Online numerical association rule miner

A b s t r a c t

Green AI refers to those AI methods that are friendly to the environment, i.e., are capable to keep the con- sumption of electrical energy at a minimum. In this sense, a new numerical association rule miner is pro- posed that presents a combination of the already existing offline uARMSolver, belonging to a Red AI class, and a newly developed onlineNARM miner representing the new Green AI. The former is devoted to exhaustive search of the evolutionary solution space, while the latter for faster exploiting of already explored search space. The experimental results on four transaction databases showed that, by sacrificing the quality of the results by 0:7 %, by the onlineNARM we can obtain the results almost 85:0 % faster than with the uARMSolver in the best test scenario.

1. Introduction

The implications and expanding the number of applications in Artificial Intelligence (AI) have an impact on almost every step of human endeavor. The main methods under the umbrella of AI have become a central point for researchers and practitioners across the globe. The accuracy of the modern AI solutions now achieve limits that were unimaginable several decades ago. For example, AI image recognition methods are nowadays going closer to or even beyond the accuracy achieved by humans. On the other hand, AI is now helping in searching for new therapeutics [1] at a much faster pace than it was in the past. Fighting climate change is another role of AI these days, which is also researched heavily [2–4]. Indeed, climate change should be considered as one of the most significant challenges the humanity is facing today [5].

Even though AI influences on our lives drastically in a positive sense, every positive thing has its price. In the case of AI, one of those is high global energy consumption to emphasize that this process is still happening today. The accuracy obtained by AI solutions is a subject of many experiments, algorithm adaptations, parameter tunings, and optimizations that are computationally complex processes. Significantly, modern deep learning, neural architecture search, or AutoML pipelines are amongst the major contributors to the global carbon footprint that AI addresses. Therefore, AI impacts climate change in more than one way. Interestingly, Nordgren [6] pinpointed the dual role of AI concerning climate change: on the one hand, AI contributes to the fight against climate change, while on the other, it also contributes to climate change itself.

Human life has become endangered due to climatic changes on the Earth, especially global warming. A solution to this problem presents the so-called ”carbon law”, according to which emissions should be halved every decade from 2020 to 2050, when they should fall to around zero [7]. The name of the law was given in reference to exponential technologies, whose output per size is accelerating constantly, e.g., the silicon chip following Moore’s law, or modern technologies such as 5G, the Internet of Things, AI, which follows their exponential increases of output per size in specific periods. The most advantages towards a carbon-free society can be achieved by carbon-free digital solutions in domains like: Energy, Manufacturing, Agriculture, Building, Services, Transport and Traffic Management. Interestingly, these technologies, who are also part of AI, can help to reduce global carbon emissions by even 15 % [8].

The issues mentioned above are nowadays being investigated extensively in research communities. Researchers are aware that training large-scale computational models consumes a lot of electricity, contributing to the global carbon footprint. In recent years, the term Red AI has been coined, which exposes [9] that numerous AI methods are unfriendly to the environment [10]. On the contrary, all the AI methods that are friendly to the environment are referred to as Green AI. In line with this, different industries and start-ups are also trying to follow Green AI due to the environmen-

.

tal impact, the reduction of energy, and potentially saving money. This will undoubtedly come to the fore as electricity becomes more expensive due to several world crises and high inflation.

How to reduce computational power, to merge some critical steps in training computational models, to simplify preprocessing, or even Machine Learning (ML) pipelines, are topical questions that present challenges for the research community nowadays. Some solutions are smaller datasets used for training, lighter models [11], more optimized preprocessing tasks, different hardware architectures, or even different neural network architectures, e.g., spiking neural networks [12].

Association Rule Mining (ARM) is a part of data mining, where associations that are presented as implications are searched for within the transaction databases [13]. Usually, the transaction databases can represent big data when dealing primarily with market basket analyses or other business data [14]. We try to obtain new insights that help us to adapt to business needs. Nowadays, there are two primary methods for ARM, which can be classified in two groups, i.e., deterministic and stochastic. The deterministic methods ensure that we find an optimum solution in an unpredictable time. Still, we sacrifice computational and time resources, while the stochastic nature of an algorithm does not guarantee the best solution, but is much less expensive in terms of computational resources. Numerical Association Rule Mining (NARM) is a variant of canonical ARM, which can also operate with numerical and categorical attributes concurrently [15,16]. The main benefit of this approach is that it can generally provide more accurate results. A considerable portion of algorithms for NARM are based on stochastic nature-inspired population-based metaheuristics, which are well-known algorithms capable for exploring a larger search space. This is needed when transaction databases consist of many numerical attributes.

All these arguments can confirm that NARM is computationally a very expensive process, also due to the fact that the population-based algorithms need approximately between 25 to 51 independent runs to minimize the effect of randomness. Thus, there are challenges in reducing the complexity of the NARM process to simplify and reuse rule mining models to follow the Green AI.

This paper is an extension of our conference paper that was presented at the SOCO 2021 conference [17]. The primary purpose of the conference paper was to develop a Green AI method by introducing the NARM modeling that allows reusing the NARM models online instead of always generating these anew. A new algorithm named onlineNARM was developed and applied to the UCI ML Wine dataset [21]. The onlineNARM mined the association rules of similar quality compared with the results achieved by the offline uARMSolver. This was also better in the lower number of the mined association rules and the lesser computational time. In the current paper, we go a step further, and provide the following contributions/extensions to our previous work:

- the onlineNARM is hybridized with many mechanisms that allow it exploit the problem search space effectively,
- the offline uAMSolver generates a state database, representing the basis on which the onlineNARM can start (raising the new Green AI method),
- the results can be tested on four different UCI ML datasets,
- analyzing the results of the proposed method in the sense of the time and space,
- identifying the operation of the proposed method in the sense of exploration/exploitation.In the remainder of the paper, its structure is as follows: Section 2 discusses the background information needed for understanding the subjects that follow. The design and implementation of the onlineNARM is presented in Section 3. The experiments and results are the subjects of Section 4. The paper concludes with Section 5, in which the performed work is summarized and the directions are outlined for the future work.

2. Background information

The background information, needed for understanding the subjects that follows, is discussed in this section. The fundamentals of Differential Evolution (DE) are reviewed first. Actually, this algorithm is used for online NARM due to its simplicity, efficiency and adaptational ability. The basics of NARM are illustrated next. The section is concluded with a description of a uARMSolver framework running on most Linux distributions as a package for solving the NARM problems [18].

2.1. Differential Evolution

DE belongs to a class of stochastic nature-inspired population-based algorithms. Although it bases on a strong mathematical definition of vector differences, it is considered as an evolutionary algorithm because its crossover, mutation, and selection operators play a similar role as presented by Darwinian natural evolution [19]. The algorithm was developed by Storn and Price in 1995 [20] and gained quickly a big community of developers quickly by solving the continuous as well as discrete hard problems. The adaptational ability of the DE cause many new variants to emerge since its birth.

