0.04	†
30	3.58	[3.27, 3.89]	†	$\ll 0.05$	†
50	4.64	[4.33, 4.95]	†	$\ll 0.05$	†
100	5.66	[5.35, 5.97]	†	$\ll 0.05$	†

TABLE 8. Friedman and Wilcoxon statistical tests by varying the maximum fitness function evaluations for vDE_ARM^2 with $Np = 15$.

$MAXFEs$	Friedman	Nemenyi		Wilcox	
		CD	Sig.	p-value	Sig.
2,000	4.08	[3.82, 4.34]	†	$\ll 0.05$	†
5,000	3.05	[2.79, 3.31]	‡	∞	‡
10,000	3.48	[3.22, 3.74]		0.01	†
20,000	3.48	[3.22, 3.74]		0.01	†
50,000	3.45	[3.19, 3.71]		0.01	†
100,000	3.45	[3.19, 3.71]		0.01	†

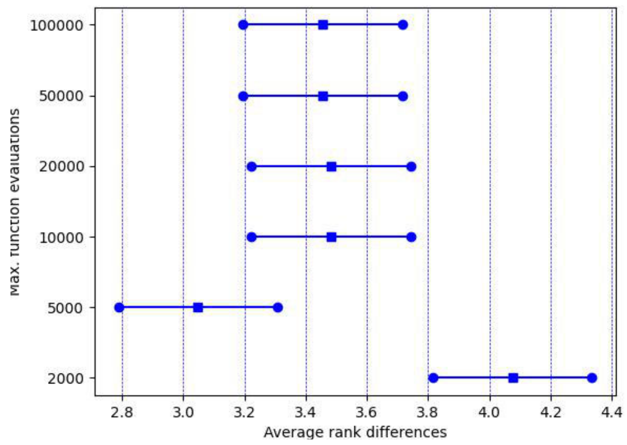


FIGURE 3. Results of Friedman non-parametric statistical test by varying the maximum function evaluations for vDE_ARM^2 with $Np = 15$.

standard setting of this parameter by DE the community and allows the search space to be explored in more detail. On the other hand, the parameter *MAXFEs* was varied in the interval $MAXFEs = \{2000, 5000, 10000, 20000, 50000, 100000\}$ in this experiment. Thereby, six different instances of the algorithm were observed for a particular population size.

The results of the first experiment are gathered numerically in Table 8 and graphically in Figure 3.

The results in both, i.e., Table 8 and Figure 3, indicate that the best association rules by vDE_ARM^2 with population size $Np = 15$ were mined when the number of fitness function evaluations was set to *MAXFEs=5000*. The results were not significantly different, only when this parameter was set to *MAXFEs=2000*. This means that the selected population size *MAXFEs=2000* was underestimated in the first experiment. The same setting of these parameters yielded results that were also significantly different for all the other various settings of the parameter. This fact was expected in some way, since a lower population size tends to converge faster. Consequently, the algorithm probably either reached convergence or started discovering uninteresting rules in the later stages of the evolutionary process by using the higher values of *MAXFEs*.

The results of the vDE_ARM^2 with the population size $Np = 100$ and the maximum number of fitness function evaluations varied in the interval $MAXFEs = \{2000, 5000, 10000, 20000, 50000, 100000\}$ are illustrated graphically in Figure 4 and numerically in Table 9. For the selected population size, the best results were obtained by the vDE_ARM^2 with the highest $MAXFEs = 100000$. This evidence was also expected, since the larger pool of candidate solutions were explored and also slower convergence was achieved. This gives the vDE_ARM^2 search process the necessary time to examine the fitness landscape thoroughly, and to discover meaningful and important rules. According to the Wilcoxon test, all the variants of vDE_ARM^2 with the other *MAXFEs* were significantly different from the observed best result.

4) COMPARISON OF THE BEST VDE_ARM VARIANT WITH EVOLUTIONARY ALGORITHMS FROM THE NIAARM FRAMEWORK

The purpose of this experiment was two-fold: (1) To analyze the runtime of the proposed vDE_ARM^2 with respect to the quality of the obtained results, (2) To compare the performance of vDE_ARM^2 with the selected NI algorithms taken from the NiaARM framework.

