ARTIFICIAL NEURAL NETWORKS REGRESSION ON ENSEMBLE STRATEGIES IN DIFFERENTIAL EVOLUTION

Iztok Fister Jr.[1],Ponnuthurai Nagaratnam Suganthan [2],
Damjan Strnad [1], Janez Brest [1],Iztok Fister [1]

[1]University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
Email: iztok.fister1@um.si
[2]Nanyang Technological University
School of Electrical and Electronic Engineering
639798, Singapore

Abstract: *Nature always serves as an inspiration to scientists for developing new algorithms to solve challenging real-world problems. Building mathematical models of the brain activity has led to the emergence of artificial neural networks (ANN) especially useful for solving problems, such as classification and regression. On the other hand, evolutionary algorithms (EAs) inspired by Darwinian natural evolution have been successfully applied to solve optimization, modelling and simulation problems. Differential evolution (DE) is a well-known EA that possesses a multitude of strategies for generating an offspring solution, where the best strategy is not known in advance. In this paper, a link between ANN and DE has been established, where the best DE strategy in each generation is identified using the ANN regression. The task of the regression is to predict the best strategy from an ensemble of DE strategies for each trial solution. The experiments on a suite of ten well-known functions showed the future potential in developing this idea.*

Keywords: *artificial neural networks, differential evolution, regression, ensemble strategies*

1 Introduction

Scientists in various research areas that are confronted with solving tough real-world problems have always searched for an inspiration in the nature. Nature not only poses the questions, but also provides answers to these. However, this task is entrusted to scientists. In computer sciences, two nature-inspired mechanisms influenced their development, as follows: human brains [22], and Darwinian theory of struggle for survivor [4]. The former inspiration from the nature has led to the emergence of artificial neural networks (ANN), while the latter to evolutionary algorithms (EAs). In this paper, ANN is used to solve a regression problem in differential evolution.

ANN is based on findings of neuroscience that mental activity consists primarily of electrochemical activity in network of brain cells called neurons [22]. In line with this hypothesis, the first mathematical model of neurons devised by McCulloch and Pitts [17]. According to this model, a neuron "fires", when a linear combination of inputs exceeds some threshold. This threshold could be either hard or soft and implements a linear classifier useful to predict an input value as a finite set of values (e.g., color can be either {*red, blue, green*}). Latter, the ANN were used also for learning problems called regression in which an appropriate numerical output value is searched for.

On the other hand, DE has become one of the most prominent EAs for solving tough real-world optimization problems. This population-based method was introduced by Storn and Price in 1995 [23]. Individuals in the population representing the solution of the problem to be solved are in a form of real-valued vectors that are subjected to the operators of crossover and mutation. Thus, a population of trial vectors is generated that compete with their parents for survival. As a result, when a fitness of the trial vector is better than the fitness of its parent laying in the same index position in the population, the parent is replaced by the trial (offspring) solution.

In order to further improve the DE algorithm, its development has been conducted in several ways. For example, adapting and self-adapting DEs assume that the parameters as set at the start of the search process may not be appropriate in the latter phases and vice versa. Therefore, these parameters are changed during the

run either according to a feedback of the search that determines the direction or magnitude of the change to the strategy parameter, by encoding into a representation of solution and undergo operations of crossover and mutation. For example, the more prominent adaptive and self-adaptive DE algorithms are jDE [3], SaDE [19], etc. On the other hand, another kind of DE algorithms tries to improve the results of the original DE algorithm by combining the various ensemble of parameters and mutation DE strategies [16, 24, 25]. A complete survey of DE methods can be found in [5, 26].

This paper combines the ANN and DE algorithms in order to predict the best DE mutation strategy for each individual in the population. Similar to parameter setting, where the best parameter values depend on the progress of the search process, also selection of the proper DE mutation strategy is not known in advance. Although the DE mutation strategy can be determined adaptively, like in [15], we propose the ANN regression for predicting the best DE mutation strategy for each individual from an ensemble of DE strategies. In this way, various DE strategies are applied for each individual, where the best value of element obtained by all strategies in the DE ensemble are used to predict the best value of the corresponding trial vector. Although the similar idea was already presented by Fister et al. in [7], here the emphasis on the ANN regression was exposed. On the other hand, the test suite was broadened and the comparative study according to various dimensions of the test functions were performed.