The DE is a population-based algorithm, where its population consists of NP real-valued vectors. Thus, each solution represents a particular solution of the problem to be solved. Operators of crossover, mutation, and selection are applied to this population. There are many ways how to apply the variation operators (i.e., crossover and mutation) to the population of solutions. The basic so-called 'rand/1/bin' mutation strategy, for instance, selects two solutions randomly, and adds their scaled difference to the third solution, in other words:

$$u_i^{(t+1)} = x_{r0}^{(t+1)} + F \cdot (x_{r1}^{(t+1)} - x_{r2}^{(t+1)}); \quad \text{for } i = 1, \ldots, Np; \tag{1}$$

where $u_i^{(t+1)}$ denotes the trial vector, $F \in [0.1, 1.0]$ is the scaling factor regulating the rate of modification, Np denotes the population size and $r0$, $r1$, $r2$ are randomly selected values in the interval $1, \ldots, NP$.

Two kinds of crossover are used in the DE algorithm: binomial (denoted as 'bin') and exponential (denoted as 'exp'). The binomial crossover copies on the same position layed elements, either from the trial or target vector, while the exponential crossover is similar to the 1-point crossover in genetic algorithms and copies all the elements from the trial vector until a probability of crossover is true. The other elements are copied from the target vector. The binomial crossover can be expressed mathematically as:

$$w_{ij}^{(t+1)} = \begin{cases} u_{ij}^{(t+1)} & \text{rand}_j(0,1) \leq CR \lor j = j_{rand}; \\ x_{ij} & \text{otherwise}; \end{cases} \tag{2}$$

where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution, while a condition $j = j_{rand}$ ensures that the trial vector differs from the original solution $x_i^{(t+1)}$ in at least one element. The selection in the DE is usually called as 'one-to-one' and it can be expressed mathematically as:

$$x_i^{(t+1)} = \begin{cases} w_i^{(t+1)} & \text{if } f(w_i^{(t+1)}) \leq f(x_i^{(t+1)}); \\ x_i^{(t+1)} & \text{otherwise:} \end{cases} \tag{3}$$

In the 'one-to-one' selection, both the trial and the target solutions compete for surviving into the next generation, where the better between both according to the fitness function goes to the next generation.

2.2. Numerical association rule mining

The ARM problem is defined formally as follows: Let us suppose a set of objects $O = \{o_1, \ldots, o_m\}$ and transaction database D are given, where each transaction T is a subset of objects $T \subset O$, and the variable m designates the number of objects. Then, an association rule is defined as an implication:

$$X \Rightarrow Y, \tag{4}$$

where $X \subset O$; $Y \subset O$, in $X \cap Y = \emptyset$. The quality of the association rule is evaluated using the following three measures [13]:

$$conf(X \Rightarrow Y) = \frac{n(X \cup Y)}{n(X)}, \tag{5}$$

$$supp(X \Rightarrow Y) = \frac{n(X \cup Y)}{N}, \tag{6}$$

$$incl(X \Rightarrow Y) = \frac{ante(X \Rightarrow Y) \wedge cons(X \Rightarrow Y)}{m}. \tag{7}$$

where $conf(X \Rightarrow Y) \geq C_{min}$ denotes confidence, $supp(X \Rightarrow Y) \geq S_{min}$ support and $incl(X \Rightarrow Y)$ inclusion of the particular association rule $X \Rightarrow Y$. The parameter N in Eq. (6) represents the number of transactions in the transaction database D, and $n(:)$ is the number of repetitions of the particular rule $X \Rightarrow Y$ within D. Additionally, C_{min} denotes minimum confidence and S_{min} minimum support, determining that only those association rules with confidence and support higher than C_{min} and S_{min} are taken into consideration, respectively.

Let us notice that functions $ante(X \Rightarrow Y)$ and $cons(X \Rightarrow Y)$ in Eq. (7) represent a set of objects belonging to either the antecedent or consequent, respectively. Mathematically, these functions are expressed as:

$$ante(X \Rightarrow Y) = \{o_p \mid p_j < Cp_i^{(t)} \wedge Th(Attr_{p_j}^{(t)}) = enabled\},$$
$$cons(X \Rightarrow Y) = \{o_p \mid p_j \geq Cp_i^{(t)} \wedge Th(Attr_{p_j}^{(t)}) = enabled\}.$$

The results of the optimization algorithm are stored into an archive of mined association rules. Opposed to the traditional algorithms for ARM, where each rule with support and confidence better than S_{min} and C_{min} is stored unconditionally, in NARM the archive is no longer generated uniformly. Instead, only those rules that outperform the current one according to the best fitness are stored into the archive. In this way, the size of the archive is reduced crucially.

Interestingly, each numerical attribute in NARM is determined by an interval of feasible values limited by its lower and upper bounds. The broader the interval, the more association rules can be mined. The narrower the interval, the more specific relations are discovered between attributes. Introducing intervals of feasible values has two effects on the optimization: To change the existing discrete search space to continuous, and to adapt these continuous intervals to suit the problem of interest better.

2.3. uARMSolver

The universal framework for ARM [18] (i.e., the uARMSolver) is used as a testbed for evaluating the proposed online NARM. It was developed in the C++ programming language and, therefore, its main advantages are: speed, modular design, and open-source coding. This miner covers all three steps needed for solving the NARM problems, i.e., preprocessing, optimization, and visualization. Due to its simplicity and efficiency, this is included as a software package suitable for running on most current Linux distributions (e.g., Fedora, RedHat, etc.).

The uARMSolver is an appropriate tool to deal with datasets in the UCI ML format [21]. These input datasets need to be preprocessed before entering into the optimization. Thus, a discretization of attributes is made in order to be ready for entering into a transaction database. Although more stochastic nature-inspired population-based algorithms are planned for inclusion into the framework, up to this point only two algorithms are available, i.e., DE [20] and Particle Swarm Optimization [22] (PSO). Both algorithms treat the ARM as an optimization problem. The visualization step supports the so-called Explanation AI (EAI), where the different visualization methods are applied in the sense of NARM.

It is necessary to modify the following three components of the original DE algorithm for the uARMSolver:

- representation of a solution,
- genotype-phenotype mapping,
- fitness function evaluation.

Let us notice that the maximum number of fitness function evaluations (*maxFEs*) is employed as a termination condition. The initial population is generated randomly, while the standard mutation strategies as proposed by Storn and Price [20] and one-to-one selection, are used in this algorithm. In the remainder of the section, the aforementioned components are described in detail. The section concludes with a description of the uARMSolver concept.

2.4. Representation of a solution

Solutions in a uARMSolver (i.e., association rules) are represented in the genotype space as real-valued vectors:

$$x_i = \{x_{i;1}, \ldots, x_{i;D}, x_{i;D+1}\}, \tag{8}$$

where sequences of elements $x_{i;j}$ for $j = 1, \ldots, D$ encode attributes of features, and the $x_{i;D+1}$ is the control point separating the antecedent from consequent part of the particular association rule. Interestingly, the length of each solution is variable, and calculated according to the following equation:

$$D = |Attr^{(C)}| \cdot N^{(C)} + |Attr^{(R)}| \cdot N^{(R)} + 1, \tag{9}$$

where $|Attr^{(C)}|$ and $|Attr^{(R)}|$ denote the size, while $N^{(C)}$ and $N^{(R)}$ the number of either categorical or numerical attributes, respectively. Obviously, one is added in the equation to consider the size of the additional element, i.e., the control point.

