Performance of selected Nature-Inspired Metaheuristic Algorithms used for Extreme Learning Machine

Karol Struniawski[1][0000−0002−4574−2986], Ryszard Kozera[1,2][0000−0002−2907−8632] and Aleksandra Konopka[1][0000−0003−1730−5866]

[1] Institute of Information Technology,
Warsaw University of Life Sciences - SGGW,
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
{karol_struniawski, ryszard_kozera, aleksandra_konopka}@sggw.edu.pl
[2] School of Physics, Mathematics and Computing,
The University of Western Australia,
35 Stirling Highway, Crawley, WA 6009, Perth, Australia
ryszard.kozera@uwa.edu.au

Abstract. This work presents a research on Nature Inspired Metaheuristic Algorithms (MA) used as optimizers in training process of Machine Learning method called Extreme Learning Machine (ELM). We tested 19 MA optimizers measuring their performance directly on sample datasets. The impact of input parameters such as number of hidden layer units, optimization stopping conditions and population size on the accuracy results, training and prediction time is evaluated here. Significant differences in performance of applied methods and their parameters' values are detected. The most meaningful outcome of this paper shows that an increase of the number of MA iterations does not yield significant boost in accuracy with a huge increase in training time. Indeed a cap on number of MA iterations ranging from 1 to 5 is sufficient for analyzed machine learning tasks. In our research the best results are obtained for population size ranging between 50 and 100. Hybridized ELM outperforms classical implementation of ELM as higher accuracy is reached for the same number of neurons.

Keywords: Computational Optimization · Metaheuristic Algorithms · Bio-inspired computing · Extreme Learning Machine · Machine Learning

1 Introduction

Mathematical optimization algorithms play a vital role in many contemporary technology applications such as e.g. GPS or IT banking sector tools. In addition, the optimization driven behavior is also prevalent within all living organisms commonly relying on it while e.g. hunting or trying to move more efficiently. In fact, searching for the efficiency in nature can be a matter of life and death. Among all the latter is physically demonstrated by the reproduction capabilities.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

Thus, organisms that perform life activities more efficiently are better adopted to the environment and are more likely to pass these abilities to their offsprings by genes according to the Darwin's Theory [7].

Scientists attempt to describe "optimized" activities of organisms in terms of mathematical modeling [27] commonly called Metaheuristic Algorithms (MA). The combination of bio-inspired optimization algorithms with machine learning models may improve their performance [30]. Here one of such approaches is called Extreme Learning Machine (ELM) - the Machine Learning method with growing popularity since its formulation in 2004 [14].

Recently, a hybrid MA-ELM that combines ELM with Metaheuristic Algorithms (MA) is proposed and evaluated in practical applications. In doing so, Chia et al. [8] used particle swarm (PSO), moth-flame (MFO) and whale optimization algorithm (WOA). In other practical related context, Wu et al. [30] applied genetic algorithm (GA), ant colony optimization (ACO), cuckoo search algorithm (CSA) and flower pollination algorithm (FPA). The above research demonstrates superiority of hybridized ELM over the regular one. Nevertheless, most of the works in this topic deal with the practical applications of these methods. There is a shortage in literature on comprehensive comparison of metaheuristic algorithms used in ELMs. In this paper we evaluate hybrid ELM on MNIST handwritten and Wine Quality White datasets [9] for different MA. The comparison analysis for a separate set of parameters to investigate their impact on attained accuracy and registered computational time is also performed for each examined algorithm. The experiment is carried out on a single machine in MATLAB R2021b, Ryzen 9 3900X CPU, 64GB RAM, GTX 1660TI GPU.

2 Extreme Learning Machine

Extreme Learning Machine (ELM) is a dense feed-forward neural network classifier and regressor introduced by Huang et al. in 2004 [14]. The network's topology consists of input layer, a single hidden-layer and an output layer of neurons. The numbers of selected neurons in input and output layer depends on the task characteristics. The number of hidden layer units requires an empirical determination as a consequence of the theoretical method scarcity permitting to determine upfront its optimal numbers controlling the topology of the ELM.

2.1 Classification

Input data regarding supervised classification task with N observations can be described as pairs of values $\{(x_i, t_i)\}_{i=1}^{N}$, where x_i is i-th vector of d features and t_i is i-th label of class to which selected x_i belongs. Here, $t_i = 0, \dots, M-1$, where M is the amount of distinctive classes in the classification task in question. Note here that for multiclass classification (when object belongs to more than one class) t_i is a vector. Based on the latter, matrix $X = (x_1, x_2, \dots, x_N) \in \mathbb{M}_{d\times N}(\mathbb{R})$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

is formed, where $x_i \in \mathbb{R}^d$ with vector $T = \{t_i\}_{i=1}^N$:

$$X = \begin{bmatrix} x_{11} \dots x_{1N} \\ \vdots \ \ddots \ \vdots \\ x_{d1} \dots x_{dN} \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}.$$

The ELM input layer comprises of d neurons and its output layer consists of units number equal to M. As an output of the network, the corresponding N values $\{y_i\}_{i=1}^N$ are calculated forming the matrix $Y = (y_1, y_2, \dots, y_N) \in \mathbb{M}_{N\times M}(\mathbb{R})$, where $y_i \in \mathbb{R}^M$. The recognition of a given input x_i is performed based on extracting the maximal value of y_i observed on the p-th index which assigns x_i to the p-th class. Thus, matrix Y is actually reformatted as N values $\{y_i\}_{i=1}^N$, where $y_i = [0, \dots, \overset{p}{1}, \dots, 0]$. Subsequently, one has to properly format T in order to facilitate comparison with Y. In the next step $1 - of - K$ scheme is applied to the vector T - see [6]. Such procedure is designed to reformat t_i as $\{t_{ij}\}_{j=0}^M$ that yields all values set to zero except one element at s-th index that in turn is set to one. Consequently, t_i can be written as $t_i = [0, \dots, \overset{s}{1}, \dots, 0]$, where s-th element indicates that i-th input vector of X affiliates to the s-th class. Correct classification is observed if and only if $s = p$ for a given input vector x_i.