The structure of the remainder of the paper is as follows. In Section 2, fundamentals of ANN and DE algorithms are presented in detail. Section 3 proposes a new DE algorithm with ANN regression (nnDE). The experiments and the results are presented in Section 4. The paper concludes with a review of the performed work and a closer look at the future work.

2 Background

2.1 Artificial neural networks

Origin of ANN development comes from biology. ANN can be defined as a very simplified model of a human brain. Similar as the human brain, artificial neworks consists of many artificial neurons. Natural neuron receives signal via synapses, which lay on dendrites or membranes of the neuron [21]. When the received input signals are greater than the some threshold value, the neuron output is activated, and emits a signal throughout an axon. This signal might be either sent to another synapse, or activate other neurons in the neighborhood [9]. On basis of this natural process in the human brain, the artificial neuron is modelled by McCulloch and Pitts [17]. It also consists of many inputs (similar as synapse in human brain) that are multiplied by weigths and then computed by a mathematical function determining the activation level of the neuron [9]. This is activated, when the activation level exceeds the some predefined threshold value. Another function in this process computes output signal of the neuron. Weights that play a very important role in this process can be adjusted during the run adaptively. In general, this process is also called learning. In fact, there are three important types of learning according to the types of feedback that is available to learn from [22]:

- supervised learning [10]: the learner observes some example input-output pairs and learns a function that maps from input to output (e.g., backpropagation),
- unsupervised learning [13]: the learner learns patterns in the input even though no explicit feedback is supplied (e.g., self-organizing map), and
- reinforcement learning [1]: the learner learns from a series of reinforcements (i.e., rewards or punishments).

This article uses the supervised learning, i.e., backpropagation, where artificial neurons are organized in layers that send signal forward (feed-forward neural networks) and then, errors are propagated backwards. The network receives inputs via neurons in the input layer, and the output of the network is given by the neurons in the output layer. Additionaly, there may be one or more intermediate hidden layers [9]. The task of the backpropagation is to minimize the error. At the end of each learning step, weights are adjusted such that the error is reduced. Readers are also invited to look at a graphical representation of neural network in Fig. 1.

ANNs are widely used for solving the classification, regression, and prediction problems. Some of the well-suited application domains, of the ANNs are, as follows: industry and business [27], data mining [2], civil engineering [12], and a fire analysis of steel frames [11].

2.2 Differential evolution

Differential evolution (DE) belongs to the class of evolutionary algorithms and is appropriate for solving continous as well as discrete optimization problems. DE was introduced by Storn and Price in 1995 [23] and since then many DE variants have been proposed. Some of the the most efficient DE variants are [3, 18, 19, 20, 14].

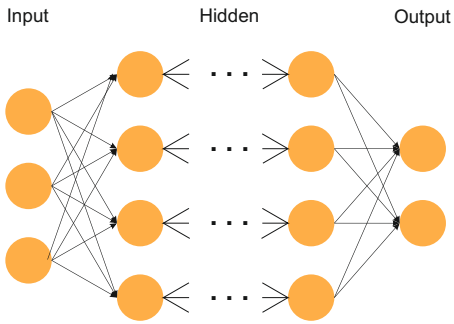


Figure 1: Graphical representation of simple artificial neural network

The original DE algorithm is represented by real-valued vectors and is population-based. The DE supports operators, such as mutation, crossover, and selection.

In the basic mutation, two solutions are randomly selected and their scaled difference is added to the third solution, as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r0}^{(t)} + F \cdot (\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}), \quad \text{for } i = 1 \dots NP, \tag{1}$$

where $F \in [0.1, 1.0]$ denotes the scaling factor that scales the rate of modification, while $r0$, $r1$, $r2$ are randomly selected values in the interval $1 \dots NP$ and NP represents the population size. Note that the proposed interval of values for parameter F was enforced in the DE community, although Price and Storn proposed the slightly different interval, i.e., $F \in [0.0, 2.0]$.