2.5. Genotype-phenotype mapping

The aim of the genotype-phenotype mapping is to decode the solution in the genotype space to its counterpart in the phenotype space. The solution in the genotype space is of variable size and encodes the attributes of two types: categorical and numerical. Each type is decoded from a fixed sequence of real values. More specifically, the categorical attributes are encoded as triples:

$$Attr_j^{(C)} = \langle p_j, attr_j, Th_j \rangle,$$

where

- p_j – is the order number in the permutation of attributes;
- $attr_j$ – is the value of the categorical attribute;
- Th_j – is the threshold determining the presence or absence of the attribute in the rule:

On the other hand, the numerical attributes are encoded in the genotype as quadruples:

$$Attr_j^{(R)} = \langle p_j, Lb_j, Ub_j, Th_j \rangle,$$

where

p_j –is the order number in the permutation of attributes;
$Lb_j; Ub_j$ –are the lower and upper bounds of the numerical attribute;
Th_j –is the threshold determining the presence or absence of the attribute in the rule:

Straightforward mathematical equations are employed for decoding the particular values of the phenotype [18]. This process is illustrated graphically in Fig. 1, from which it can be seen that three integer vectors are incorporated in the decoding process: (1) a permutation of attributes p, (2) a position pos, and (3) an attribute type. The permutation vector is obtained after sorting the elements p_j for $j = 1; \ldots ; n$ of the corresponding attribute tuples, the position vector denotes the starting position of the particular attribute in vector x_i, and the type vector highlights the type of each attribute (i.e., categorical 'C' or numerical 'R'). This vector is actually used for determining the size of the particular attribute.

2.6. Fitness function evaluation

After decoding an association rule $X \rightarrow Y$ from a genotype x_i, the quality of the solution needs to be evaluated in the phenotype space. The quality is evaluated using the fitness function expressed as follows:

$$f(x_i) = \frac{a \cdot supp(X \rightarrow Y) + b \cdot conf(X \rightarrow Y) + c \cdot incl(X \rightarrow Y)}{a + b + c}; \quad (10)$$

where $a; b$, and c denote weights, $supp(X \rightarrow Y); conf(X \rightarrow Y)$, and inclusion $incl(X \rightarrow Y)$ represent the support, the confidence, and the inclusion of the observed association rule, respectively.

2.7. The offline concept of the uARMSolver

The uARMSolver processes transactions in the transaction database offline (Fig. 2). This means that all transactions are processed sequentially, while the results of this processing are the mined association rules. Let us notice that only those rules that improve the value of fitness function are stored into an archive of association rules, quite the contrary to the traditional algorithms for ARM, like Apriori [13], where the number of stored association rules is regulated by the thresholds S_{min} and C_{min}.

Obviously, if more transactions emerge for importing to the transaction database, the new association rules need to be mined. In this case, the ARM process starts anew, wherein the consumption of computational resources (e.g., space and time) increase drastically. Therefore, the online NARM is proposed to continue the mining process with reusing the model discovered by the uARMSolver.

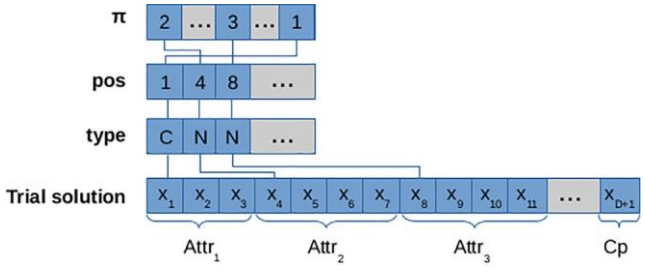

Fig. 1. Genotype-phenotype mapping.

Fig. 2. Offline ARM concept.

3. Online NARM solver

The uARMSolver using a DE algorithm serves as a tool allowing a detailed exploration of the NARM search space. The detailed exploration is possible because of an extended termination condition allowing a long-term stochastic searching, and the nature of the original DE algorithm guaranteeing that the DE search process does not trap into a local optima too quickly. As a result, the mined rules expose the huge qualities in the sense of the fitness function. Contrarily, the motivation behind developing the online NARM algorithm is to mine the high quality solutions in the short-term. This demands that the online NARM needs to be designed very carefully by incorporating various components allowing the issue. The incorporated components for addressing this issue are as follows:

- heuristic initialization,
- adaptation of the DE control parameters,
- mutation strategy,
- non-dominated solution selection,
- local search improvement heuristic,
- termination condition.

In the remainder of the section, the aforementioned components are illustrated in detail. The section terminates with a description of the concept of the online NARM solver.

3.1. Heuristic initialization

Initialization of a population can have a crucial impact on the results of the optimization for many types of problems [23]. This component is implemented in the onlineNARM twofold: heuristic and random. The heuristic initialization bases on knowledge explored by the uARMSolver. The explored knowledge, however, is accumulated within the population. Therefore, the original uARMSolver is modified, such that all the population individuals are saved into the so-called state population file, which can be restored later by the onlineNARM in the sense of initializing the population.

The heuristic population acts as follows: The individuals from the state population are sorted first, and then inserted at random positions in the initial population. The number of inserted state individuals is controlled by a parameter $ratio \in [0.0; 1.0]$. This means, when the $ratio = 0.0$, all the individuals from the state population are copied into the initial population, and vice versa when $ratio = 1.0$, the initial population is initialized randomly. In the case where the initial population size is less than the state one, the remainder of the places in the initial population are initialized randomly.

3.2. Adaptation of the DE control parameters

The Success History based Adaptive DE (SHADE) developed by Tanabe and Fukubaga in 2013 [24] is one of the more successful variants of the DE algorithm, whose main advantage lays in adaptation of the DE control parameters F and CR. The adaptation has a crucial impact on the results of optimization in the sense of the

time and the quality. Therefore, this mechanism is included into the onlineNARM as well.

Historical memories M_{CR} and M_F, with a size limited by parameter H, are used by the adapting mechanism. Initially, the elements of the history memory $M_{CR\ i}$and $M_{F\ i}$are initialized to 0.5, and modified according to the following equation [24]:

$$CR_i = N(M_{CR;r_i}; 0.1); \quad F_i = C(M_{F;r_i}; 0.1); \tag{11}$$

where $N(\mu; \sigma)$ denotes the randomly selected value drawn from the Gaussian distribution with mean μ and standard deviation σ; $C(\mu; \sigma)$ is the randomly selected value drawn from the Cauchy distribution with mean μ and standard deviation σ, while r_i is the randomly selected value drawn from the uniform distribution in the interval $[1; H]$.