Let L be a number of neurons in hidden layer that is chosen a priori. Weights between input and hidden layer determine the matrix $W \in \mathbb{M}_{d\times L}(\mathbb{R})$, where w_{ij} represent the weights associated with the connection of i-th input layer neuron with j-th in hidden layer (see left equation (1)). Bias connections are represented by a vector $b = \{b_i\}_{i=1}^N$. In learning process of ELM coefficients of W and b are computed using uniform distribution function $U(-1, 1)$. The outputs of hidden layer neurons are stored in matrix $H \in \mathbb{M}_{N\times L}(\mathbb{R})$ (see right equation (1)):

$$W = \begin{bmatrix} w_{11} \dots w_{1L} \\ \vdots \ \ddots \ \vdots \\ w_{d1} \dots w_{dL} \end{bmatrix}, H = \begin{bmatrix} f(\sum_{i=1}^d x_{i1}w_{i1} + b_1) & \dots & f(\sum_{i=1}^d x_{i1}w_{iL} + b_1) \\ \vdots & \ddots & \vdots \\ f(\sum_{i=1}^d x_{iN}w_{i1} + b_N) & \dots & f \sum_{i=1}^d x_{iN}w_{iL} + b_N) \end{bmatrix}. \tag{1}$$

The activation function $f : \mathbb{R} \to \mathbb{R}$ represents in our investigation a sigmoid function $f(x) = f_\alpha(x) = \frac{1}{1+e^{-\alpha x}}$, with $\alpha = 1$. The weights β between hidden and output layer can be computed upon solving the following equation $Y = H\beta$. The system cannot be directly solved since H with probability equal to 1 is irreversible and $||H\beta - Y|| = 0$ (see Huang et al. [14]). We estimate β as a minimizer of mean residual square error:

$$\hat{\beta} = \underset{\beta}{argmin} \, \|H\beta - T\|^2 = H^\dagger T, \tag{2}$$

where $H^\dagger$ defines a Moore-Penrose generalized inverse of H [26]. The Pseudo-inverse of matrix $H^\dagger$ is uniquely determined and in the case of a non-singular matrix H it coincides with an ordinary inverse i.e. $H^\dagger = H^{-1}$. The matrix $H^\dagger$ gives solution $\hat{\beta}$ so that $H\hat{\beta}$ is close to Y in terms of mean square error (MSE).

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

Assigning random values to weights and bias between input and hidden ELM layer makes the network not susceptible to overtraining. Most importantly, the computed solution $\hat{\beta}$ is a global minimizer of (2). The latter contrasts with Multi-Layer Perceptron (MLP) supervised training procedure. Indeed Backpropagation Algorithm finds generically only a local minimizer of the given network's loss function that measures how well the neural network classifies the training data [12]. In addition, the optimal value of $\hat{\beta}$ is found here upon performing a non-iterative procedure in (2). The learning speed of ELM can be thousands times faster than other methods like MLP (see [15]).

3 Genetic Extreme Learning Machine

The original concept of ELM relies on selecting weights between input and hidden layer together with bias values as randomly generated. This principle has a remarkable advantage in terms of computational efficiency [15]. Still such randomness in weights generation in ELM can lead to the unstable performance [4]. The idea here is to somehow estimate weights and bias values in order to maximize the accuracy and stability of the model. A possible remedy to this problem is to combine the Genetic Algorithm (GA) (see [13]) with ELM to form the so-called hybridized Genetic Extreme Learning Machine (GELM). GAs are created as a computational representation of Darwinian evolution theories to search for the optimal solution of global non-linear optimization task by simulating the process of biological natural selection concept [16]. Our hope is that reflecting the natural processes of selection, crossover and mutation the fittest individuals are selected for reproduction that will provide better offspring in terms of improving an appropriate fitness evaluation function [5].

4 Nature-Inspired Metaheuristic Algorithms

In general, the constrained optimization problem can be formulated in terms of minimizing some objective function: $minimize f(x)$ with $x = (x_1, \ldots, x_n)$ admitted to fulfill either some equality(ies) and/or inequality(ies) [31]. All modern nature-inspired algorithms are called Metaheuristic Algorithms [17]. Up to now there is no commonly accepted definition of MA, but one can outline the following selected principles of MA adopted in the literature (see [27, 31, 24]): a strategy that the main aim is to guide the search process avoiding the disadvantages of iterative improvement allowing the local search to escape from local optima; starting to find solutions in more intelligent way than just providing random initial solutions; dealing with randomness in an biased form incorporating search experience (in a form of memory) to guide the search; in the simulation stage considered as a set of assumptions about the natural environment.

The search strategies of different MA are highly dependent on the philosophy of the metaheuristic itself. In this paper, as a comparison of the MA applied in ELM learning process, we use methods simulating behaviors of living organisms in terms of the following optimization processes: Artificial Ecosystem-based

Optimization (AEO) [35], Artificial Hummingbird Algorithm (AHA) [34], Artificial Rabbits Optimization (ARO) [29], African Vultures Optimization Algorithm (AVOA) [2], Coyote Optimization Algorithm (COA) [25], Dandelion Optimizer (DO) [32], Fast Cuckoo Search (FCS) [23], Gorilla Troops Optimizer (GTO) [3], Grey Wolf Optimizer (GWO) [20], Hybrid Grey Wolf and Cuckoo Search Optimization Algorithm (GWO-CS) [11], Improved Grey Wolf Optimizer (I-GWO) [21], Leader Harris Hawks Optimization (LHHO) [22], Mountain Gazelle Optimizer (MGO) [1], Manta Ray Foraging Optimization (MRFO) [36], Northern Goshawk Optimization (NGO) [10], Pelican Optimization Algorithm (POA) [28], Hybrid Particle Swarm Optimization and Gravitational Search Algorithm (PSOGSA) [19], Sea-horse Optimizer (SHO) [33] and lastly Salp Swarm Algorithm (SSA) [18].