The goal of the first part of the experimental study was to develop a Green ML method, which will be able to compete with recent approaches in the sense of lower computational cost, and, indirectly, the lower carbon footprint. In line with

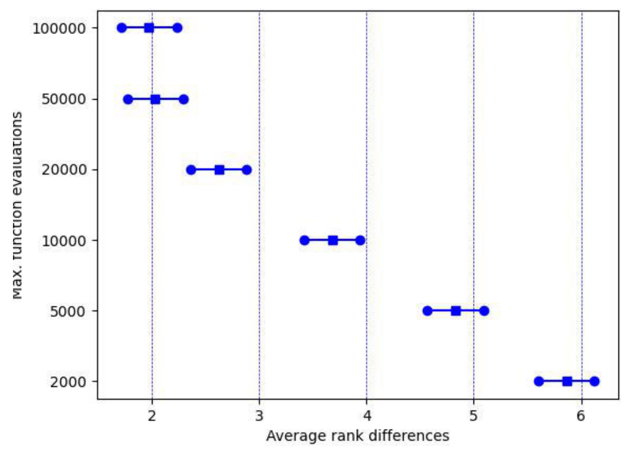


FIGURE 4. **Results of the Friedman non-parametric statistical test by varying the maximum function evaluations for vDE_ARM^2 with $Np = 100$.**

TABLE 9. **Friedman and Wilcoxon statistical tests by varying the maximum fitness function evaluations for vDE_ARM^2 with $Np = 100$.**

MAXFEs	Friedman	Nemenyi		Wilcox	
		CD	Sig.	p-value	Sig.
2,000	5.87	$[5.61, 6.13]$	†	$\ll 0.05$	†
5,000	4.83	$[4.57, 5.09]$	†	$\ll 0.05$	†
10,000	3.68	$[3.42, 3.94]$	†	$\ll 0.05$	†
20,000	2.62	$[2.36, 2.88]$	†	$\ll 0.05$	†
50,000	2.03	$[1.77, 2.29]$		$\ll 0.05$	†
100,000	1.97	$[1.71, 2.23]$	‡	∞	‡

this, two variants of vDE_ARM^2 were considered utilizing the following parameter setups:

- Green vDE_ARM^2: using $Np = 15$ and $MAXFEs = 2000$,
- Red vDE_ARM^2: using $Np = 100$ and $MAXFEs = 100000$,

The first variant (denoted as Green) is more suitable for computers of limited resources (e.g., microcomputers), while the second one (denoted as Red) demands full computational resources.

The run times of both methods in seconds are summarized in Table 10, along with all the other observed metrics. It can be concluded that the Green vDE_ARM^2 was able to achieve very similar results compared to the Red vDE_ARM^2 in a much shorter time. In fact, there was a nearly 3,300 % decrease in computational time. The only metric which performed worse was the number of obtained ARs. However, this was expected, since a larger search space can be traversed when using more function evaluations. The obtained rules with the Red vDE_ARM^2 were also analyzed manually. The results of the analysis showed that many very similar ARs were produced by the vDE_ARM^2, whereas just small changes appeared in the boundaries of specific attributes.

In the second part of the experimental study, the results of vDE_ARM^2 were compared with the results obtained by several selected SI-based and EAs from the NiaARM framework [24]. The selected algorithms were Differential Evolution (DE) [20], Particle Swarm Optimization (PSO) [34],