The historical memories M_{CR} and M_F are modified according to the number of successfully changed individuals that are updated during the particular generation, and recorded in the so-called success history S_{CR} and S_F . The contents of these memories are modified as follows:

$$M_{CR;k}^{(t+1)} = \begin{cases} mean_{WA}(S_{CR}); & \text{if } S_{CR} \neq 0; \\ M_{CR;k}^{(t)}; & \text{otherwise;} \end{cases} \tag{12}$$

$$M_{F;k}^{(t+1)} = \begin{cases} mean_{WL}(S_F); & \text{if } S_F \neq 0; \\ M_{F;k}^{(t)}; & \text{otherwise;} \end{cases} \tag{13}$$

where functions $mean_{WA}$ and $mean_{WL}$ are expressed as:

$$mean_{WA}(S_{CR}) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR;k}; \tag{14}$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k}; \tag{15}$$

and $\Delta f_k = |f(u^{(t)}) - f(x^{(t)})|$. The weighted Lehmer mean $mean_{WL}(S_F)$ in Eq. (13) is expressed as follows:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F;k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F;k}}. \tag{16}$$

Finally, the parameter k represents a position in the memory where an update is performed. This position changes in each generation, starting from the value $k = 1$ to $k = H$, where the parameter is reset to its initial value.

3.3. Mutation strategy

Also, an employed mutation strategy 'current-to-pbest/1/bin' is borrowed from the SHADE algorithm. The main advantage of this strategy is the usage of an archive of best solutions found during the evolutionary process that collaborate actively by improving the current best solution.The mutation strategy is expressed as follows:

$$v_i^{(t)} = x_i^{(t)} + F_i^{(t)} \cdot (x_{pbest}^{(t)} - x_i^{(t)}) + F_i^{(t)} \cdot (x_{r_0}^{(t)} - x_{r_1}^{(t)}); \tag{17}$$

where $F_i^{(t)}$ denotes the scaling factor corresponding to the i-th vector, $x_{pbest}^{(t)}$ is a randomly selected value drawn from the top $NP \times p_i$ members in generation t, and r_1 is a randomly selected individual from the best ratio of the state population determined by the parameter p_i. Thus, p_i is calculated as follows:

$$p_i = rand[p_{min}; 0.2]; \tag{18}$$

where p_{min} is set such that the pbest individual can in the worst case be selected between two vectors, i.e., $p_{min} = 2/NP$. However, the archive in the online NARM presents a state population that is initialized with the best population borrowed from the uARMSolver and updated with the best individuals mined during the evolutionary search process.

3.4. Non-dominated solution selection

The onlineNARM uses a concept of domination [25] by selecting the best solutions. As it can be seen in Eq. (10), the fitness function is expressed as a linear combination of three objectives: a support, confidence and inclusion. These objectives are weighted by the coefficients $\alpha; \beta$, and γ, while the sum is maximized as a whole.

However, all three objectives are conflicting to each other. This means that the multi-objective optimization approach [25] should be used in this case. On the other hand, we are interested in the maximum value of the fitness function. Therefore, in our study, we left the fitness function the same as in the uARMSolver (thus comparison was kept between two miners), but changed the selection operation in DE (Eq. (3)). Actually, the selection operation in the original DE based on relation $f(w_i^{(t)}) \leq f(x_i^{(t)})$ is changed with the domination relation $f(w_i^{(t)}) \prec f(x_i^{(t)})$, where the sign $\prec$ denotes the domination relation. According to the domination relation, the trial solution $w_i^{(t)}$ is no worse than $x_i^{(t)}$ in all objectives, and is strictly better in at least one objective [25]. This dominance is expressed mathematically as $w_i^{(t)} \prec x_i^{(t)}$ (is better than).

3.5. Local search improvement heuristics

Evolutionary Algorithms (EAs) can be applied to a broad spectrum of problems, where little domain specific knowledge has already been explored. However, their performance can be improved drastically when the algorithm is hybridized with this knowledge. Practically, the domain specific knowledge can hybridize all components of the EAs. In our study, the proposed onlineNARM is hybridized with the local search heuristics, besides the heuristic initialization. According to Moscato [26], these kinds of algorithms are known under the name Memetic Algorithms (MA).

Three types of local search heuristics are developed for the onlineNARM, as follows:

- swap,
- move,
- amend.

All the aforementioned local search heuristics act on the genotype level, while their effects follow the so-called Lamarckian theory [27], according to which all changes in a parent organism acquired during its lifetime are passed to its offspring. The swap local search heuristic takes one attribute in antecedent and one in consequent randomly, and changes their corresponding elements between each other, except the permutation one. The permutation value ensures that the ordering does not change, and changes are made on the values of swapped attributes only. Let us mention that the different sizes of attributes (e.g., swapping between categorical and numerical attributes) also need to be taken into consideration by this operator. The move local search acts by selecting the i-th attribute in the antecedent and k-th in the consequent randomly and assigning the permutation value either as $p_i \prec p_k$ or vice versa (i.e., $p_k \prec p_i$), depending on the direction of the move operation. Let us emphasize that the sort operator in genotype-phenotype mapping moves the i-th attribute from antecedent to consequent and, oppositely, the j-th

attribute from consequent to antecedent. The last local search heuristic affects the values of the categorical attributes according to the following equation:

$$x^{\delta C\flat}_{i;pos\flat j]\flat 1} \leftarrow x^{\delta C\flat}_{i;pos\flat j]\flat 1} \flat sign - N\delta x^{\delta C\flat}_{i;pos\flat j]\flat 1}; 0{:}05\flat; \quad \delta 19\flat$$

while the numerical and real-valued values as follows:

$$\begin{cases} x^{\delta R\flat}_{i;pos\flat j]\flat 1} \leftarrow x^{\delta R\flat}_{i;pos\flat j]\flat 1} \flat sign - R\delta x^{\delta R\flat}_{i;pos\flat j]\flat 1}; 0{:}05\flat; & \text{if } U\delta 0{:}0; 1{:}0\flat < 0{:}5 \\ x^{\delta R\flat}_{i;pos\flat j]\flat 2} \leftarrow x_{i;pos\flat j]\flat 2} \flat sign - R\delta x_{i;pos\flat j]\flat 2}; 0{:}05\flat; & \text{otherwise;} \end{cases} \quad \delta 20\flat$$

where a function *sign* returns either −1 if $U\delta 0{:}0; 1{:}0\flat < 0{:}5\flat$ or + 1 otherwise, and $N\delta x^{\delta:\flat}_{i;pos\flat j]\flat k}; 0{:}05\flat$ for $k \leftarrow \delta 1j2\flat$ denotes the random number drawn from the Gaussian distribution with mean $x^{\delta:\flat}_{i;pos\flat j]\flat k}$ and standard deviation 0.05.