5 Experiments and Results

The metaheuristic algorithms (briefly outlined in the previous section) used in this work have common prerequisites. In particular, from now on, the term MA directly refers to the algorithms exclusively used in this paper (see Section 4).

MA define a concept of population as a set S of S_n candidate solutions, where s_i, $i = 1, \dots, S_n$ is a solution vector called also an individual and implement the concept of intelligent iterative ransacking search space taking as an input dimension of the vector $S_d = dim(s_i)$, number of population S_n and constraints applied to s_i. A termination condition for the algorithm and appropriate fitness function must be determined. As MA fall into iterative methods, in k-th iteration the set S^k called generation is produced with $s_i^k \in S^k$ representing generation's individual, where $k = 1, \dots, k_n$. The output of MA $s^{min} = s_i^{k_n}$ yields a minimal value of a given fitness function in the last generation of the algorithm.

To integrate MA with ELM we first need to specify input parameters for MA. Analogously to GELM our aim is to evaluate optimal values of weights between input and hidden layer including bias. In fact, the output of the MA is a vector $s^{min} \in \mathbb{R}^{S_d}$, where $S_d = dN + N$. As a consequence we can reformat s^{min} properly constructing W and b:

$$W = \begin{bmatrix} s_{11}^{min} & \dots & s_{1N}^{min} \\ \vdots & \ddots & \vdots \\ s_{d1}^{min} & \dots & s_{dN}^{min} \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} s_{dN+1}^{min} \\ \vdots \\ s_{dN+N}^{min} \end{bmatrix}.$$

Vector s^{min} forms the final, optimal value of $W^{s^{min}}$ and $b^{s^{min}}$. Fitness function g is prepared based on the response of the ELM network represented as Y_i^k for a given s_i^k (forming W_i^k and b_i^k) compared to the expected results T, $g(X, W_i^k, b_i^k, T) = \frac{1}{N} \sum_{j=1}^{N} (Y_{ij}^k - T_j)^2$, where $Y_i^k = H\beta$, $\beta = H^\dagger T$ and $H = f(X^T W_i^k + b_i^k)$ (see also (1)). The inequality constraints $-1 < s_{ij} < 1$ (with $j \in [1, \dots S_d]$ for each $i \in [1, \dots, S_n]$) enforce $s_i \in [-1; 1]^{S_d} \subseteq \mathbb{R}^{S_d}$. The impact of parameters S_n and the termination condition on the algorithm performance is investigated in this research. The admitted values for S_n are equal to 50, 100 or 200. It is noteworthy that lowering the vales of S_n led to unstable results, while higher S_n values resulted in impractically long evaluation times for our experiment. Two different approaches of selecting stopping conditions of MA are here considered. *First*, the stopping flag is activated once one of two conditions is fulfilled. More specifically, the upper limit on k iterations is a priori set (here $k_n = 10000$). In conjunction with the latter, the optimization procedure terminates once the following a posteriori condition is met $\left|g(X, W_i^k, b_i^k, T) - g(X, W_i^{k+1}, b_i^{k+1}, T)\right| < \varepsilon$ holding for longer than 200 iterations (here $\varepsilon = 0.0001$). In further presentation of calculation results the first variant of stopping condition is marked as *"Limit 0"*. *Second*, the impact of fixing ad hock an upper bound k on number of iterations is also analyzed here for $k_n = 1$, $k_n = 5$ and $k_n = 50$ that can be recognized in further considerations as *"Limit 1"*, *"Limit 5"* and *"Limit 50"*, respectively.

Another parameter taken also here into consideration is the number of neurons L in hidden layer of ELM. At this point one should mention a dilemma of evaluating results applying testing and training sets once MA is used for optimizing W and b. A core principle of the Machine Learning (ML) is to examine results returned by a selected method on data that cannot be used for training process. To enforce the latter the data is usually a priori divided into training and testing sets or alternatively one resorts to a cross-validation method [12]. Cross-validation is an iterative method that uses different portions of data to test and to train a model applying randomness. Thus, matrices W and b are optimized upon using MA on training data exclusively and cannot be specified as optimal on testing set. A similar approach should be adopted for β evaluation while computing weights between hidden and output layer of ELM. It is implicitly assumed here that dependencies for both training and testing sets are similar. Therefore the optimized W and b based on training set can equally successfully operate on testing set. The case of unbalanced number of observations obtained on training and testing sets deserves a short note. Indeed, should the latter occurs, the matrices W and b generated by MA on training set cannot be directly applied to estimate Y on testing set. In ML there exists an implicit assumption that the testing set should be essentially smaller than a training one. Typically, the proportion of observations abides from 9:1 to 7:3 ratio. Consequently, s^{min} is too large to be properly re-formatted to W and b which can still act on testing set. Assuming testing set contains N^{test} observations we solve this problem by taking $k = N^{test} \times d$ first elements of s^{min} transforming them into matrix $W^{s^{min}}_{N^{test} \times d}$ and last N^{test} elements of s^{min} creating vector $b^{s^{min}}$:

$$W^{s^{min}} = \begin{bmatrix} s_1^{min} & \dots & s_N^{min} \\ \vdots & \ddots & \vdots \\ s_{k-N}^{min} & \dots & s_k^{min} \end{bmatrix} \quad \text{and} \quad b^{s^{min}} = \begin{bmatrix} s_{S_d - N^{test}}^{min} \\ \vdots \\ s_{S_d}^{min} \end{bmatrix}.$$

The entire calculation process is presented in the flowchart (see Fig. 5).

