Nonlinear Dyn (2016) 84:895–914
DOI 10.1007/s11071-015-2537-8

ORIGINAL PAPER

Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution

Iztok Fister · Ponnuthurai Nagaratnam Suganthan · Iztok Fister Jr. · Salahuddin M. Kamal · Fahad M. Al-Marzouki · Matjaž Perc · Damjan Strnad

Received: 24 April 2015 / Accepted: 28 November 2015 / Published online: 18 December 2015

Abstract Nature frequently serves as an inspiration for developing new algorithms to solve challenging real-world problems. Mathematical modeling has led to the development of artificial neural networks (ANNs), which have proven especially useful for solving problems such as classification and regression. Moreover, evolutionary algorithms (EAs), inspired by Darwin's natural evolution, have been successfully applied to solve optimization, modeling, and simulation problems. Differential evolution (DE) is a particularly 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, the ANN regression is applied as a local search heuristic within the DE algorithm that tries predicting the best strategy or attempting to generate a better offspring from an ensemble of DE strategies. This local search heuristic is applied to the population of solutions according to a control parameter that regulates between the time complexity of calculation and the quality of the solution. The experiments on a CEC 2014 test suite consisting of 30 benchmark functions reveal the full potential in developing this idea.

Keywords Nonlinear dynamics · Artificial neural network · Differential evolution · Regression · Local search · Ensemble strategies

I. Fister · I. Fister Jr. · D. Strnad
Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia

P. N. Suganthan
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

S. M. Kamal · F. M. Al-Marzouki · M. Perc
Department of Physics, Faculty of Sciences, King Abdulaziz University, Jeddah, Saudi Arabia

M. Perc (✉)
Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, 2000 Maribor, Slovenia
e-mail: matjaz.perc@uni-mb.si; matjaz.perc@um.si

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. In computer sciences, two nature-inspired mechanisms have been widely used: human brains [46] and Darwinian theory of struggle for survival [7]. The former inspiration from the nature has led to the emergence of artificial neural networks (ANNs), while the latter to evolutionary algorithms (EAs). In this paper, ANN is used as a regression method to enhance the performance of differential evolution (DE).

Operation of an ANN simulates the electrochemical activity of brain cells called neurons [46]. The first mathematical model of neurons was devised by McCulloch and Pitts [37]. According to their model, a neuron "fires," when a linear combination of weighted inputs

exceeds some threshold. Nonlinear response characteristic of a neuron is usually achieved through a sigmoid transfer function, which transforms the activation value to neuron output. In the most widely used type of ANNs, the neurons are organized in layers. The outputs of one layer become the weighted inputs to the next layer, with no interconnections of neurons within the layer. One of primary practical uses of ANNs is to perform nonlinear regression on a set of input–output pairs. Network training is executed in a supervised fashion by introducing the inputs to the network, observing the output error with respect to the target values, and adjusting the connection weights to improve the performance in the next round.

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 [48]. 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 or equal to the fitness of its parent at 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 later phases. Therefore, these parameters are encoded into the solution vector and undergo operations of crossover and mutation. Examples of successfully applied adaptive and self-adaptive DE algorithms are jDE [4] and SaDE [42]. Another kind of DE algorithms tries to improve the results of the original DE algorithm by using ensembles of parameters and mutation DE strategies [34,49,50]. A complete survey of DE methods can be found in [8,51].

Selection of proper DE mutation strategy is problem specific. Furthermore, the best strategy may change with the search progress in the same way as other DE parameters. The problem of adapting the DE mutation strategy has previously been addressed in [33]. In this paper, we propose the use of ANN to build an adaptive regression model for the best DE mutation strategy from an ensemble of DE strategies.

In [14], various DE strategies are applied for each individual, where the best value of element obtained by all strategies in the DE ensemble is used to predict the best value of the corresponding trial vector. This contribution tries to overcome the time complexity of the ANN regression applied to each individual. Here, the ANN regression is used as a local search heuristic applied to a candidate solution according to the control parameter called probability of regression. The higher the value of this parameter, the more candidate solutions undergo ANN local search heuristic.