The local search heuristics are controlled using the parameter *LS_ratio*. When the local search heuristics are launched, they try to improve the target *i*-th vector in the population. These are active until an improvements is detected. In each local search phase, the neighborhood of the target vector is generated according to the following equation:

$$Neigh\delta i\flat \leftarrow \begin{cases} \begin{cases} swap & \text{if } U\delta 0{:}0; 1{:}0\flat < :5 \\ move & \text{otherwise} \end{cases} & \text{if } I: \\ \begin{cases} amend & \text{if } U\delta 0{:}0; 1{:}0\flat < :9 \\ n{=}a & \text{otherwise:} \end{cases} & \text{if } II: \end{cases} \quad \delta 21\flat$$

As can be seen from the aforementioned equation, either swap or move local search heuristics are selected with equal probability in the first phase. Then, the amend local search heuristic is applied, with the probability 0.9 on the changed hybrid vector.

3.6. Termination condition

Selecting the proper termination condition in an onlineNARM has a crucial impact on its performance. The maximum number of fitness function evaluations (*maxFEs*) remains the main condition for terminating the algorithm. However, too high a number of this parameter increases the time complexity, while a too low value of this parameter can terminate it too quickly, and, consequently, the better solution can be lost. Therefore, a balance needs to be found between the too fast convergence and too long exploration of the search space.

In this algorithm, also the second termination condition is applied, according to which the algorithm is terminated after the definite number of generations (i.e., convergence window *CW*), where no improvements are detected anymore.

3.7. The concept of the onlineNARM solver

The original transaction database is divided into: a *broker*, and *delta* (Fig. 3). The former represents the original transaction database in time t_0 that is optimized initially using the offline uARMSolver. Two additional results are produced after the optimization, i.e., the archive of association rules (also the *model*), and the *state* representing a population of solutions obtained after the last exploring of the best solution.

In time t_1, new transactions have arisen that are collected into the *delta* transaction database. The onlineNarm miner merges the *broker* and *delta* databases into the original transaction database in time t_2. Besides the original database, the online miner generates a new archive of association rules *model'* and a new *state'* population using the existing *model* and *state*. Thus, the onlineNARM solver does not optimize the transaction database anew, but continues the optimization in the state where the last miner ended. In this way, this consumes less computer resources, especially time.

Last but not least, the results of the onlineNARM can enter into the next cycle of the optimization process, where the original transaction database becomes the *broker* database, the *model'* changes to the *model*, and the *state'* to the *state* in the time t_0 of the next cycle. This means that the model is not reused only once, but can be reused more times.

4. Experiments and results

The goal of our experimental work was to show that the results of the uARMSolver, using the exhaustive evolutionary search, could be reached, or even improved with the results obtained by the onlineNARM, which employs the described mechanisms, allowing it to converge in significantly less time. As a result, the energy consumption is reduced drastically by decreasing the computational complexity, and, thus, justifies the foundations of the green AI.

Two types of the DE algorithm were used in the comparative study: (1) the original offline uARMSolver, and (2) the hybrid onlineNARM. The former serves for the initial exhaustive search space exploration, while the latter for the subsequent fast search space exploitation. Thus, the onlineNARM algorithm was hybridized with: heuristic initialization, adaptation of the DE control parameters, and using the 'current-to-pbest/1/bin' mutation strategy, the non-dominated selection and the local search heuristics. The algorithms used the parameters as illustrated in Table 1 during the

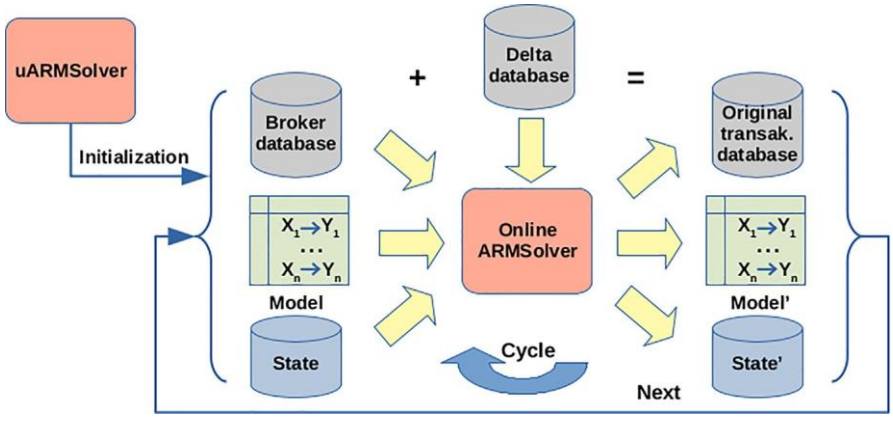

Fig. 3. Online ARM.

Table 1
Parameter setting of the DE algorithms in the tests.

Nr.	Parameter	Abbr.	uARMSolver	OnlineNARM
1	Population size	*NP*	100	100
2	Fitness func. evaluations	*maxFEs*	1,000,000	1,000
3	Convergence window	*CW*	200	2
4	Scale factor	*F*	0.5	adaptive
5	Crossover rate	*CR*	0.9	adaptive
8	Mutation strategy	DE	'rand/1/bin'	'cur.-to-pbest/1/bin'
7	Heuristic initialization	*ratio*	n/a	0.5
8	Local search probability	*LS_ratio*	n/a	0.1

runs. Each algorithm was run 30 times, and the best solutions according to the fitness function were observed in the analysis.

The population size presents a standard value as proposed by the DE community. The termination condition *maxFEs* highlights the particular algorithm's exploration/exploitation characteristics: While the offline miner emphasizes the exploration capability, the onlineNARM is more exploitative. Actually, these values represent a fair balance between too exhaustive exploration and too fast convergence to the local optima, as found during the extensive experiments. The DE control parameters *F* and *CR* were set either to the fixed values in the offline uARMSolver or to be adaptive in the onlineNARM. The original 'rand/1/bin' mutation strategy was employed in the first and the lSHADE 'current-to-pbest/1/bin' in the second case. The last two parameters (i.e., *ratio* and *LS_ratio*) are applied in the onlineNARM only.

Both algorithms were applied for solving the transaction databases from the UCI ML datasets [21]. As can be seen from Table 2, four datasets were selected, with different numbers of transactions, numbers of attributes, and their corresponding types. The purpose of the selection was to capture the various datasets according to these various characteristics.

In order to simulate a trend of incoming transactions, a reversible process was taken into consideration, which divides the original database into two partial databases according to the amount of transactions expressed by the percentage of the original ones. If, for instance, 10 % is selected, the higher 10 % of the transactions in the original database were attached to the broker, and the remainding 90 % to the delta database.

To assess the results obtained by the uARMSolver on the final databases with the results of the onlineNARM on the partial broker database substituted with the delta ones, a cosine similarity between two classifiers u and *v* (i.e., the vectors of the results) is used that is expressed by the Schwartz-Cauchy inequality [28], as follows:

$$\cos / \frac{ju - vj}{jjujj - jjvjj}; \qquad \text{ð22Þ}$$

where the term ju – vj denotes the inner product of two vectors, and the term jj:jj refers to the absolute value of the vector.

Table 2
Characteristics of the UCI ML datasets.