DE employs a binomial (bin) or exponential (exp) crossover. The trial vector is built from parameter values copied from either the mutant vector generated by Eqn (1) or parent at the same index position i. Mathematically, this crossover can be expressed as follows:

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

where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution. The condition $j = j_{rand}$ ensures that the trial vector differs from the original solution $\mathbf{x}_i^{(t)}$ in at least one element.

Mathematically, the selection can be expressed as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise}. \end{cases} \tag{3}$$

Crossover and mutation can be performed in several ways in differential evolution. Therefore, a specific notation was introduced to describe the varieties of these methods (also strategies), in general. For example, 'rand/1/bin' denotes that the base vector is randomly selected, 1 vector difference is added to it, and the number of modified parameters in the trial/offspring vector follows a binomial distribution. The other standard DE strategies are illustrated in Table 1. These strategies also form an ensemble of DE strategies (ES).

Table 1: An ensemble of DE-strategies

Nr.	Strategy	Mutation Expression	Crossover
1	Best/1/Exp	$x_{i,j}^{(t+1)} = best_j^{(t)} + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	exponential
2	Rand/1/Exp	$x_{i,j}^{(t+1)} = x_{r1,j}^{(t)} + F \cdot (x_{r2,j}^{(t)} - x_{r3,j}^{(t)})$	exponential
3	RandToBest/1/Exp	$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (best_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	exponential
4	Best/2/Exp	$x_{i,j}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$	exponential
5	Rand/2/Exp	$x_{i,j}^{(t+1)} = x_{r1,i}^{(t)} + F \cdot (x_{r2,i}^{(t)} + x_{r3,i}^{(t)} - x_{r4,i}^{(t)} - x_{r5,i}^{(t)})$	exponential
6	Best/1/Bin	$x_{i,j}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} - x_{r2,i}^{(t)})$	binomial
7	Rand/1/Bin	$x_{i,j}^{(t+1)} = x_{r1,j}^{(t)} + F \cdot (x_{r2,j}^{(t)} - x_{r3,j}^{(t)})$	binomial
8	RandToBest/1/Bin	$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (best_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	binomial
9	Best/2/Bin	$x_{i,j}^{(t+1)} = best_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$	binomial
10	Rand/2/Bin	$x_{i,j}^{(t+1)} = x_{r1,i}^{(t)} + F \cdot (x_{r2,i}^{(t)} + x_{r3,i}^{(t)} - x_{r4,i}^{(t)} - x_{r5,i}^{(t)})$	binomial

3 The proposed algorithm

The proposed ANN regression on ensemble of DE strategies algorithm (nnDE) (pseudo-code in Algorithm 1) modifies the generation of the trial vector in the original DE algorithm. In place of using a specific DE strategy,

a set T of trial solutions based on the current vector $\mathbf{x}_i$ are generated, where the DE strategy is selected randomly in each generation step. As a result, the NP-randomly generated trial vectors are obtained in the test set using the target vector $\mathbf{x}_{best}$. The validation set V consists of the vector obtained by applying the 'rand/1/bin' strategy on the current vector $\mathbf{x}_i$. Finally, the regression vector $\mathbf{r}_i$ is obtained by predicting each element of the validation vector V from the same positioned elements of NP-vectors in the test set T using the ANN regression method.

Algorithm 1 The proposed nnDE algorithm

```
Initialize the DE population x_i = (x_i1, ..., x_iD) for i = 1 ... NP
repeat
  for i = 1 to i ≤ NP
    Create test set T on vector x_i using random strategy from ES and target vector x_best;
    Create validation set V by applying strategy 'rand/1/bin' on vector x_i;
    Build regression vector r_i by applying the ANN using T and V;
    if (f(r_i) < f(x_i))
      x_i = r_i;
    end if
  endfor
until (Termination condition meet)
```

The selection process of the original DE algorithm stays intact in nnDE, i.e., when the fitness of the regression trial vector $\mathbf{r}_i$ is better than the fitness of the current vector $\mathbf{x}_i$, the vector $\mathbf{x}_i$ is replaced by vector $\mathbf{r}_i$. Note that here only one fitness evaluation is spend because the generation of the regression vector is performed in genotypic and not in phenotypic search space.