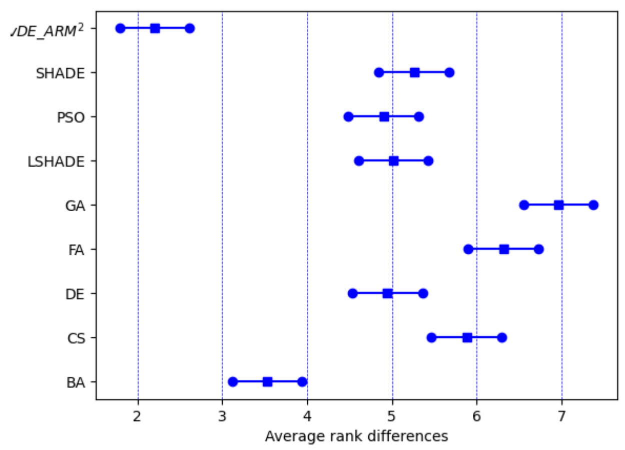


FIGURE 5. **Graphical representation of Friedman critical distances for the comparison study of vDE_ARM^2 with the NiaARM framework ($Np = 15$, $MAXFEs = 2,000$).**

Bat Algorithm (BA) [35], Cuckoo Search (CS) [36], Genetic Algorithm (GA) [37], [38], Success-History based Adaptive DE (SHADE) [39], and Success-History based Adaptive DE using Linear Population Size Reduction (LSHADE) [40]. All the selected algorithms in the comparative study were used with their default parameter settings as provided by the NiaARM framework. Although Fitness-Distance Balance (FDB) meta-heuristics algorithms were also taken into consideration [41], they were not included into our study since these are implemented in MatLab, and, thus, not the best from the performance point of view. On the other hand, the experiments revealed that the better algorithms in solving global optimization problems are not also better at solving the ARM problems. The only parameter that was adjusted was the population size, set to $Np = 15$. Each algorithm in the comparison was executed 30 times, so all the reported results are an average of 30 runs. The maximum number of function evaluations was set to $MAXFEs = 2000$ and $MAXFEs = 100000$ similar to the vDE_ARM^2. The special parameters of GA were set as follows: the probability of crossover $p_c = 0.25$, and the probability of mutation $p_m = 0.25$..

The statistical results of comparing algorithms for settings $Np = 15$ and $MAXFEs = 2000$ are collated in Table 11 and Figure 5, while for the settings $Np = 15$ and $MAXFEs = 100000$ in Table 12 and Figure 6. In both tests the proposed vDE_ARM^2 obtained the highest rank using the Friedman test, while also performing significantly better when comparing algorithms according to the Wilcoxon test.

It is worth mentioning that each rule, which was generated within the NiaARM framework and was also feasible, was put in the final archive, so, by design, many very similar rules could be generated. Our approach was somewhat different. Although also all feasible rules were being used, they were put in the final archive after each passed generation. Let us emphasize that this means that the proposed method will always generate a lower number of rules compared to the NiaARM framework.

TABLE 10. Comparison between the red and green vDE_ARM^2.

vDE_ARM^2	#avgRules	avgSupport	avgConfidence	avgCoverage	avgInterest	avgFitness	avgTime [sec]
Green	99.29 ± 37.88	0.67 ± 0.19	0.83 ± 0.12	0.76 ± 0.17	0.67 ± 0.18	0.75 ± 0.15	6.19 ± 9.64
Red	653.48 ± 220.29	0.67 ± 0.18	0.83 ± 0.12	0.76 ± 0.16	0.67 ± 0.18	0.75 ± 0.15	210.32 ± 255.04
Difference	558.14 %	0.07 %	0.17 %	-0.26 %	0.31 %	0.13 %	3,330.26 %

TABLE 11. Friedman and Wilcoxon statistical tests for the comparison study of vDE_ARM^2 with the NiaARM framework ($Np = 15$, $MAXFEs = 2000$).

Algorithm	Friedman	Nemenyi		Wilcox	
		CD	Sig.	p-value	Sig.
BA	3.53	$[3.11, 3.94]$	†	$\ll 0.05$	†
CS	5.88	$[5.46, 6.29]$	†	$\ll 0.05$	†
DE	4.94	$[4.53, 5.36]$	†	$\ll 0.05$	†
FA	6.31	$[5.90, 6.73]$	†	$\ll 0.05$	†
PSO	4.90	$[4.49, 5.31]$	†	$\ll 0.05$	†
GA	6.96	$[6.55, 7.38]$	†	$\ll 0.05$	†
SHADE	5.26	$[4.85, 5.68]$	†	$\ll 0.05$	†
LSHADE	5.01	$[4.60, 5.43]$	†	$\ll 0.05$	†
vDE_ARM^2	2.20	$[1.79, 2.61]$	‡	∞	‡

TABLE 12. Friedman and Wilcoxon statistical tests for the comparison study of vDE_ARM^2 with the NiaARM framework ($Np = 15$, $MAXFEs = 100000$).