Nr.	Dataset	#tran.	#attr.	Type
1	Abalone	4,177	9	Mixed
2	Page blocks	5473	11	Numerical
2	Mushroom	8,125	22	Categorical
3	Adult	32,561	14	Mixed

The quality of the mined association rules were estimated according to Eq. (10) by both algorithms, thus, making the comparison possible. However, the better between trial and target solutions in one-to-one selection in the offline uARMSolver is selected according to the fitness function, while the non-dominated selection decides, which of the two solutions will survive in the onlineNARM.

4.1. Hardware configuration

All runs were made on a personal computer IBM Lenovo using the following configurations:

- Processor - AMD Ryzen 7–1700 3.90 GHz x 8,
- RAM - 16 GB,
- Operating system - Linux Ubuntu 22.04 Jammy Jellyfish (x86-64)19,
- Cinnamon Version - 5.2.7.

All versions of the tested algorithms were implemented within the Eclipse CDT Framework Version 2022–03.

4.2. Results

The following four reports are provided to justify the hypothesis set at the beginning of the section:

- analysis of the detailed results,
- quality analysis of the aggregate results,
- time complexity analysis of the aggregate results,
- analysis of the convergence speed.

In the remainder of the section the aforementioned tests are discussed in detail.

4.2.1. Analysis of the detailed results

In this experiment, we compared the performance of the onlineNARM by various partial broker databases with the results as obtained by the uARMSolver on the original transaction database. The partial broker databases were observed at milestones determined by 10 %, 25 %, 50 %, 75 %, and 90 % of the original database that are substituted with the corresponding delta databases to the original ones. The results of the offline uARMSolver, representing 100 % of the original database, and this one, were added to the study as well. In summary, six instances of the problem were considered. The goal of the onlineNARM was to get as near to the results of the uARMSolver as possible disregarding which broker database it was started from.

The detailed results obtained by the ARM on the Adult database are presented in Table 3. The table is arranged into columns representing the particular instances and rows representing various variables. The variables refer to the performance indicators and highlights the behaviour of the algorithm from different points of view. The meanings of the variables are presented in Table 4. Because the advanced analysis of the results according the quality and the time are performed in the remainder of the section, the focus, here, is on the detailed analysis of the performance indicator Rules. Indeed, the number of Rules increases from the value 611 achieved by the uARMSolver to the value 731 by the onlineNARM except its instance starting with 25 % of the transactions of the original database, where 600 association rules are mined only.

4.2.2. Quality analysis of the aggregate results

The goal of this experiment was to compare the results of the onlineNARM obtained by optimization of all four observed transaction databases according to a quality of solutions, and to show that

Table 3
Detailed results obtained by the Adult mining.

Variable	onlineNARM					uARMSolver
	10 %	25 %	50 %	75 %	90 %	100 %
Broker DB	3,257	8,141	16,280	24,420	29,304	32,561
Delta DB	29,304	24,420	16,281	8,141	3,257	0
Rules	713	600	703	662	731	611
Best fitness	0.799120	0.799498	0.799652	0.799693	0.799693	0.799693
At FEs	318	181	170	96	65	845
At time	149.62	93.17	104.33	71.76	73.59	479.60
LS calls	239	219	217	251	259	n/a
LS success	113	81	66	87	104	n/a
LR rate	0.4728	0.3699	0.3041	0.3466	0.4015	n/a
Total time	355.54	340.35	371.64	291.17	352.07	5,460.32

Table 4
Meaning of the variables.

Variable	Meaning
Broker DB	the size of the broker database per a particular instance
Delta DB	the size of the delta database per a particular instance
Rules	the number of rules mined
Best fitness	the best fitness function value
At FEs	the effective fitness evaluations needed for achieving the best fitness
At time	the effective time needed for achieving the best fitness
LS statistics	the number of calls, successful calls, and the rate of the successful calls
Total time	the total time needed for fulfilling the prescribed fitness function evaluations

the onlineNARM can achieve results near to the optimal (i.e., as achieved by the uARMSolver). In line with this, the detailed results obtained by both solvers in the first experiment are aggregated and examined closely.

The mentioned results according to the quality are presented in Table 5, from which it can be seen that the onlineNARM achieved results equal to the optimal at the higher instances of the broker databases (i.e., P 50 ~ %), while these are near to the optimal at the lower instances.

Two statistical test were conducted to show that the results achieved by the onlineNARM are not statistically significantly different from the results of the uARMSolver: (1) a 2-tailed pairwise *t*-test for significance level α ¼ 0:01, and (2) a cosine similarity test. The results of these statistical tests are illustrated in Table 6. As can be seen from the table, the results of the onlineNARM are not significantly different from the optimal results according to the 2-tailed parwise *t*-test for the significance level 0.01. Also the cosine similarity test indicates the irrelevant differences in the quality of mined association rules between both miners.

4.2.3. Time complexity analysis of the aggregate results

The analysis of time complexity addresses the results obtained by both miners according to the total time, and it is divided into two parts: In the first part, the comparison between the onlineNARM by mining the association rules on various instances of the broker databases is conducted without considering the initialization phase, while, in the second part, this phase is also taken into consideration.

Table 6
Results of the statistical tests.

Dataset	Parwise *t*-test		Cosine similarity	
	p-value	$p < 0{:}01$	cos /	S_C
Abalone	0.145997	No	0.999998	1
Page blocks	0.058864	No	0.999971	1
Mushroom	0.149438	No	0.999961	1
Adult	0.175609	No	1.000000	1

The results of the first part are presented in Table 7, from which it can be seen that the onlineNARM spends from 4.27 % by handling the Page blocks transaction database to 6.32 % for the Adult database of the total time spent by the uARMSolver, on average.

The situation is changed slightly, when the initialization phase, performed with the uARMSolverI (i.e., uARMSolver on the particular broker database) on the particular broker database, is taken into account (Table 8). The results in the table show that the total time of the initialization phase consumed by the uARMSolverI cannot be neglected. Indeed, the consumption of time is expanded from 36.47 % by mining the rules in the Mushroom database to 46.17 % by mining in the Abalone database, in average.

Table 7
Results of the UCI ML dataset mining according to the total processing time without considering the initialization phase.

Dataset	onlineNARM					uARMSolver
	10 %	25 %	50 %	75 %	90 %	100 %
Abalone	50.70	45.77	51.44	52.81	49.43	1047.63
Page blocks	56.13	61.86	74.10	73.55	69.47	1553.15
Mushroom	37.65	35.62	36.29	30.71	34.61	654.49
Adult	355.54	340.35	371.64	291.18	352.07	5,460.32
Average	125.00	120.90	133.37	112.06	126.39	2,178.89

Table 5
Results of the UCI ML dataset mining according to the best fitness.

Dataset	onlineNARM					uARMSo.
	10 %	25 %	50 %	75 %	90 %	100 %
Abalone	0.958733	0.959691	0.962883	0.962883	0.962883	0.962883
Page blocks	0.893438	0.902153	0.905320	0.911594	0.912446	0.912629
Mushroom	0.703177	0.703177	0.692967	0.688764	0.703177	0.703177
Adult	0.799120	0.799498	0.799652	0.799693	0.799693	0.799693
Average	0.838617	0.841129	0.840205	0.840733	0.844549	0.844595

Table 8
Results of the UCI ML dataset mining according to the total processing time with also considering the initialization phase.