The structure of the remainder of the paper is as follows: In Sect. 2, we give a short overview of ANN and DE. Section 3 proposes a new DE algorithm with ANN-based regression of DE strategies (nnDE). The experiments and the results are presented in Sect. 4. The paper concludes with a review of the paper contributions and prospects for future work.

2 Background

2.1 Artificial neural networks

An ANN is a mathematical model of human brain. It consists of a set of interconnected artificial neurons, which simulate the operation of natural neurons (i.e., brain cells) [45]. The electrochemical signals in the brain are amplified and propagated along neuronal chains, whereby each neuron receives a number of input signals through ramified sensors (dendrites) and forwards an output signal through its single extension (axon) [21]. The simplest artificial neuron model is shown in Fig. 1a. It receives the input vector $\mathbf{x} = (x_0, \ldots, x_n)$ and produces output y using the following equation [37]:

$$y = \phi\left(\sum_{i=0}^{n} w_i x_i\right) \tag{1}$$

The signal transfer function is programmable through a set $\mathbf{w} = (w_0, \ldots, w_n)$ of weights on the input lines, where line 0 serves as an intercept with fixed input value $x_0 = -1$. A common choice for the transfer function ϕ is a Heaviside function or a sigmoid function like tanh.

With respect to the variety of connectivity types that emerge in different functional areas of the brain, many different neural network architectures have been proposed in the past. The one that received the widest prac-

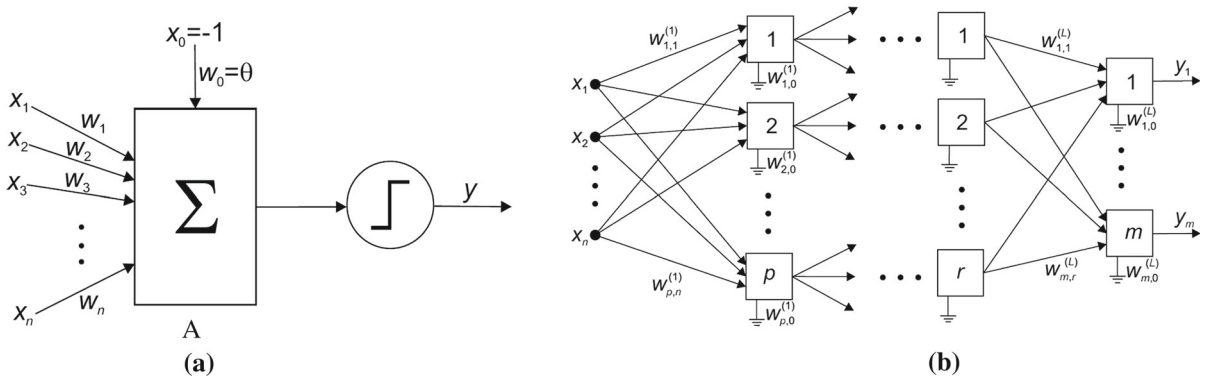


Fig. 1 An example of an artificial neuron (**a**), and an artificial neural network (**b**). **a** Artificial neuron, **b** Multilayer perception

tical application and is also employed in the context of our paper is a feedforward multilayer ANN that consists of neurons organized in several layers (Fig. 1b). Numerical input signals enter the network on the left and propagate through layers toward the outputs on the right. The interlayer signaling works by using the outputs of layer i as inputs of layer $i+1$. An established term for such type of ANN is a multilayer perceptron (MLP). All layers except the last output layer are referred to as hidden layers.

The spectrum of application domains for the ANNs is wide and includes, among others, applications in industry and business [54], data mining [3], civil engineering [27], and fire safety [25]. In the field of machine learning, the ANNs are used in classification and regression tasks [28,30,56].