Algorithm	Friedman	Nemenyi		Wilcox	
		CD	Sig.	p-value	Sig.
BA	3.72	$[3.30, 4.13]$		$\ll 0.05$	†
CS	4.85	$[4.44, 5.27]$	†	$\ll 0.05$	†
DE	5.82	$[5.41, 6.24]$	†	$\ll 0.05$	†
FA	4.37	$[3.96, 4.79]$	†	$\ll 0.05$	†
PSO	5.26	$[4.85, 5.68]$	†	$\ll 0.05$	†
GA	7.01	$[6.60, 7.43]$	†	$\ll 0.05$	†
SHADE	5.42	$[5.01, 5.84]$	†	$\ll 0.05$	†
LSHADE	5.42	$[5.01, 5.84]$	†	$\ll 0.05$	†
vDE_ARM^2	3.11	$[2.70, 3.53]$	‡	∞	‡

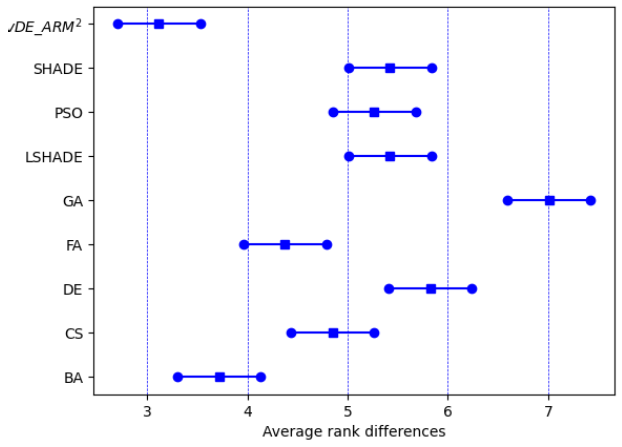


FIGURE 6. Graphical representation of Friedman critical distances for the comparison study of vDE_ARM^2 with the NiaARM framework ($Np = 15$, $MAXFEs = 100,000$).

The biggest difference between the vDE_ARM^2 and the comparison methods is in the number of generated rules. The difference emerged from the fact that all feasible candidate rules were added in the final archive in the competing methods. It is also apparent that the proposed vDE_ARM^2 could consistently find rules, even on the harder datasets (with a large number of attributes), whereas the other algorithms in the comparative study often failed.

The runtime of the Green vDE_ARM^2 variant was also compared to the runtimes of the NI algorithms from the NiaARM framework. All the algorithms in the study were run using the same parameters as the Green vDE_ARM^2 (i.e., $Np = 15$ and $MAXFEs = 2000$). The results of this comparison are collated in Table 13, which show that the Green vDE_ARM^2 was able to provide the best results by considering all the statistical metrics, in much less time.

It seems that the comparing BA algorithm obtained fairly good results, when compared to vDE_ARM^2, mainly because of its advanced exploration evolutionary operators.

5) RULE LENGTH SIZE CHANGING THROUGHOUT THE EVOLUTIONARY PROCESS

The goal of this experiment was to demonstrate how the lengths of individuals change during the evolutionary process in regard to variable-length encoding. Figure 7 illustrates the variations in rule sizes across successive generations for each experimental dataset for vDE_ARM^2. The mean and standard deviation (std) of 30 independent runs were calculated for each generation. The findings revealed that at the initial stages of the evolutionary process, the generated rules exhibit similar lengths (low std) and encompass numerous attributes. However, as the evolutionary search progresses, the discovered rules tend to possess different lengths (high std) and involve a reduced number of attributes. These outcomes suggest that the proposed vDE_ARM^2 algorithm, initially explores longer and more general rules. Yet, as it advances further in the evolutionary process, it tends to discover more intricate and specific rules.