Dataset	uARMSolverI + onlineNARM					uARMSolver
	10 %	25 %	50 %	75 %	90 %	100 %
Abalone	138.07	260.074	687.89	689.11	1,044.71	1,047.63
Page blocks	191.60	480.97	877.85	1,324.33	1,635.76	1,553.15
Mushroom	103.56	172.49	441.42	554.91	806.43	654.49
Adult	910.76	1,775.24	3,369.75	4,608.17	5,957.40	5,460.32
Average	335.99	672.19	1,344.23	1,794.13	2,361.08	2,178.89

However, when particular instances of the broker databases are taken into consideration, the following assertion holds: The lower the instance, the higher the total time savings. Obviously, the opposite also holds: The higher the instance, the smaller the savings. For instance, the savings amount to even more than 85 % for the instance containing the 10 % of transactions from the original transaction database, and 0 % saving for the instance containing the 90 % of transactions of the original transaction database.

4.2.4. Analysis of the convergence speed

This experiment was devoted to indicating how effective an evolutionary search process of the particular miners is. In line with this, the number of fitness function evaluations (FEs) was recorded, when the best fitness was detected. This number has a crucial influence on setting the convergence window *CW* that represents the second termination condition used in both miners.

The results of the experiment are illustrated in Table 9. From the table, it can be seen that the instances of broker databases containing the smaller number of transactions in the broker database demand more fitness functions evaluations to converge. When comparing the FEs achieved by the onlineNARM with those obtained by the offline uARMSolver, it can be concluded that the last one needed more evaluations to converge. This fact can be ascribed to the fact that the offline uARMSolver does not use any hybrid methods for improving its evolutionary search process.

Interestingly, the convergence window *CW* terminates the execution of the search process in the uARMSolver, while the onlineNARM is more sensitive on the *maxFEs* termination condition. Although the convergence window was set extremely low in the onlineNARM, the diversity of population ensured discovering the new promising solutions, and, thus, prevented it from terminating prematurely.

4.3. Discussion

The time complexity of the stochastic nature-inspired population-based algorithms, like EAs, is usually limited by using the parameter *maxFEs*. Here, the main issue is how to determine this parameter such that the corresponding EA is already able to discover the new best solutions, and to prevent the evolutionary search process from getting stuck in the local optima. There, we are confronted with the problem of exploration/exploitation [29]. This means too much selection strength, too fast loss of the population diversity. On the other hand, although the population diversity is a required condition for discovering the best solutions, it does not ensure that these can really be discovered. Due to a lack of theoretical studies in the domain, the reasonable setting of these values was determined experimentally. As a result, the proposed values, as found in our study, could be a good starting point for other potential researches.

Table 9
Results of the UCI ML dataset mining according to the FEs.

Dataset	onlineNARM					uARMSolver
	10 %	25 %	50 %	75 %	90 %	100 %
Abalone	561	194	100	100	171	288
Page blocks	228	196	321	183	98	718
Mushroom	268	530	97	227	54	9,236
Adult	318	181	170	96	65	845
Average	343.75	275.25	172.00	151.50	97.00	2,771.75

In order to decrease the value of the parameter *maxFEs*, generally, the proposed method applies the offline uARMSolver for an exhaustive evolutionary search at the beginning that consumes a reasonable number of the fitness function evaluations. The reasonable number was determined by the second termination condition in the form of the convergence window *CW* experimentally, such that this was suitable for all the problems under consideration in general. For instance, the uARMSolver needs almost 100 generations (i.e., *FEs* ¼ 9; 236 or 9; 236=100 ≈ 100 generations) to obtain the best solution by mining the rules within the Mushroom database, although the same algorithm found the optimal solutions earlier for the other databases (Table 9). Therefore, the selection of the parameter *CW* ¼ 200 seems reasonable in summary.

The onlineNARM is applied with reusing the model built by uARMSolver, whose time complexity can be controlled with the lower value of *maxFEs* ¼ 1; 000 and *CW* ¼ 2 due to the quicker convergence of the online algorithm. The results of this algorithm according to the total time are promising, especially for the instances with a lower number of transactions in the broker databases. For instance, the quality of the results by the onlineNARM are only 0:7 ~ % worse than by the uARMSolver in average (Table 5), although these were obtained by the former algorithm in even 84:24 ~ % less time in average (Table 8), when the smallest size of the broker database (i.e., 10 ~ %) is observed. Finally, the question should arise, how to define the value of the parameter *maxFEs* more precisely. Obviously, the answer to this question might be found in an analysis of the exploration/exploitation behavior of the onlineNARM algorithm on different problems.

Furthermore, the faster convergence of this algorithm is ensured by additional mechanisms, like: heuristic initialization, adaptation of the *F* and *CR* parameters, mutation strategy 'current-to-pbest/1/bin' using an archive of the previous best solutions, non-dominated selection, and local search heuristics. Heuristic initialization, for instance, depends on the setting of the parameter *ratio* that regulates if the initial solutions are generated heuristically or randomly. When the majority of the initial solutions are generated heuristically, the final result is usually a fast convergence to the local optima, while the majority of the random initial solutions cause a slow convergence. The evidence of the faster convergence is reflected in Table 9, where the onlineNARM used 87:59% of fitness function evaluations less than the uARMSolver (Table 9) in average.

Indeed, some drawbacks of the method have been discovered during the research study. These can be summarized in two facts as follows:

- identifying the distribution of transaction classes in the broker database with the original one,
- improper handling with the big data.

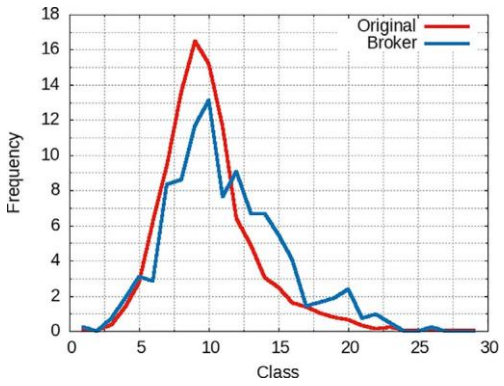

Fig. 4. The Figure presents a comparison of distributions of transaction classes between the original and broker datasets obtained from the Mushroom UCI ML dataset by dividing percentage of 10%. As can be seen from the Figure, there are 29 different classes into which transactions can be classified. In the original dataset, these classes are normally distributed, while the distribution is more scattered in the case of the broker dataset.

The present study assumes that the transactions in UCI ML datasets are aggregated in some history ordering. Therefore, the broker datasets are generated by dividing the original dataset into two parts according to the dividing percentage and regardless of the classes of the transactions being subjects of the dividing. This dividing, actually, does not consider that the distribution of the transactions in a broker database needs to be equal with those in the original (Fig. 4).