First, we evaluate the model in question for a different number of neurons L in hidden layer, population size S_n and MA termination condition taken as *"Limit 0"*. Unfortunately, even for $S_n = 50$ and $L = 100$ computation time for

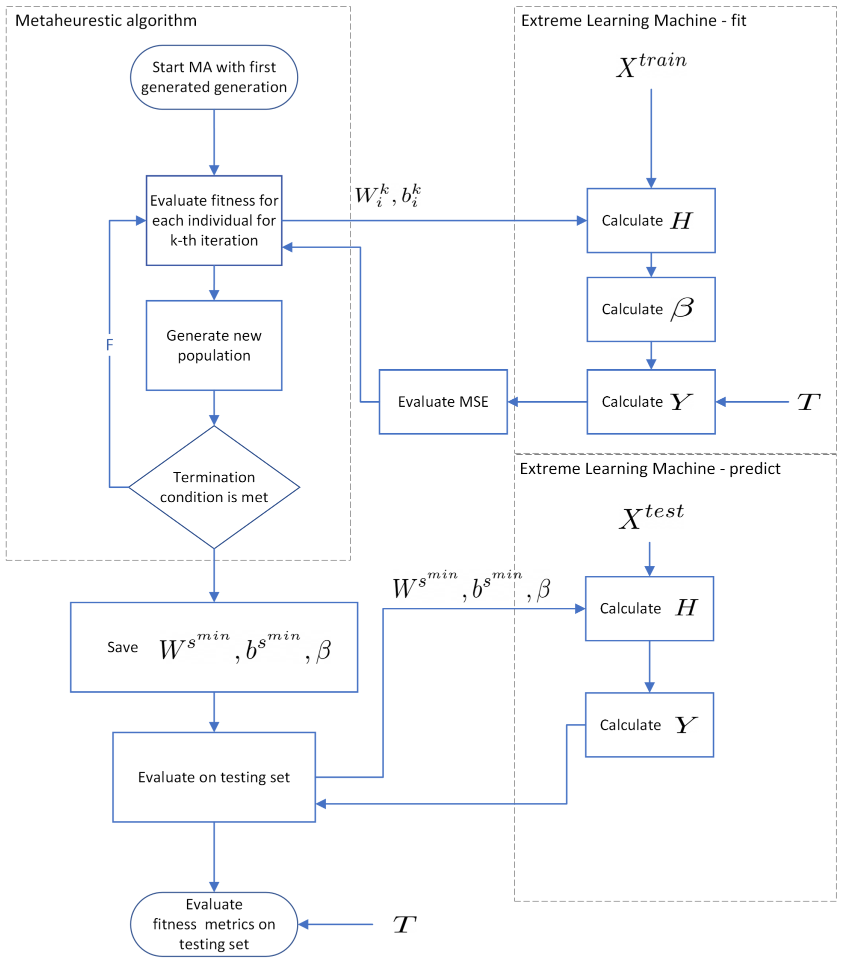
Metaheurestic algorithm
Start MA with first generated generation
Evaluate fitness for each individual for k-th iteration
Generate new population
Termination condition is met
F
W_i^k, b_i^k
Extreme Learning Machine - fit
X^{train}
Calculate H
Calculate β
Calculate Y
T
Evaluate MSE
Extreme Learning Machine - predict
X^{test}
$W^{s^{min}}, b^{s^{min}}, \beta$
Calculate H
Calculate Y
Save $W^{s^{min}}, b^{s^{min}}, \beta$
Evaluate on testing set
Evaluate fitness metrics on testing set
T

Fig. 1. Flowchart of MA-ELM training and testing process.

most of the methods exceeded a few hours. Then, a full comparison to the other Limits is impossible. Therefore we discarded *"Limit 0"* calculations and leave it for a future investigation. For our research two exemplary datasets are used. The first set is called MNIST handwritten digits. The dataset contains 60000 training and 10000 testing samples that are handwritten digits saved as greyscale images of size 28×28 pixels. Thus, input vectors' size is 784 after flatten operation is applied to the original image transforming image to the row by row vector. The dataset is a typical example of classification task where accuracy of the classifier is evaluated. The second set is called Wine Quality White which is composed of features describing chemical and physical parameters of white wine i.e. fixed acidity, volatile acidity, pH etc. Our task is to assign to a given wine sample the quality measure ranging from 0 to 10. Here classification performance is measured with MSE as we recognize differences between inappropriate class assignments i.e. attaching a wine which is truly categorized as 0 to class 9 is a graver mistake than assigning this wine to class 1. The dataset does not contain separate training and testing subsets and because of that to obtain the most meaningful results a 20% cross-validation method is applied that is repeated 50 times to properly estimate a statistically significant value of MSE and to discard randomness influence on the final results.

ACC [%]	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
AEO	78.31	79.04	82.05	85.19	85.06	86.14	84.95	78.74	88.89	88.00
AHA	77.08	81.02	83.80	84.17	86.77	85.22	87.89	88.29	85.29	89.61
ARO	73.74	83.43	81.06	83.17	83.76	85.96	88.27	89.20	88.08	88.02
AVOA	68.50	82.11	84.06	81.66	84.46	85.78	86.55	88.85	89.84	89.60
COA	72.75	78.67	83.83	84.62	86.36	86.04	86.28	89.36	88.47	90.09
DO	73.48	81.41	83.61	80.31	87.55	81.95	89.22	86.20	87.54	85.98
FCS	72.89	80.43	80.94	86.15	85.77	87.17	86.76	89.25	88.02	88.80
GTO	78.54	81.73	78.35	84.67	75.38	88.85	84.99	80.26	83.8	84.41
GWO	69.42	74.79	81.01	83.28	83.95	88.52	87.58	85.05	88.65	90.08
GWO-CS	46.43	51.76	63.67	68.60	62.92	70.41	68.98	70.54	73.66	71.76
I-GWO	71.36	79.02	82.68	80.52	86.51	86.38	88.23	86.47	84.77	85.78
LHHO	72.48	81.02	80.57	87.39	84.42	86.93	85.53	87.43	87.88	87.84
MGO	71.59	78.65	78.06	80.13	86.27	86.93	86.93	88.36	88.42	88.92
MRFO	73.52	81.21	86.47	83.36	82.47	88.8	88.34	89.16	90.01	89.25
NGO	69.89	81.43	84.35	84.63	84.9	86.17	87.09	88.54	89.72	90.70
POA	69.81	81.06	83.16	84.73	85.89	87.31	87.83	88.09	87.36	89.60
PSOGSA	74.91	78.97	83.23	84.25	85.34	86.70	86.36	84.90	86.89	90.19
SHO	72.87	81.92	81.92	84.57	86.44	84.13	83.34	89.73	89.49	85.07
SSA	74.92	80.61	81.97	82.41	82.41	86.66	88.8	87.78	87.21	87.77

Table 1. Accuracy of ELM with selected MA applied for a given number of neurons in hidden layer, population size 50 and limit of iterations equal to 1 used directly as a classifier on MNIST handwritten digits dataset.