V. DISCUSSION

The obtained results prove that the proposed vDE_ARM (specifically the vDE_ARM^2 variant) was successful in obtaining good, and quality AR. The proposed method was demonstrated to be more efficient than the comparison algorithms in the NiaARM framework in almost all aspects.

Notwithstanding, it can be said that an advantage of our method is in its straightforward structure, and the use of simple evolutionary operators. Additionally, unlike some other optimization algorithms, the vDE_ARM algorithm, as an integral part of the evolutionary ARM, has only two parameters (i.e. parameters Np and $MAXFEs$). Undoubtedly,

TABLE 13. Comparison between the green vDE_ARM^2 and the Niaarm framework.

Algorithms	#avgRules	avgSupport	avgConfidence	avgCoverage	avgInterest	avgFitness	avgTime [sec]
BA	1,036.08 ± 604.27	0.49 ± 0.36	0.79 ± 0.23	0.60 ± 0.35	0.50 ± 0.35	0.64 ± 0.27	6.69 ± 5.37
CS	787.94 ± 489.14	0.21 ± 0.16	0.59 ± 0.20	0.35 ± 0.18	0.24 ± 0.17	0.40 ± 0.17	7.78 ± 5.92
DE	489.29 ± 249.94	0.30 ± 0.16	0.63 ± 0.17	0.48 ± 0.20	0.30 ± 0.16	0.46 ± 0.15	6.74 ± 5.63
FA	685.79 ± 451.89	0.13 ± 0.11	0.45 ± 0.16	0.33 ± 0.16	0.15 ± 0.11	0.29 ± 0.12	6.88 ± 5.41
PSO	317.40 ± 336.46	0.40 ± 0.24	0.72 ± 0.20	0.53 ± 0.26	0.40 ± 0.23	0.56 ± 0.20	6.85 ± 5.89
GA	372.45 ± 299.31	0.05 ± 0.04	0.30 ± 0.16	0.25 ± 0.11	0.07 ± 0.07	0.18 ± 0.09	14.56 ± 9.61
SHADE	1069.35 ± 487.37	0.47 ± 0.30	0.75 ± 0.20	0.57 ± 0.29	0.49 ± 0.29	0.61 ± 0.23	9.48 ± 7.08
LSHADE	963.47 ± 428.30	0.50 ± 0.31	0.77 ± 0.20	0.59 ± 0.30	0.52 ± 0.30	0.63 ± 0.24	14.81 ± 9.20
Green vDE_ARM^2	99.29 ± 37.88	0.67 ± 0.19	0.83 ± 0.12	0.76 ± 0.17	0.67 ± 0.18	0.75 ± 0.15	6.19 ± 9.64