This distribution is not so important in datasets of variable size, where the transactions are added to the transaction database online. However, simulating the online process, like in our case, can prevent the onlineNARM from achieving the same results as the uARMSolver on each instance of the problem, especially, due to the different distribution of transaction classes.

As discovered during the experimental work, the uARMSolver could have a problem with handling big data, due to reading the whole data from the transaction database into a computer memory. Obviously, the problem could be solved easily by paging parts of the real from the virtual memory, where only parts of the whole database are presented in the computer memory at once. As a matter of fact, we did not observe this kind of problem, because we did not deal with the raw big data, in our study.

However, only one cycle of the onlineNARM was tested during our preliminary tests. Obviously, when the onlineNARM with reusing the model would repeat over more cycles, the savings in time (and indirectly in energy consumption) would be increased drastically.

5. Conclusion

The ARM is a hard ML problem in the sense of time and space complexity and consequently demands a lot of electrical energy for solving on digital computers. Therefore, it is classified in the class of Red AI, that represents a set of algorithms consuming too much electrical energy, and are indirectly unfriendly for the environment. This paper proposes a combination of the offline uARMSolver and onlineNARM, capable of drastic decreasing of the time complexity, and, consequently, also the electrical consumption by solving the problem. This method could be a potential candidate for classification in the Green AI class.

A lot of directions exists for improving the method: At first, the more complex transaction databases could be taken into consideration (e.g., the Hadoop environment for big data analytics). Next, the onlineNARM could be included into the uARMSolver framework as an independent process that could use the uARMSolver for initialization. Furthermore, the onlineNARM could be applied to data arising in data centers and cloud computing platforms, where huge electricity consumption takes place. Finally, additional improvements could be conducted, in order to reveal the processes of exploration and exploitation in more detail. The result of this study could help us to determine the termination condition in the onlineNARM more precisely.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Iztok Fister Jr. is grateful the Slovenian Research Agency for the financial support under Research Core Funding No. P2-0057. Iztok Fister thanks the Slovenian Research Agency for the financial support under Research Core Funding No. P2-0042 - Digital twin. Akemi Galvez and Andres Iglesias thank the financial support of the European project PDE-GIR of the European Union's Horizon 2020 research & innovation program (Marie Sklodowska-Curie action, grant agreement No 778035), and of the Spanish government project #PID2021-127073OB-I00 of the MCIN/AEI/10.13039 /501100011033/FEDER, EU.

References

[1] J. Patten, P.T. Keiser, D. Gysi, G. Menichetti, H. Mori, C.J. Donahue, X. Gan, I. do Valle, K. Geoghegan-Barek, M. Anantpadma, et al., Identification of druggable host targets needed for sars-cov-2 infection by combined pharmacological evaluation and cellular network directed prioritization both in vitro and in vivo.

[2] W. Leal Filho, T. Wall, S.A.R. Mucova, G.J. Nagy, A.-L. Balogun, J.M. Luetz, A.W. Ng, M. Kovaleva, F.M.S. Azam, F. Alves, et al., Deploying artificial intelligence for climate change adaptation, Technological Forecasting and Social Change 180 (2022).

[3] J. Cowls, A. Tsamados, M. Taddeo, L. Floridi, The ai gambit: leveraging artificial intelligence to combat climate change–opportunities, challenges, and recommendations, Ai & Society (2021) 1–25.

[4] S.-M. Cheong, K. Sankaran, H. Bastani, Artificial intelligence for climate change adaptation, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (2022) e1459.

[5] D. Rolnick, P.L. Donti, L.H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A.S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al., Tackling climate change with machine learning, ACM Computing Surveys (CSUR) 55 (2) (2022) 1–96.

[6] A. Nordgren, Artificial intelligence and climate change: ethical issues, Journal of Information, Communication and Ethics in Society.

[7] U. of Melbourne, A 'carbon law' offers pathway to halve emissions every decade, accessed August 3, 2022 (2022). www.sciencedaily.com/releases/2017/03/170323141338.htm.

[8] B. Ekholm, J. Rockström, Digital technology can cut global emissions by 15 %. Here's how, accessed August 3, 2022 (2022). https://www.weforum.org/agenda/2019/01/why-digitalization-is-the-key-to-exponential-climate-action/.

[9] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green ai, arXiv preprint arXiv:1907.10597.

[10] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in nlp, arXiv preprint arXiv:1906.02243.

[11] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108.

[12] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, E. Beigne, Spiking neural networks hardware implementations and challenges: A survey, ACM Journal on Emerging Technologies in Computing Systems (JETC) 15 (2) (2019) 1–35.

[13] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[14] M. Kaur, S. Kang, Market basket analysis: Identify the changing trends of market data using association rule mining, Procedia computer science 85 (2016) 78–85.
[15] E. Varol Altay, B. Alatas, Performance analysis of multi-objective artificial intelligence optimization algorithms in numerical association rule mining, Journal of Ambient Intelligence and Humanized, Computing 11 (8) (2020) 3449–3469.
[16] I.F. Jr, I. Fister, A brief overview of swarm intelligence-based algorithms for numerical association rule mining, arXiv preprint arXiv:2010.15524.
[17] I. Fister Jr., A. Iglesias, A. Galvez, I. Fister, Toward reusing the numerical association rule mining models, in: International Workshop on Soft Computing Models in Industrial and Environmental Applications, Springer, 2021, pp. 198–206.
[18] I. Fister, I.F. Jr., uarmsolver: A framework for association rule mining, CoRR abs/2010.10884. https://arxiv.org/abs/2010.10884.
[19] C. Darwin, On the Origin of Species, Harvard University Press, 1852.
[20] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. of Global Optimization 11 (1997) 341–359, https://doi.org/10.1023/A:1008202821328.
[21] D. Dua, C. Graff, UCI machine learning repository (2017). http://archive.ics.uci.edu/ml.
[22] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948 vol 4. doi:10.1109/ICNN.1995.488968.
[23] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, 2nd Edition., Springer Publishing Company, 2015, Incorporated.
[24] R. Tanabe, A. Fukunaga, Evaluating the performance of shade on cec 2013 benchmark problems, in: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, 2013, pp. 1952–1959, https://doi.org/10.1109/CEC.2013.6557798.
[25] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, in: Wiley Interscience Series in Systems and Optimization, Wiley, 2001.
[26] P. Moscato, Memetic Algorithms: A Short Introduction, McGraw-Hill Ltd., UK, GBR, 1999, pp. 219–234.
[27] R. Burkhardt, Jean-Baptiste Lamarck: Biological Visionary, University of Chicago Press, 2018, pp. 21–34. doi:10.7208/chicago/9780226570075.003.0002.
[28] S. Lipschutz, K. Kirkpatrick, M. Lipson, Schaum's Easy Outline of Linear Algebra, Schaum's Easy Outlines, McGraw-Hill Companies, 2002, Incorporated, https://books.google.si/books?id=pkESXAcIiCQC.
[29] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM Comput. Surv. 45 (3). doi:10.1145/2480741.2480752.