4 Experimental results

The goal of our experimental work is to show that using the ANN regression within the DE algorithm (nnDE) can improve the results of the original DE algorithm significantly. In line with this, the results of nnDE are compared with the results of the original DE as well as self-adaptive jDE [3] algorithms obtained by optimizing the suite of ten well-known functions of dimensions $D = 10$, $D = 20$, and $D = 30$ taken from the literature (Table 2). In order to complete this comparative study, also two swarm intelligence algorithms were included namely, the classical firefly algorithm (FA) [28] and bat algorithm (BA) [29].

Table 2: Definitions of benchmark functions

f	Function	Definition	Range
f_1	Griewangk	$f(\mathbf{x}) = -\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{n} \frac{x_i^2}{4000} + 1$	$-600 \leq x_i \leq 600$
f_2	Rastrigin	$f(\mathbf{x}) = n * 10 + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$-15 \leq x_i \leq 15$
f_3	Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$-15 \leq x_i \leq 15$
f_4	Ackley	$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(20 + e^{-20} e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}\right)$	$-32.768 \leq x_i \leq 32.768$
f_5	Schwefel	$f(\mathbf{x}) == 418.9829 * D - \sum_{i=1}^{D} s_i \sin(\sqrt{\lvert s_i \rvert})$	$-500 \leq x_i \leq 500$
f_6	Sphere	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$-600 \leq x_i \leq 600$
f_7	Easom	$f(\mathbf{x}) = -(-1)^D (\prod_{i=1}^{D} \cos^2(x_i)) \exp[-\sum_{i=1}^{D} (x_i - \pi)^2]$	$-2\pi \leq x_i \leq 2\pi$
f_8	Michalewicz	$f(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i)[\sin(\frac{ix_i^2}{\pi})]^{2\cdot 10}$	$0 \leq x_i \leq \pi$
f_9	Xin-She Yang	$f(\mathbf{x}) = (\sum_{i=1}^{D} \lvert x_i \rvert) \exp[-\sum_{i=1}^{D} \sin(x_i^2)]$	$-2\pi \leq x_i \leq 2\pi$
f_{10}	Zakharov	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + (\frac{1}{2}\sum_{i=1}^{D} ix_i)^2 + (\frac{1}{2}\sum_{i=1}^{D} ix_i)^4$	$-6 \leq x_i \leq 10$

The control parameters of the DE and nnDE algorithms during the test were set as follows: $F = 0.5$, $CR = 0.9$, and $NP = 100$. The population size parameter NP is the same also by all algorithms in tests. The jDE algorithm parameters were set as follows: $F \in [0.1, 1.0]$, $CR \in [0.0, 1.0]$. FA used the following control parameters: $\alpha = 0.1$, $\beta = 0.2$, and $\gamma = 0.9$, while the BA algorithm parameters were set as follows: loudness $A = 0.5$, pulse rate $r = 0.5$, and frequency $Q \in [0.0, 2.0]$. As the termination condition, the fitnes function evaluations were used, as $T_{\max} = 1,000 \cdot D$. Each function was optimized 25-times. The ANN terminates when the error rate is snaller than zero or the maximum 1,000 iterations of back-propagations were reached. The ANN implementation from the OpenCV library was used in the nnDE algorithm.

The obtained results of the mentioned algorithms according to mean values and their standard deviations for ten functions of dimensions $D = 10$, $D = 20$, and $D = 30$ are illustrated in Table 3, from which it can be seen that the nnDE algorithm outperformed the results of the other algorithms for all functions except the Schwefel (f_5) and the Easom (f_7) functions, where the DE and jDE achieved better results.