For MNIST similarly to the ELM (see Tab. 7) for MA-ELM we obtain better results upon increasing number of neurons (see Tab. 1). Then it was decided to fix L to the value of 1000 in further calculations of MA-ELM. In contrast, for Wine Quality White dataset the best results are observed for $L = 100$ (see Tab. 2), so this value will be fixed analyzing the dataset. Fitting process (see Tab. 3) for MNIST, $L = 1000$, $S_n = 50$ and selected limit of iterations combined with a method in question may take from a few seconds to some hours. The fastest methods turned out to be AVOA, DO, PSOGSA, SHO and SSA. In particular, the last one yields the most prominent results with 8s, 284s and 2784s fit time for limit k_n set to 1, 5 and 50, respectively. One can also notice that raising the number of iterations more or less linearly increases the resulting computation time. Differences in fitting time between various methods testify

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

MSE	100	200	300	400	500	600	700	800	900	1000
AEO	0.641	0.647	0.659	0.679	0.692	0.699	0.725	0.734	0.755	0.775
AHA	0.634	0.645	0.661	0.669	0.685	0.701	0.718	0.731	0.759	0.768
ARO	0.648	0.650	0.654	0.670	0.684	0.697	0.716	0.739	0.747	0.765
AVOA	0.645	0.647	0.663	0.675	0.688	0.706	0.722	0.739	0.753	0.776
COA	0.639	0.647	0.659	0.680	0.690	0.712	0.721	0.738	0.751	0.771
DO	0.653	0.652	0.661	0.681	0.692	0.708	0.725	0.743	0.754	0.768
FCS	0.647	0.650	0.659	0.677	0.694	0.707	0.727	0.738	0.762	0.761
GTO	0.650	0.650	0.659	0.661	0.674	0.719	0.712	0.723	0.740	0.756
GWO	0.648	0.647	0.661	0.678	0.690	0.709	0.719	0.741	0.757	0.774
GWO-CS	0.625	0.628	0.641	0.642	0.671	0.672	0.726	1.105	0.722	1.056
I-GWO	0.644	0.651	0.668	0.672	0.693	0.709	0.723	0.747	0.749	0.770
LHHO	0.630	0.638	0.660	0.676	0.695	0.681	0.722	0.727	0.758	0.794
MGO	0.635	0.637	0.656	0.668	0.695	0.694	0.721	0.726	0.726	0.752
MRFO	0.641	0.642	0.656	0.657	0.674	0.711	0.720	0.730	0.751	0.765
NGO	0.647	0.654	0.664	0.674	0.686	0.708	0.726	0.739	0.751	0.771
POA	0.656	0.651	0.664	0.676	0.691	0.704	0.724	0.745	0.756	0.773
PSOGSA	0.647	0.648	0.662	0.677	0.695	0.703	0.732	0.737	0.758	0.772
SHO	0.651	0.649	0.659	0.677	0.696	0.710	0.723	0.738	0.756	0.772
SSA	0.647	0.652	0.663	0.675	0.696	0.708	0.720	0.742	0.755	0.767

Table 2. MSE of Extreme Learning Machine with selected Metaheuristic Algorithms applied for a given number of neurons in hidden layer, population size 50 and limit of iterations equal to 1 used directly as a classifier on Wine Quality White dataset.

their practical applicability i.e. MGO needs fourfold more time for *"Limit 1"* to achieve comparable results with SSA. It should be emphasized here that there is no correlation between more computational time involved versus achieving better results. Indeed, a GWO method surpasses in terms of ACC the other methods that still need twice longer time to be executed. In previous section MA are defined as methods that tend to create the new generations with individuals characterized by lower fitness function value. Simultaneously, as we stated the low MSE value for a given individual being MA solution cannot be directly recognized as better in terms of accuracy upon applying on a testing set, because of the fact that in training and testing different subsets of dataset are used. Surprisingly, for many of the methods increasing number of iterations does not improve ACC (see Tab. 3). For most of them we observe a slight increase of ACC between *"Limit 1"* and *"Limit 5"*. The decrease of ACC between *"Limit 5"* and *"Limit 50"* was not expected. For some of the methods the lowest ACC is obtained for *"Limit 50"*. The classifier performance on Wine Quality White confirms results obtained on MNIST. For the majority of methods we do not observe significant changes of MSE. Setting k_n to higher value even increases MSE in the case of AHA, COA, GWO-CS, LHHO, MGO and PGOGSA. For the remaining methods change of k_n from 1 to 5 results in a slight decrease of MSE. Methods AEO, AVOA, DO, FCS, GWO, MGO, MRFO and NGO can be characterized by decreasing MSE once k_n changes from 1 to 5. In terms of