In its role as nonlinear regressor, the MLP must be trained using a set of training samples with known target values. This approach is known as supervised learning [23]. The training procedure is an iterative process in which the network weights are progressively adjusted such that the discrepancy between the network outputs and target values is minimized. The common error measures for network prediction quality are the mean-squared error (MSE) and the cross-entropy error (CEE). The best known supervised learning algorithm for MLP is the back-propagation method, which uses the gradient of the error function to adapt the weight vectors. Weight changes are usually performed after the presentation of each individual training pattern and the determination of output error, but can be delayed to after the completion of a cycle of presentations (called an epoch). The termination criterion for the training is determined by the allowed error tolerance, maximum number of training epochs, or one of the more advanced methods like cross-validation of prediction efficiency on a separate test set.

In this paper, we propose the use of a two-layer MLP as an aggregator for an ensemble of DE strategies, where the best member of current population is the regression target and the trial vectors derived by different DE strategies are used as training inputs.

2.2 Differential evolution

Differential evolution (DE) belongs to the class of evolutionary algorithms and is appropriate for solving continuous as well as discrete optimization problems. DE was introduced by Storn and Price in 1995 [48] and since then many DE variants have been proposed. 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:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r0}^{(t)} + F \cdot (\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}), \quad \text{for} \quad i = 1 \ldots NP, \qquad (2)$$

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 $1 \ldots NP$ 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 Eq. (2) or parent at the same index position i. Mathematically, this crossover can be expressed as follows:

Table 1 An ensemble of DE strategies (ES)

No.	Strategy	Mutation expression	Crossover
1	Best/1/Exp	$x_{i,j}^{(t+1)} = \text{best}_j^{(t)} + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	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	$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (\text{best}_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	Exponential
4	Best/2/Exp	$x_{i,j}^{(t+1)} = \text{best}_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$	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	$x_{i,j}^{(t+1)} = \text{best}_i^{(t)} + F \cdot (x_{r1,i}^{(t)} - x_{r2,i}^{(t)})$	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	$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + F \cdot (\text{best}_i^{(t)} - x_{i,j}^{(t)}) + F \cdot (x_{r1,j}^{(t)} - x_{r2,j}^{(t)})$	Binomial
9	Best/2/Bin	$x_{i,j}^{(t+1)} = \text{best}_i^{(t)} + F \cdot (x_{r1,i}^{(t)} + x_{r2,i}^{(t)} - x_{r3,i}^{(t)} - x_{r4,i}^{(t)})$	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

$$w_{i,j}^{(t)} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0,1) \leq CR \vee j = j_{\text{rand}}, \\ x_{i,j}^{(t)} & \text{otherwise}, \end{cases} \tag{3}$$

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:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if} \quad f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise}. \end{cases} \tag{4}$$

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*).

2.3 An evolution of DE algorithms

Since its introduction in 1995, many variants of DE algorithm have been developed so far. Zhang et al. [55] combined differential evolution with particle swarm optimization, and result was a new algorithm called DEPSO. Fan and Lampinen [13] added a new trigonometric mutation operator to DE in 2003. Lin et al. [31] developed co-evolutionary hybrid DE. Chakraborty et al. [6] proposed an improved variant of original DE/best/1 scheme by utilizing the concept of the local topological neighborhood of each vector. Scheme tries to balance exploration and exploitation abilities of DE, without using additional FES. Qin et al. [42] proposed a differential evolution algorithm with strategy adaptation. In 2006, Brest [4] proposed the concept of self-adaptation of control parameters. A new algorithm jDE was proposed in [4]. Rahnamayan et al. [43] incorporated an opposition-based mechanism in DE, while Das et al. [9] proposed DE using neighborhood-based mutation operator. Piotrowski [41] combined some well-known DE approaches and gathered together in one framework as a new adaptive memetic DE with global and local neighborhood-based mutation operators. Han et al. [22] created dynamic group-based differential evolution using a self-adaptive strategy to cope with global optimization problems, while Cai and Wang [5] developed differential evolution with neighborhood and direction information. Neri et al. [38–40] proposed compact differential evolution (cDE) which could run also on very limited hardware.