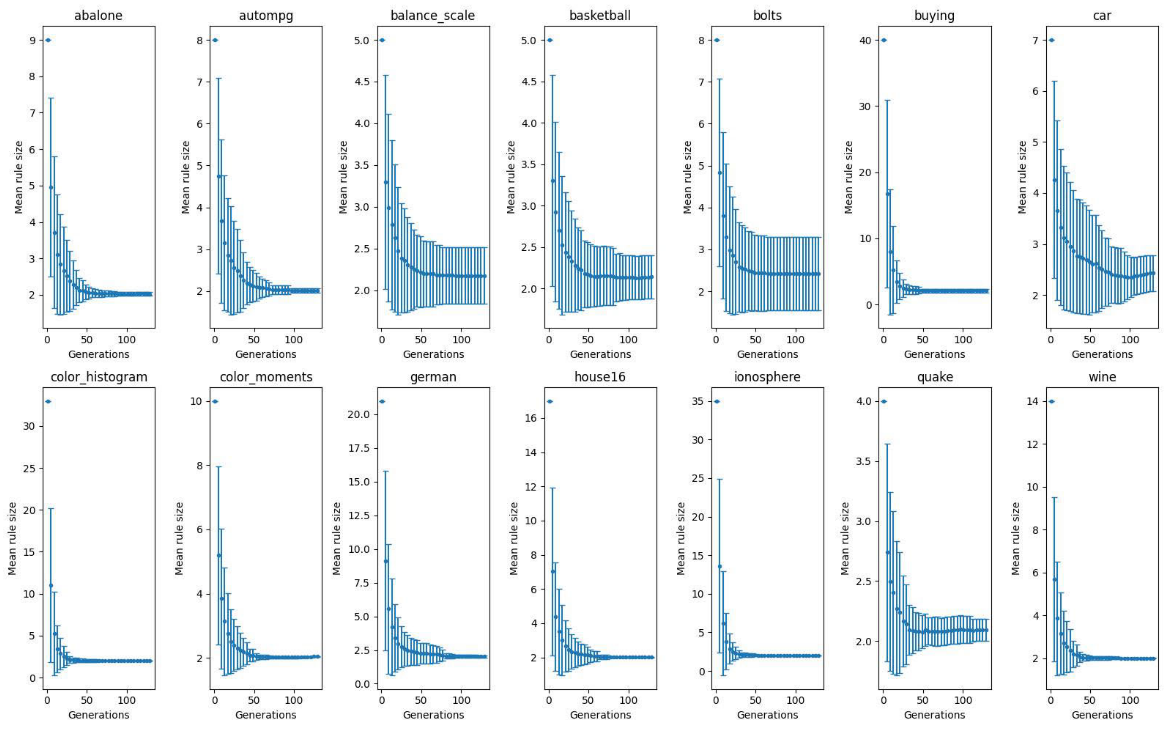


FIGURE 7. Rule size changes through generations for vDE_ARM^2.

the advantage of using the vDE_ARM algorithm for ARM instead of other EAs is in its ability to maintain a diverse population of vectors easily. Namely, the newly created solutions participate in the creation of trial solutions instantly. This ensures an elitism within the reproduction. By ARM, this means that good rules will be discovered early, and that the search process is continued within the neighbourhood of those rules in subsequent generations of the vDE_ARM algorithm. Also an important part of the vDE_ARM is its ability to produce rules, which can contain different numerical attribute intervals.

It can also be stated that the population size has a significantly notable effect on the quality of the mined rules using the vDE_ARM^2 algorithm. This is evident from the statistical results in Tables 6 and 7. Using a lower population size means that the evolutionary search process will focus more on the local neighborhood of good solutions (rules), whereas a bigger population size will put more rules with a low fitness value in the archive. Thus it can be concluded that the lower population size is favorable when using the vDE_ARM algorithm.

The experimental work also indicated, that increasing the maximum number of function evaluations, does not necessarily mean better results. This fact is evident by observing the results in Table 8, where the lower value of *MAXFEs* is also a better choice with respect to the Green AI directive (see results in Tables 10 and 13).

When the results of the proposed vDE_ARM^2 were analyzed from the stability analysis point of view [42], where these are estimated according to metrices, like Success Rate (SR), Mean Fitness Evaluation number of success solutions (MFE), and Mean Computation Duration of success solutions (MCD), we established that the NARM problem does not have only one unique solution. This means that there can be more successful solutions distinguished between each other by various NARM metrics. Consequently, we cannot use the proposed metrics to analyze the stability of the obtained results due to absence of the best solution. As a result, we are interested in the quality solutions estimated regarding, e.g., support or confidence metrics, found as fast as possible. In line with this, the proposed vDE_ARM^2 fulfills the posted demand completely. Furthermore, we showed that the integrated mechanism for variable-length individual encoding and representation is beneficial, since the proposed algorithm also outperformed the more recent state-of-the-art algorithms like Shade and L-Shade. In general, the implemented mechanism built into either an evolutionary or SI-based algorithm improves the results by solving ARM problems more than using the more sophisticated algorithm.

A shortcoming of the proposed vDE_ARM is undoubtedly in its inability to produce more feasible rules during the evolutionary process, although being very close in the genotype space. The reason lies in many critical sanity checks when evaluating each rule. For example, with the proposed variable-length rule encoding, it can happen that an attribute is selected for both the antecedent and the consequent, which is illegal.