	MNIST						Wine Quality White					
	Fit time [s]			ACC [%]			Fit time [s]			MSE		
Method / k_n	1	5	50	1	5	50	1	5	50	1	5	50
AEO	237	571	5697	88.00	86.79	84.95	0.6	2.4	21.6	0.641	0.635	0.641
AHA	155	312	3989	89.61	88.98	84.62	0.3	1.7	10.7	0.634	0.646	0.631
ARO	156	309	4028	88.02	88.15	81.87	0.3	1.4	11.0	0.648	0.637	0.636
AVOA	78	258	4050	89.60	84.94	77.36	0.1	1.0	10.9	0.645	0.629	0.634
COA	278	692	7274	90.09	82.7	88.83	0.6	3.2	26.6	0.639	0.643	0.624
DO	77	273	2666	85.98	88.46	86.9	0.1	1.3	11.5	0.653	0.649	0.650
FCS	233	564	5041	88.80	86.51	77.86	0.5	2.5	20.9	0.647	0.647	0.651
GTO	103	567	5062	84.41	89.99	89.89	0.5	2.6	20.5	0.650	0.637	0.633
GWO	250	277	2705	90.08	88.57	87.78	0.2	1.1	11.8	0.648	0.648	0.653
GWO-CS	270	311	3046	71.76	72.99	87.88	0.2	1.3	13.7	0.625	0.643	0.630
I-GWO	445	608	5466	85.78	88.99	86.67	0.6	3.2	24.7	0.644	0.643	0.635
LHHO	240	896	8879	87.84	89.87	76.34	0.6	3.6	36.7	0.630	0.635	0.623
MGO	236	1218	11486	88.92	90.55	89.89	1.0	5.6	52.6	0.635	0.633	0.640
MRFO	232	566	5069	89.25	88.14	83.28	0.5	2.9	21.3	0.641	0.642	0.643
NGO	141	566	5038	90.70	91.45	87.42	0.5	2.6	21.5	0.647	0.645	0.647
POA	212	556	4974	89.60	81.25	86.32	0.5	2.8	20.6	0.656	0.647	0.638
PSOGSA	82	507	4800	90.19	85.8	90.81	0.5	2.6	24.6	0.647	0.655	0.649
SHO	22	471	4152	85.07	87.37	72.69	0.5	2.4	17.0	0.651	0.644	0.634
SSA	8	284	2784	87.77	87.71	88.90	0.2	1.5	12.8	0.647	0.647	0.643

Table 3. Fit time and accuracy of Extreme Learning Machine with selected Metaheuristic Algorithms applied for $L = 1000$ neurons in hidden layer, population size $S_n = 50$ and $k_n = 1$, 5 or 50 used directly as a classifier on MNIST handwritten digits. For $L = 100$, $S_n = 50$, $k_n = 1$, 5 or 50 on Wine Quality White dataset.

fitting time we observe more or less a linear growth of computational time when k_n is enlarged. In Tab. 4 we tested a different population size S_n. It shows that increasing S_n expands the fitting time, but to a lesser extent with exceptions like SSA method that needs almost 14× computation time for $S_n = 100$ as compared to $S_n = 50$. Such tendency is similar for SHO method. Fitting time between $S_n = 100$ and $S_n = 200$ for most of the methods expand twice. In terms of accuracy, we do not observe a significant increase when population size is enlarged. The methods that are beneficial to this increase are DO, FCS, GTO, GWO-CS, I-GWO, LHHO, MGO, MRFO and SSO. Noticeably, for DO, FCS, I-GWO, MGO and MRFO the highest accuracy is registered for $S_n = 100$ and the lowest for $S_n = 200$. The observed dependencies on MNIST are even more visible for Wine Quality dataset. The lowest MSE for all methods is obtained for $S_n = 50$ and increases substantially when $S_n = 100$ or $S_n = 200$ is used.
For standard ELM classifier applied on MNIST handwritten dataset the highest accuracy 91.41% is generated for 4000 and 5000 neurons in hidden layer (see Tab. 7). In comparison, for MA-ELM the highest ACC of 91.45% is reached for NGO metaheuristic algorithm, 1000 neurons, limit of 5 iterations, population size 50 and 91.43% ACC for MRFO with 1000 neurons, limit of 1 iteration and

	MNIST						**Wine Quality White**					
	Fit time [s]			**ACC [%]**			**Fit time [s]**			**MSE**		
Method / S_n	50	100	200	50	100	200	50	100	200	50	100	200
AEO	237	331	675	88.00	86.47	88.50	0.6	30	57	0.641	0.743	0.741
AHA	155	211	428	89.61	86.46	86.07	0.3	19	36	0.634	0.804	0.788
ARO	156	208	435	88.02	84.53	86.26	0.3	19	36	0.648	0.732	0.771
AVOA	78	104	215	89.60	88.13	88.87	0.1	9.	18	0.645	0.745	0.712
COA	278	369	785	90.09	89.42	88.25	0.6	35	68	0.639	0.717	0.744
DO	77	102	209	85.98	89.93	87.88	0.1	9	17	0.653	0.704	0.734
FCS	233	313	654	88.80	90.06	88.67	0.5	29	57	0.647	0.737	0.711
GTO	103	321	649	84.41	80.56	89.26	0.5	28	54	0.650	0.804	0.716
GWO	250	115	231	90.08	88.99	88.85	0.2	11	21	0.648	0.750	0.731
GWO-CS	270	142	315	71.76	84.23	89.93	0.2	16	34	0.625	0.733	0.759
I-GWO	445	345	721	85.78	89.06	87.23	0.6	33	65	0.644	0.747	0.736
LHHO	240	422	845	87.84	88.39	89.68	0.6	39	71	0.630	0.715	0.749
MGO	236	657	1574	88.92	89.59	87.32	1.0	69	153	0.635	0.731	0.729
MRFO	232	318	673	89.25	91.43	82.92	0.5	30	60	0.641	0.774	0.788
NGO	141	312	651	90.70	86.34	88.46	0.5	29	56	0.647	0.737	0.749
POA	212	306	628	89.60	87.06	89.96	0.5	28	53	0.656	0.732	0.717
PSOGSA	82	296	996	90.19	87.65	88.91	0.5	42	140	0.647	0.664	0.747
SHO	22	286	593	85.07	88.07	89.39	0.5	28	56	0.651	0.717	0.743
SSA	8	110	229	87.77	90.69	88.18	0.2	10	20	0.647	0.687	0.746

Table 4. Fit time and accuracy of Extreme Learning Machine with selected Metaheuristic Algorithms applied for 1000 neurons in hidden layer, limit of iterations equal to 1 and a different population size $S_n = 50$, 100 or 200 used directly as a classifier on MNIST handwritten digits dataset and 100 neurons in hidden layer, limit of iterations equal to 1 and a different population size $S_n = 50$, 100 or 200 used directly as a classifier on Wine Quality White dataset.