Differential evolution has been used to solve practical problems such as electromagnetics [44], economic emission load dispatch problems [2], crop planning model [1], unit commitment problem [10], short-term electrical power generation scheduling [52], ANNs design [20], protein structure prediction [47] and many more.

2.3.1 Ensemble DE methods

In the literature, some ensemble DE methods were proposed. Mallipeddi et al. [34–36] proposed an EPSDE algorithm (differential evolution algorithm with ensemble of parameters and mutation strategies) where a pool of distinct mutation strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring. Mallipeddi and Suganthan [32] also proposed differential evolution algorithm with ensemble of populations to deal with global numerical optimization. Fister et al. [17] applied ensemble of DE strategies to the hybrid bat algorithm [18]. Elsayed et al. [12] introduced an algorithm framework that uses multiple search operators in each generation. An appropriate mix of search operators is determined adaptively. LaTorre [29] explored the use of a hybrid memetic algorithm based on the multiple offspring framework. Their algorithm combines the explorative/exploitative strength of two heuristic search methods that separately obtain very competitive results. Vrugt et al. [53] proposed a concept, where different search algorithms run concurrently and learn from each other through information exchange using a common population.

2.3.2 jDE algorithm

In 2006, Brest et al. [4] proposed an effective DE variant (jDE), where control parameters are changed during the run. In this case, two parameters, namely scale factor F and crossover rate CR, are changed during the run. In jDE, every individual is extended with F and CR:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}, F_i^{(t)}, CR_i^{(t)}).$$

jDE are changing parameters F and CR according to the following equations:

$$F_i^{(t+1)} = \begin{cases} F_l + \text{rand}_1 * (F_u - F_l) & \text{if} \quad \text{rand}_2 < \tau_1, \\ F_i^{(t)} & \text{otherwise}, \end{cases} \tag{5}$$

$$CR_i^{(t+1)} = \begin{cases} \text{rand}_3 & \text{if} \quad \text{rand}_4 < \tau_2, \\ CR_i^{(t)} & \text{otherwise}, \end{cases} \tag{6}$$

where $\text{rand}_{i=1\ldots4} \in [0, 1]$ are randomly generated values, τ_1 and τ_2 are learning steps, F_l and F_u lower and upper bound for parameter F.

3 The proposed algorithm

The proposed ANN regression on ensemble of DE strategies (nnDE) (pseudo-code in Algorithm 1) modifies the generation of the trial vector in the original DE algorithm. The trial vector $\mathbf{y}_i$ is produced from the original vector $\mathbf{x}_i$ by the default DE mutation strategy. A local search step (lines 5–10 in Algorithm 1) is then performed with probability p_r. The regression probability has a great impact on the performance of the algorithm, because it controls the number of local search steps to be launched. Therefore, it influences the exploration and exploitation of the evolutionary search process. The higher the probability of the regression, the more local search steps are initiated. On the other hand, each local search demands an additional processor time that may cause a performance degradation of the algorithm. As a result, the proper value of this parameter needs to be found for each specific problem on a case-by-case basis.

During each local search, a regression ANN is trained using a training set $T_i = \{(\mathbf{t}_i^{(k)}, \mathbf{x}_{\text{best}}); k = 1, \ldots, P\}$, where each training pattern consists of an input vector $\mathbf{t}_i^{(k)}$ of dimension D and a vector of network target outputs, which is $\mathbf{x}_{\text{best}}$ in all cases. Input vectors $\mathbf{t}_i^{(k)}$ are derived from the currently processed population member $\mathbf{x}_i$ using randomly selected DE strategies from the ensemble of strategies (ES) collected in Table 1. The neural network thus has D inputs, $1 + \log_2 D$ hidden neurons, and D output neurons.