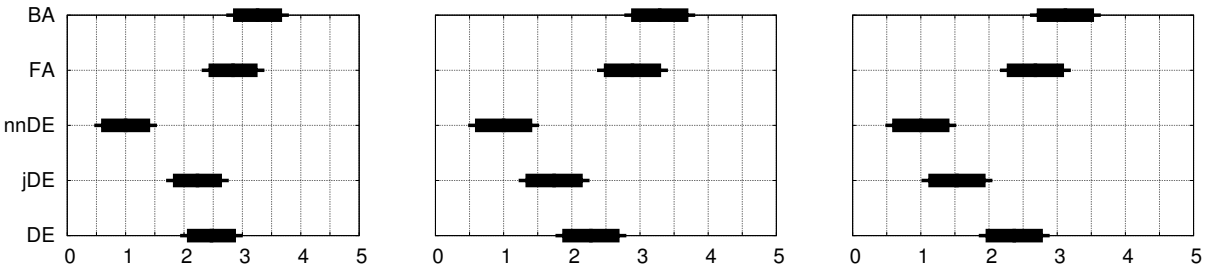
In order to evaluate the quality of the results statistically, Friedman tests [8] were conducted to compare the average ranks of the compared algorithms. Thus, a null-hypothesis is placed to state: two algorithms are

Table 3: Comparison of the results of various algorithms

F	D	DE		jDE		nnDE		FA		BA	
		Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
f_1	10	6.5E-01	1.0E-01	4.7E-01	1.0E-01	7.8E-06	3.0E-05	4.2E-01	3.3E-01	1.1E+01	7.4E+00
	30	1.0E+00	3.1E-02	2.8E-01	7.5E-02	4.2E-08	1.5E-07	4.5E-01	4.3E-01	4.1E+01	1.5E+01
	50	1.0E+00	2.2E-02	4.4E-02	2.8E-02	0.0E+00	0.0E+00	7.0E-01	6.3E-01	8.5E+01	2.2E+01
f_2	10	4.5E+01	5.8E+00	1.9E+01	2.8E+00	1.4E-03	5.7E-03	5.0E+01	4.8E+01	1.8E+02	1.0E+02
	30	1.4E+02	1.2E+01	4.4E+01	5.3E+00	5.5E-04	2.4E-03	1.5E+02	1.4E+02	6.1E+02	2.0E+02
	50	2.3E+02	1.3E+01	6.6E+01	6.5E+00	1.1E-04	4.9E-04	2.4E+02	2.4E+02	1.1E+03	2.8E+02
f_3	10	3.2E+01	1.3E+01	5.7E+01	3.2E+01	8.9E+00	2.0E-02	8.9E+01	6.6E+01	2.5E+05	8.4E+05
	30	2.2E+02	9.2E+01	1.4E+02	6.1E+01	1.8E+01	3.8E+00	3.0E+02	2.4E+02	1.7E+06	9.5E+05
	50	4.6E+02	2.3E+02	1.2E+02	7.2E+01	2.9E+01	1.3E-03	2.3E+02	1.9E+02	3.4E+06	4.0E+06
f_4	10	5.8E-01	2.0E-01	2.1E-01	5.6E-02	1.1E-03	3.5E-03	2.1E+01	2.1E+01	9.0E+00	2.8E+00
	30	1.6E+00	3.6E-01	9.2E-02	3.4E-02	5.9E-05	1.6E-04	2.1E+01	2.1E+01	1.4E+01	1.3E+00
	50	1.8E+00	3.2E-01	3.3E-02	8.7E-03	1.6E-06	8.1E-06	2.1E+01	2.1E+01	1.4E+01	9.5E-01
f_5	10	1.7E+03	1.2E+02	7.6E+02	1.7E+02	4.2E+03	4.4E-01	2.4E+03	2.4E+03	2.2E+03	3.2E+02
	30	4.5E+03	2.1E+02	2.0E+03	2.5E+02	8.3E+03	2.6E+01	5.2E+03	5.2E+03	5.5E+03	4.0E+02
	50	7.6E+03	4.4E+02	3.1E+03	3.2E+02	1.3E+04	3.9E+01	6.9E+03	7.0E+03	9.0E+03	6.3E+02
f_6	10	9.6E+00	3.9E+00	3.5E+00	2.3E+00	1.9E-09	6.8E-09	1.0E+00	1.1E+00	3.6E+04	2.4E+04
	30	9.4E+01	4.0E+01	1.9E+00	9.8E-01	3.7E-07	8.5E-07	3.0E+00	2.9E+00	1.8E+05	7.1E+04
	50	1.8E+02	7.1E+01	4.1E-01	1.9E-01	1.6E-06	8.1E-06	5.6E+00	5.6E+00	2.9E+05	5.4E+04
f_7	10	-1.0E+00	1.3E-03	-1.0E+00	3.3E-04	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	30	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	50	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
f_8	10	-5.8E+00	3.6E-01	-8.4E+00	2.6E-01	-2.6E-01	6.3E-01	-2.6E+00	-2.5E+00	-1.0E-01	1.5E-01
	30	-8.3E+00	4.6E-01	-1.5E+01	5.3E-01	-6.2E-01	1.1E+00	-4.2E+00	-4.2E+00	-4.7E-02	5.3E-02
	50	-1.1E+01	6.7E-01	-2.1E+01	6.6E-01	-2.5E+00	1.5E+00	-5.0E+00	-4.6E+00	-3.6E-02	5.5E-02
f_9	10	2.1E-03	2.3E-04	1.4E-03	1.8E-04	2.4E-04	5.5E-04	-2.3E+02	-2.2E+02	-3.2E+02	6.2E+01
	30	2.9E-07	1.2E-08	1.8E-07	2.3E-08	2.4E-04	7.6E-04	-5.4E+02	-5.4E+02	-1.1E+03	1.9E+02
	50	2.5E-11	1.3E-12	1.5E-11	1.3E-12	7.0E-09	2.0E-08	-9.7E+02	-1.0E+03	-2.5E+03	4.0E+02
f_{10}	10	1.2E-01	4.6E-02	5.7E-01	2.9E-01	6.4E-18	2.9E-17	6.3E+01	5.4E+01	2.3E+01	2.4E+01
	30	6.2E+00	1.9E+00	1.2E+01	4.8E+00	1.3E-08	2.3E-08	5.9E+02	2.5E+02	7.8E+01	9.5E+01
	50	3.8E+01	8.7E+00	6.2E+01	2.1E+01	2.2E-06	4.9E-06	1.5E+04	8.0E+03	2.2E+02	2.2E+02