Method	S_n	L	k_n	**Fit MSE**	**Fit time [s]**	**Prediction time [s]**	**ACC [%]**
NGO	50	1000	5	0.059	566	0.074	91.45
MRFO	100	1000	1	0.057	318	0.082	91.43
GTO	50	900	5	0.058	471	0.081	90.88
GTO	200	900	5	0.057	2928	0.069	90.85
SSA	50	900	5	0.062	236	0.065	90.81
PSOGSA	50	1000	50	0.060	4800	0.068	90.81

Table 5. The 6 highest ACC for Extreme Learning Machine with selected Metaheuristic Algorithms used directly as a classifier on MNIST handwritten digits dataset.

population size 100 (see Tab. 5). Here we obtained comparable results of ELM and MA-ELM but for a different number of neurons. Training time, that is stated as a fit time for MA-ELM, is a lot longer than in case of typical ELM. Note here that in many practical application cases of ML a short prediction time is crucial. The time is extended when net is composed of more hidden layer units. Focusing on prediction time we should compare nets with 1000 hidden layer neurons, then

Method	S_n	L	k_n	**Fit MSE**	**Fit time [s]**	**Prediction time [s]**	**MSE**
LHHO	50	100	50	0.417	36.706	0.008	0.623
COA	50	100	50	0.414	26.659	0.008	0.624
GWO-CS	50	100	1	0.425	0.285	0.005	0.625
GWO-CS	50	200	1	0.398	0.561	0.010	0.628
AVOA	50	100	5	0.421	1.073	0.008	0.629

Table 6. The 5 lowest MSE for Extreme Learning Machine with selected Metaheuristic Algorithms used directly as a classifier on Wine Quality White dataset.

Neurons	**Fit Time [s]**	**Prediction Time [s]**	**ACC [%]**
1000	3	0.069	88.69
2000	7	0.134	90.57
3000	14	0.256	91.26
4000	26	0.268	91.41
5000	45	0.412	91.41
6000	71	0.511	91.30
7000	115	0.527	90.58
8000	165	0.707	90.83

Table 7. Extreme Learning Machine used directly as a classifier on MNIST handwritten digits dataset results.

Neurons	**Fit Time [s]**	**Prediction Time [s]**	**MSE**
100	0.004	0.0003	0.646
200	0.008	0.0014	0.652
300	0.021	0.0032	0.660
400	0.028	0.0034	0.677
500	0.025	0.0025	0.691
600	0.040	0.0033	0.710
700	0.052	0.0037	0.721
800	0.053	0.0041	0.744
900	0.105	0.0061	0.757
1000	0.130	0.0084	0.770

Table 8. Extreme Learning Machine used directly as a classifier on Wine Quality White results.

MA-ELM achieve ACC higher by 3pp (percentage points) over EML. Prediction time is highly dependent on L, then there is no difference of prediction time between MA-ELM and ELM. When we compare a similar ACC from the both methods (for MA-ELM $L = 1000$ and ELM $L = 4000$) prediction time for 1000 is 6 times shorter which can be very beneficial in models that require short classification time. Summing up the results obtained for the Wine Quality with ELM classifier applied leads to a rise of MSE when number of neurons in hidden layer increases (see Tab. 8). The lowest MSE=0.646 is generated for $L = 100$ with training time of the net equal to $0.004s$ and prediction coinciding with $0.0003s$.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

According to Tab. 6 which presents the top 5 lowest MSE across all exploited parameters' values of MA-ELM the best results are produced for $S_n = 50$, $L = 100$ and $k_n = 50$ for LHHO and COA methods. The lowest observed MSE=0.623 for MA-ELM is a better result than using core ELM method for which MSE=0.646 is detected. Both classifiers for this dataset have comparable prediction time as the best results are reached for the same value of L.

6 Conclusions

In this paper the concept of hybridized ELM with MA is introduced. Subsequently, the influence of the parameters' value selection on final results is examined. More precisely, the impact on the results of the number of neurons in hidden layer of ELM, the size of the population and the stopping conditions for MA are investigated. Based on this research we conclude that higher accuracy of the hybridized ELM can be detected even for lower number of neurons in hidden layer than in typical ELM. The latter leads to a significant fall in prediction time of the model. Surprisingly, the best results assessing MA termination condition of MA are registered as a hard limit of 5 iterations for MNIST handwritten and of 50 iterations for Wine Quality White dataset. Unfortunately, we were not able to examine termination condition of MA as a limit of 10000 iterations or changes of fitness less than $\varepsilon = 0.0001$ because of the computational complexity involved. This aspect should be further investigated. The population sizes examined in our study were set to 50, 100, and 200, as lower values led to unstable results, and higher values resulted in impractically long evaluation times for our experiment. In total 19 MA methods are tested and across all SSA and MRFO stand out for their high accuracy combined with shorter training times compared to other methods. Notwithstanding, one ought to emphasize that there is no method that yields excellent results on both datasets. It is worth noting that there is no direct correlation between increased computational time and improved results. The selection of the appropriate MA algorithm for a particular task should be based on comprehensive evaluation. However, in our work, we observed that certain algorithms exhibit a high computational complexity without a significant improvement in classification accuracy. Therefore, MGO, AEO, COA, and LHHO may not be suitable for hybridized ELM and can be discarded from further consideration.