A trained ANN performs the regression of the best found solution from the set of trial solutions provided by the ensemble of DE strategies. When the strategies agree and the trial solutions are similar, the nonlinear transformation represented by the trained ANN performs a narrowly directed local search. When this is not the case, a random search ensues.

After training, the regression vector $\mathbf{r}_i$ is obtained by introducing the trial vector $\mathbf{y}_i$ to the network. The regression vector replaces the trial and is used in its place for subsequent fitness comparison with the original vector $\mathbf{x}_i$. Note that only one fitness evaluation is spent during this local search step because the generation of the regression vector is performed in genotypic and not in phenotypic search space.

Algorithm 1 The nnDE algorithm

```
Initialize the DE population x_i = (x_i1, ..., x_iD) for i = 1...NP where NP is the population size.
repeat
  for i = 1 to NP do
    Generate trial vector y_i from x_i using default DE strategy;
    if rand() < p_r then
      Create a training set T_i from vector x_i using ES and target vector x_best;
      Train the ANN using T_i till the number of epochs epoch exceeded;
      Build regression vector r_i from y_i using the trained ANN;
      y_i = r_i;
    end if
    if f(y_i) < f(x_i) then
      x_i = y_i;
      if f(y_i) < f(x_best) then
        x_best = y_i;
      end if
    end if
  end for
until DE termination condition is met
```

In general, the efficiency of the local search depends on two facts [24]: How often it is launched and how extensively the local search process explores the search space. The former is controlled with the parameter p_r, while the latter depends on the number of epochs needed for training the ANN. Typically, a designer needs to choose between the often launched short-term and rarely launched long-term local search. The short-term local search demands a smaller, while the long-term a larger number of ANN training epochs. In this study, a rarely launched long-term local search heuristic is tested.

4 Experimental results

The goal of our experimental work is to show that using the ANN-based regression within the DE (nnDE) and self-adaptive jDE [4] (nnjDE) can improve the results of the original DE and jDE algorithms. In line with this, the comparative study of the mentioned algorithms was performed by solving the CEC 2014 test suite. The SaDE [42] method was also included in this comparative study. In order to analyze the impact of ANN regression on the original DE and jDE algorithms, the following issues were investigated:

- the impact of the regression probability p_r and the number of ANN training epochs *epoch*,
- the impact of the fitness function evaluations, and
- the impact of problem dimensionality.

A comparative efficiency study of methods with and without the use of ANN local search was performed, and their convergence graphs were analyzed. 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 was the same for all compared algorithms. The proper value of this parameter mainly depends on the problem to be solved and was determined after extensive experimentation. The jDE and nnjDE algorithms' parameters were set as follows: $F \in [0.1, 1.0]$, $CR \in [0.0, 1.0]$, $\tau_1 = \tau_2 = 0.1$. As a termination condition, the number of fitness function evaluations was used, as specified in the CEC 2014 benchmark suite, i.e., $T_{\max} = 10000 \cdot D$. Each function was optimized 25 times. The ANN training in nnDE and nnjDE was terminated at 1000 epochs or when the training MSE dropped below 10^{-6}, whichever occurred first. The ANN implementation from the OpenCV library was used that supports a back-propagation method of training, which was used in our tests with the learning rate and momentum scale set to 0.5 and 0.1, respectively.

4.1 Test suite

The CEC 2014 test suite (Table 2) consists of 30 benchmark functions that are divided into four classes:

- unimodal functions (1–3),
- simple multimodal functions (4–16),
- hybrid functions (17–22),
- composition functions (23–30).

Unimodal functions have a single global optimum and no local optimums. Unimodal functions in this suite are non-separable and rotated. Multi-modal functions are either separable or non-separable. In addition, they are also rotated or/and shifted. To develop the hybrid functions, the variables are randomly divided into some subcomponents, and then, different basic functions are used for different subcomponents [26]. Composition functions consist of a sum of two or more basic functions. In this suite, hybrid functions are used as the basic functions to construct composition functions. Characteristics of these hybrid and composition functions depend on the characteristics of the basic functions.