VI. CONCLUSION

The novel vDE_ARM^2 algorithm for mining association rules was presented in this paper. The proposed representation of the association rules support mining negative and positive intervals of the numerical attributes, while also being able to work with discrete attributes. The produced association rules are interesting and simple, while offering good coverage of the datasets. The rules are evaluated using a single-objective fitness function, by using a weighted sum of support and confidence metrics. The weights can be adjusted by the users, giving them the total control over the importance of said measures in finding rules.

The proposed vDE_ARM was tested on several publicly available UCI ML datasets, and compared to several nature inspired algorithms, which are available in the NiaARM framework. The experiments show promising results compared to other nature inspired algorithms, based on the used evaluation metrics, and also provide interesting rules which include a low number of attributes. Last but not least, the algorithm complies with the principles of Green AI in the sense of quickly finding a solution and, thus, reducing the calculation complexity that is connected with the higher energy consumption and indirectly also the increased carbon footprint.

For the future work, we would like to test the proposed method on larger UCI ML datasets, perform a study of parameter settings of the algorithm on the quality of the mined rules based on the new objective functions.

REFERENCES

[1] A. Syed, K. Gillela, and C. Venugopal, "The future revolution on big data," *Future*, vol. 2, no. 6, pp. 2446–2451, 2013.

[2] H. G. Miller and P. Mork, "From data to decisions: A value chain for big data," *IT Prof.*, vol. 15, no. 1, pp. 57–59, Jan. 2013.

[3] J. Han, J. Pei, and H. Tong, *Data Mining: Concepts and Techniques* (The Morgan Kaufmann Series in Data Management Systems). Amsterdam, The Netherlands: Elsevier, 2022.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. Int. Conf. Very Large Data Bases*, vol. 1215. 1994, pp. 487–499.

[5] A. Telikani, A. H. Gandomi, and A. Shahbahrami, "A survey of evolutionary computation for association rule mining," *Inf. Sci.*, vol. 524, pp. 318–352, Jul. 2020.

[6] P. Dhar, "The carbon impact of artificial intelligence," *Nature Mach. Intell.*, vol. 2, no. 8, pp. 423–425, Aug. 2020.

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," 2019, *arXiv:1906.02243*.

[8] 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.

[9] I. Fister Jr. and I. Fister, "A brief overview of swarm intelligence-based algorithms for numerical association rule mining," in *Applied Optimization and Swarm Intelligence* (Springer Tracts in Nature-Inspired Computing), E. Osaba and X. S. Yang, Eds. Singapore: Springer, 2021, pp. 47–59.

[10] B. Alatas, E. Akin, and A. Karci, "MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 646–656, Jan. 2008.

[11] 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.

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

[13] K. E. Heraguemi, N. Kamel, and H. Drias, "Multi-swarm bat algorithm for association rule mining using multiple cooperative strategies," *Int. J. Speech Technol.*, vol. 45, no. 4, pp. 1021–1033, Dec. 2016.

[14] U. Mlakar, M. Zorman, I. Fister, and I. Fister, "Modified binary cuckoo search for association rule mining," *J. Intell. Fuzzy Syst.*, vol. 32, no. 6, pp. 4319–4330, May 2017.

[15] 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*. Madrid, Spain: Springer, 2018, pp. 79–88.

[16] S. A. Alwahab, H. J. Nassr, Z. H. Thanon, and B. J. Al-Khafaji, "Genetic based method for mining association rules from text," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 9, pp. 3166–3171, 2021.

[17] Z. F. Sokhangoee and A. Rezapour, "A novel approach for spam detection based on association rule mining and genetic algorithm," *Comput. Electr. Eng.*, vol. 97, Jan. 2022, Art. no. 107655.

[18] C. K. H. Lee, K. L. Choy, G. T. S. Ho, and C. H. Y. Lam, "A slippery genetic algorithm-based process mining system for achieving better quality assurance in the garment industry," *Exp. Syst. Appl.*, vol. 46, pp. 236–248, Mar. 2016.