References

1. Abdollahzadeh, B., Gharehchopogh, F.S., Khodadadi, N., Mirjalili, S.: Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. Adv. Eng. Softw. **174**, 103282 (2022). https://doi.org/10.1016/j.advengsoft.2022.103282
2. Abdollahzadeh, B., Gharehchopogh, F.S., Mirjalili, S.: African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems. Comput. Ind. Eng. **158**, 107408 (2021). https://doi.org/10.1016/j.cie.2021.107408

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

3. Abdollahzadeh, B., Soleimanian Gharehchopogh, F., Mirjalili, S.: Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. Int. J. Intell. Syst. **36**(10), 5887–5958 (2021). https://doi.org/10.1002/int.22535
4. Albadr, M.A.A., Tiun, S., AL-Dhief, F.T., Sammour, M.A.M.: Spoken language identification based on the enhanced self-adjusting extreme learning machine approach. PLOS One **13**(4), 1–27 (2018). https://doi.org/10.1371/journal.pone.0194770
5. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann Publishers Inc. (1998)
6. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
7. Brooks, C.M.: The nature of life and the nature of death. Journal of UOEH **5**(2), 133–145 (1996)
8. Chia, M.Y., Huang, Y.F., Koo, C.H.: Swarm-based optimization as stochastic training strategy for estimation of reference evapotranspiration using extreme learning machine. Agric. Water Manag. **243**, 106447 (2021). https://doi.org/10.1016/j.agwat.2020.106447
9. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. **47**(4), 547–553 (2009). https://doi.org/10.1016/j.dss.2009.05.016
10. Dehghani, M., Hubalovsky, S., Trojovsky, P.: Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems. IEEE Access **9**, 162059–162080 (2021). https://doi.org/10.1109/ACCESS.2021.3133286
11. Gupta, A.: Hybrid grey wolf and cuckoo search optimization algorithm (2023), https://www.mathworks.com/matlabcentral/fileexchange/69392-hybrid-grey-wolf-and-cuckoo-search-optimization-algorithm
12. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, 1st edn. (1995)
13. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
14. Huang, G.B., Zhu, Q.Y., Siew, C.: Extreme learning machine: A new learning scheme of feedforward neural networks. In: IEEE Int. Conf. Neural Netw. vol. 2, pp. 985 – 990 (2004). https://doi.org/10.1109/IJCNN.2004.1380068
15. Li, H.T., Chou, C.Y., Chen, Y.T., Wang, S.H., Wu, A.Y.: Robust and lightweight ensemble extreme learning machine engine based on eigenspace domain for compressed learning. IEEE TCAS-I **66**(12), 4699–4712 (2019). https://doi.org/10.1109/TCSI.2019.2940642
16. Liu, W.C., Chung, C.E.: Enhancing the predicting accuracy of the water stage using a physical-based model and an artificial neural network-genetic algorithm in a river system. Water **6**(6), 1642–1661 (2014). https://doi.org/10.3390/w6061642
17. lreau, M., Potvin, J.Y.: Handbook of Metaheuristics. Springer Publishing Company, Incorporated, 2nd edn. (2010)
18. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
19. Mirjalili, S., Hashim, S.Z.M.: A new hybrid PSOGSA algorithm for function optimization. In: ICCASM 2010. pp. 374–377 (2010). https://doi.org/10.1109/ICCIA.2010.6141614

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38

20. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
21. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S.: An improved grey wolf optimizer for solving engineering problems. Expert Syst. Appl. **166**, 113917 (2021). https://doi.org/10.1016/j.eswa.2020.113917
22. Naik, M.K., Panda, R., Wunnava, A., Jena, B., Abraham, A.: A leader harris hawks optimization for 2-D masi entropy-based multilevel image thresholding. Multimed. Tools. Appl. **80**(28), 35543–35583 (2021). https://doi.org/10.1007/s11042-020-10467-7
23. Naik, M.K., Swain, M., Panda, R., Abraham, A.: Novel square error minimization-based multilevel thresholding method for COVID-19 x-ray image analysis using fast cuckoo search. Int. J. Image Graph. (2022). https://doi.org/10.1142/s0219467824500049
24. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography. Ann. Oper. Res. **63**, 511–623 (1996)
25. Pierezan, J., Dos Santos Coelho, L.: Coyote optimization algorithm: A new metaheuristic for global optimization problems. In: IEEE-CEC 2018. pp. 1–8 (2018). https://doi.org/10.1109/CEC.2018.8477769
26. Rao, C.R., Mitra, S.K.: Generalized Inverse of Matrices and its Applications. John Wiley & Sons (1971)
27. Stützle, T.: Local search algorithms for combinatorial problems - analysis, improvements, and new applications. In: DISKI (1999)
28. Trojovsky, P., Dehghani, M.: Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. Sensors **22**(3) (2022). https://doi.org/10.3390/s22030855
29. Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., Zhao, W.: Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. **114**, 105082 (2022). https://doi.org/10.1016/j.engappai.2022.105082
30. Wu, L., Zhou, H., Ma, X., Fan, J., Zhang, F.: Daily reference evapotranspiration prediction based on hybridized extreme learning machine model with bio-inspired optimization algorithms: Application in contrasting climates of China. J. Hydrol. **577**, 123960 (2019). https://doi.org/10.1016/j.jhydrol.2019.123960
31. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press (2010)
32. Zhao, S., Zhang, T., Ma, S., Chen, M.: Dandelion optimizer: A nature-inspired metaheuristic algorithm for engineering applications. Eng. Appl. Artif. Intell. **114**, 105075 (2022). https://doi.org/10.1016/j.engappai.2022.105075
33. Zhao, S., Zhang, T., Ma, S., Wang, M.: Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems. Appl. Intell. (2022). https://doi.org/10.1007/s10489-022-03994-3
34. Zhao, W., Wang, L., Mirjalili, S.: Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications. Comput. Methods Appl. Mech. Eng. **388**(1), 114194 (2022). https://doi.org/10.1016/j.cma.2021.114194
35. Zhao, W., Wang, L., Zhang, Z.: Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural. Comput. Appl. **32**, 1–43 (2020). https://doi.org/10.1007/s00521-019-04452-x
36. Zhao, W., Zhang, Z., Wang, L.: Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. **87**, 103300 (2020). https://doi.org/10.1016/j.engappai.2019.103300

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_38