The functions of dimensions $D = 10$, $D = 20$, and $D = 30$ were used in our experiments. The

Table 2 Summary of the CEC'14 test functions

	No.	Functions	$F_i^* = F_i(x*)$
Unimodal functions	1	Rotated high conditioned elliptic function	100
	2	Rotated bent cigar function	200
	3	Rotated discus function	300
Simple multimodal functions	4	Shifted and Rotated Rosenbrocks function	400
	5	Shifted and rotated Ackleys function	500
	6	Shifted and rotated Weierstrass function	600
	7	Shifted and rotated Griewanks function	700
	8	Shifted Rastrigins function	800
	9	Shifted and rotated Rastrigins function	900
	10	Shifted Schwefels function	1000
	11	Shifted and rotated Schwefels function	1100
	12	Shifted and rotated Katsuura function	1200
	13	Shifted and rotated HappyCat function	1300
	14	Shifted and rotated HGBat function	1400
	15	Shifted and rotated expanded Griewanks plus Rosenbrocks function	1500
	16	Shifted and rotated expanded Scaffers F6 function	1600
Hybrid functions	17	Hybrid function 1 ($N = 3$)	1700
	18	Hybrid function 2 ($N = 3$)	1800
	19	Hybrid function 3 ($N = 4$)	1900
	20	Hybrid function 4 ($N = 4$)	2000
	21	Hybrid function 5 ($N = 5$)	2100
	22	Hybrid function 6 ($N = 5$)	2200
Composition functions	23	Composition function 1 ($N = 5$)	2300
	24	Composition function 2 ($N = 3$)	2400
	25	Composition function 3 ($N = 3$)	2500
	26	Composition function 4 ($N = 5$)	2600
	27	Composition function 5 ($N = 5$)	2700
	28	Composition function 6 ($N = 5$)	2800
	29	Composition function 7 ($N = 3$)	2900
	30	Composition function 8 ($N = 3$)	3000

search range of the problem variables is limited to $x_i \in [-100, 100]$.

4.2 Impacts of the regression probability and the number of ANN training epochs

The goal of this experiment is twofold. Firstly, we aim to discover how the parameter p_r affects the results of the nnDE and nnjDE algorithms on the CEC 2014 test suite, and secondly, we want to explore how the number of ANN training epochs influences the results of the nnDE algorithm on the same test suite.

In the first experiment, the probability of local search application was varied in the interval $p_r \in [0.005, 0.05]$ in steps of 0.005, resulting in ten launch configurations per problem size D. The results of each configuration according to five statistical measures (i.e., minimum, maximum, average, median, and standard deviation) accumulated over 25 runs for each function were aggregated into a statistical classifier consisting of $30 \times 5 = 150$ variables that served as input to Friedman statistical test. The Friedman test [11] calculates the average method ranks over all problem instances (i.e., benchmark functions) for each of the test configurations. For the case $D = 10$, Fig. 2