equivalent and therefore, their ranks should be equal. When the null-hypothesis is rejected, the Bonferroni-Dunn test [6] is performed. In this test, the critical difference is calculated between the average ranks of those two algorithms. If the statistical difference is higher than the critical difference, the algorithms are significantly different.

For each algorithm, three Friedman tests were performed regarding data obtained by optimizing 10 functions according to measures, as: the minimum, maximum, mean, and median values together with the standard deviation (i.e., 50 instances per classifier). The tests were conducted at the significance level 0.05. The results of the Friedman non-parametric test can be seen in Fig. 2 that is divided into three diagrams. Each diagram shows the ranks and confidence intervals (critical differences) for the algorithms under consideration with regard to the dimensions of the functions. Note that the significant difference between two algorithms is observed if their confidence intervals denoted as thickened lines in Fig. 2 do not overlap.



(a) $D = 10$ (b) $D = 20$ (c) $D = 30$

Figure 2: Results of the Friedman non-parametric test

Fig. 2.a shows that the results of DE, jDE, FA and BA are outperformed by the results of nnDE algorithm. The DE and jDE are better than the FA and BA, but this advantage is not significant. The situation do not considerably change when the results are compared according to dimensions $D = 20$ and $D = 30$ (Fig. 2.b and 2.c). Also here, the results of DE, FA and BA algorithms are significantly outperformed by the results of nnDE, while the results of jDE are also significantly better than the results of FA and BA.

5 Conclusion

This paper links the ANN with the EAs, more precisely DE, to enhance quality of the offspring generation. Motivation behind this idea was to predict the best location of a trial vector by using an ensemble of DE strategies. The proposed nnDE algorithm tries to find the best trial vector for each parent. The main advantage of this approach is that regressing for the best trial vector is performed in the genotypic search space. Thus, no additional fitness function evaluations is spent.