[19] A. Ghosh, S. Dehuri, and H. Kalia, "Fitness inheritance in multi-objective genetic algorithms: A case study on fuzzy classification rule mining," *Int. J. Adv. Intell. Paradigms*, vol. 23, no. 1/2, p. 89, 2022.

[20] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[21] 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*, M. Song, I.-Y. Song, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham, Switzerland: Springer, 2020, pp. 38–47.

[22] D. J. Prajapati, S. Garg, and N. C. Chauhan, "Interesting association rule mining with consistent and inconsistent rule detection from big sales data in distributed environment," *Future Comput. Informat. J.*, vol. 2, no. 1, pp. 19–30, Jun. 2017.

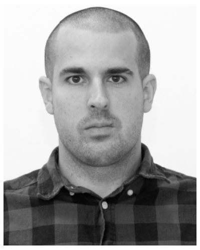
[23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing* (Natural Computing Series). Berlin, Germany: Springer, 2015.

[24] G. Vrbancic, L. Brezocnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *J. Open Source Softw.*, vol. 3, no. 23, p. 613, Mar. 2018.

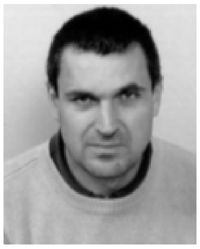
[25] Ž. Stupan and I. Fister Jr., "NiaARM: A minimalistic framework for numerical association rule mining," *J. Open Source Softw.*, vol. 7, no. 77, p. 4448, Sep. 2022.

[26] D. Dua and C. Graff, "UCI machine learning repository," Univ. California, Irvine, School Inf. Comput. Sci., Tech. Rep., 2017.
[27] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, Dec. 1937.
[28] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[29] S. Garcia and F. Herrera, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.
[30] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[31] P. Nemenyi, *Distribution-Free Multiple Comparisons*. Princeton, NJ, USA: Princeton Univ., 1963.
[32] D. Rey and M. Neuhauser, *Wilcoxon-Signed-Rank Test*. Berlin, Germany: Springer, 2011, pp. 1658–1659.
[33] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 152–161, 2016.
[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Nov. 1995, pp. 1942–1948.
[35] X. Yang and A. Hossein Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Computations*, vol. 29, no. 5, pp. 464–483, Jul. 2012.
[36] X.-S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014.
[37] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.
[38] S. Ventura and J. M. Luna, *Pattern Mining with Evolutionary Algorithms*, 1st ed. Berlin, Germany: Springer, 2018.
[39] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 71–78.
[40] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.
[41] H. T. Kahraman, S. Aras, and E. Gedikli, "Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms," *Knowl.-Based Syst.*, vol. 190, Feb. 2020, Art. no. 105169.
[42] S. Gürgen, H. T. Kahraman, S. Aras, and I. Altin, "A comprehensive performance analysis of meta-heuristic optimization techniques for effective organic Rankine cycle design," *Appl. Thermal Eng.*, vol. 213, Aug. 2022, Art. no. 118687.

UROŠ MLAKAR (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Maribor, in 2010, 2014, and 2019, respectively. He is currently an Assistant Professor with the Faculty of Electrical Engineering and Computer Science, Laboratory for System Software. He has participated in the development of over 30 scientific articles. His research interests include evolutionary computation, data mining, and image processing. He is also a reviewer of several international journals.



IZTOK FISTER JR. (Member, IEEE) 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 sports science. He has acted as a program committee member of more than 30 international conferences. Furthermore, he is a member of the editorial boards of three different international journals.



IZTOK FISTER (Member, IEEE) received the degree in computer science from the University of Ljubljana, in 1983, and the Ph.D. degree from the Faculty of Electrical Engineering and Computer Science, University of Maribor, in 2007. Since 2010, he has been a Teaching Assistant with the Computer Architecture and Languages Laboratory, Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include computer architectures, programming languages, operational research, artificial intelligence, and evolutionary algorithms. In his research areas, he has published many original scientific papers, review papers, book chapters, edited books, and contributed to more prominent international scientific conferences around the world. He is also one of the presidents of the international student conference in computer science. In many journals with impact factors, he has been promoted as an outstanding reviewer.