On the other hand, the quality of solution obtained during the experimental investigations outperformed the results achieved by the original DE, self-adaptive jDE and two swarm intelligence algorithms, i.e., FA and BA. Unfortunatelly, the running time of the nnDE is enormously increased in comparison to the other algorithms in our study. For example, optimizing the Schwefel function of dimension $D = 20$ demanded less than one second by DE or jDE, while nnDE took more than seven hours.

Altough the quality of the achieved results justifies this approach, there is still a need to decrease the time complexity of this algorithm. Almost two possible directions could be followed to decrease this complexity: an own optimized implementation of ANN, and using the regression as a local search improvement heuristic.

References

[1] Barto, A.G.: Reinforcement learning: An introduction. MIT press (1998)

[2] Bigus, J.P.: Data mining with neural networks: solving business problems from application development to decision support. McGraw-Hill, Inc. (1996)

[3] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on **10**(6), 646–657 (2006)

[4] Darwin, C.: On the Origin of Species. Harvard University Press, London (1859)

[5] Das, S., Suganthan, P.: Differential evolution: A survey of the state-of-the-art. Evolutionary Computation, IEEE Transactions on **15**(1), 4–31 (2011)

[6] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)

[7] Fister Jr., I., Fister, I., Brest, J.: Comparing various regression methods on ensemble strategies in differential evolution. In: Proceedings of 19th International Conference on Soft Computing MENDEL 2013, pp. 87–92. Brno (2013)

[8] Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics **11**, 86–92 (1940). DOI 10.1214/aoms/1177731944

[9] Gershenson, C.: Artificial neural networks for beginners. arXiv preprint cs/0308031 (2003)

[10] Hecht-Nielsen, R.: Theory of the backpropagation neural network. In: Neural Networks, 1989. IJCNN., International Joint Conference on, pp. 593–605. IEEE (1989)

[11] Hozjan, T., Turk, G., Srpčič, S.: Fire analysis of steel frames with the use of artificial neural networks. Journal of constructional steel research **63**(10), 1396–1403 (2007)

[12] Kartam, N., Flood, I., Garrett, J.H.: Artificial neural networks for civil engineers: fundamentals and applications. American Society of Civil Engineers (1997)

[13] Kohonen, T.: Self-organizing maps, vol. 30. Springer (2001)

[14] Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Computing **9**(6), 448–462 (2005)

[15] Mallipeddi, R., Mallipeddi, S., Suganthan, P.N.: Ensemble strategies with adaptive evolutionary programming. Information Sciences **180**(9), 1571–1581 (2010)

[16] Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing **11**(2), 1679–1696 (2011)

[17] McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943)

[18] Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. Evolutionary Computation, IEEE Transactions on **13**(2), 398–417 (2009)

[19] Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 2, pp. 1785–1791. IEEE (2005)

[20] Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. Evolutionary Computation, IEEE Transactions on **12**(1), 64–79 (2008)

[21] Rojas, R.: Neutral Networks: A Systematic Introduction. Springer (1996)

[22] Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall (2010)

[23] Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)

[24] Tvrdìk, J.: Competitive differential evolution. In: MENDEL 2006, 12th International Conference on Soft Computing, pp. 7–12. University of Technology, Brno (2006)

[25] Tvrdìk, J.: Differential evolution with competitive setting of its control parameters. TASK Quarterly **11**, 169–179 (2007)

[26] Tvrdìk, J.: Adaptation in differential evolution: A numerical comparison. Applied Soft Computing **9**, 1149–1155 (2009)

[27] Widrow, B., Rumelhart, D.E., Lehr, M.A.: Neural networks: Applications in industry, business and science. Communications of the ACM **37**(3), 93–105 (1994)

[28] Yang, X.S.: Nature-Inspired Metaheuristic Algorithms: Second Edition. Luniver Press (2010)

[29] Yang, X.S.: A new metaheuristic Bat-Inspired algorithm. In: J.R. González, D.A. Pelta, C. Cruz, G. Terrazas, N. Krasnogor (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), vol. 284, chap. 6, pp. 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). DOI 10.1007/978-3-642-12538-6_6