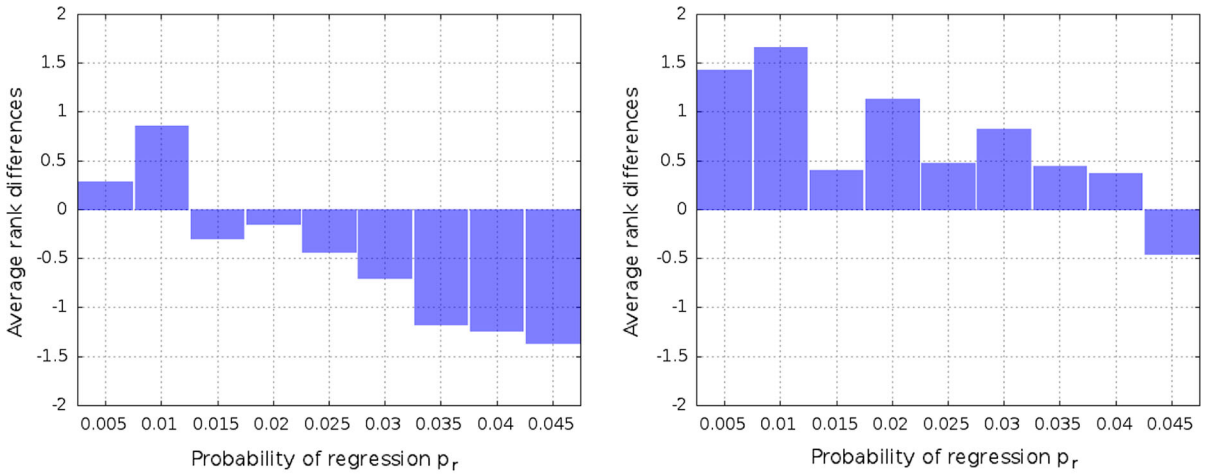


Fig. 2 Average rank differences of nnDE (**a**) and nnjDE (**b**) algorithms achieved over all problem instances with $D = 10$ for different settings of p_r. The rank difference is expressed as *bar* height and direction. As a result, all *bars* higher than zero indicate that the corresponding hybrid DE algorithm outperformed the original DE algorithm in a particular parameter setting

illustrates the differences of average ranks achieved by nnDE and nnjDE in comparison with the original DE and jDE methods, respectively. In the figure, each positive average rank distance means that the corresponding instance of nnDE and nnjDE outperformed the results of the original DE and jDE algorithms, and vice versa. Namely, negative average rank differences indicate that the original DE and jDE algorithms outperformed the results achieved by the nnDE and nnjDE algorithms.

Two conclusions can be deduced from the experimental results. Firstly, the obtained results strongly depend on the probability of regression p_r. Secondly, the best results for $D = 10$ are obtained when $p_r = 0.01$, which complies with Piotrowski [41] who proposed performing the local search with probability $p_r = 0.005$ when $100 \cdot D$ fitness function evaluations are spent per launch.

In the second experiment, the ANN training epochs in the nnDE algorithm were varied by $epoch \in \{100, 500, 1000, 2000, 5000\}$ for benchmark functions of dimensions $D = 10$, $D = 20$, and $D = 30$. Extensive experiments showed that the setting $p_r = 0.01$, where one local search step is launched in average when using the population size $Np = 100$, is not optimal. It turns out that the optimal value of this parameter lays in a range $p_r \in (0.0, 0.01]$. Therefore, it is varied as $p_r \in \{0.01, 0.005, 0.003\}$ in our tests, which corresponds to one call of the local search heuristic every one, two, and three generations, respectively.

The average ranks obtained by Friedman nonparametric tests for experiment results obtained by nnDE with different problem dimensions are illustrated in Fig. 3. Each graph plots the average rank against the number of ANN training epochs. Each line corresponds to one of the tested values of p_r.

Two facts can be concluded from the figure, as follows:

- the smaller the probability of regression, the better the results,
- the higher the dimension of the problem, the smaller the number of epochs required.

in order to obtain the best results when optimizing the benchmark functions of dimension $D = 10$, the number of ANN training epochs $epoch = 2000$ is needed, while $epoch = 100$ is enough to obtain the best results for dimension $D = 30$. The number of ANN training epochs depends on the probability of regression by optimizing the functions of dimensions $D = 20$, i.e., $epoch \in \{100, 500, 1000, 2000, 5000\}$ and $p_r \in \{0.01, 0.005, 0.003\}$.

Although infrequently executed, the local search step significantly influences the results of the optimization. On the other hand, the ANN training starts to dominate the optimization runtime with a growing number of training epochs. Fortunately, the smaller number of required epochs and infrequent launching of local search steps make solving of the problem optimization tractable in higher dimensions.

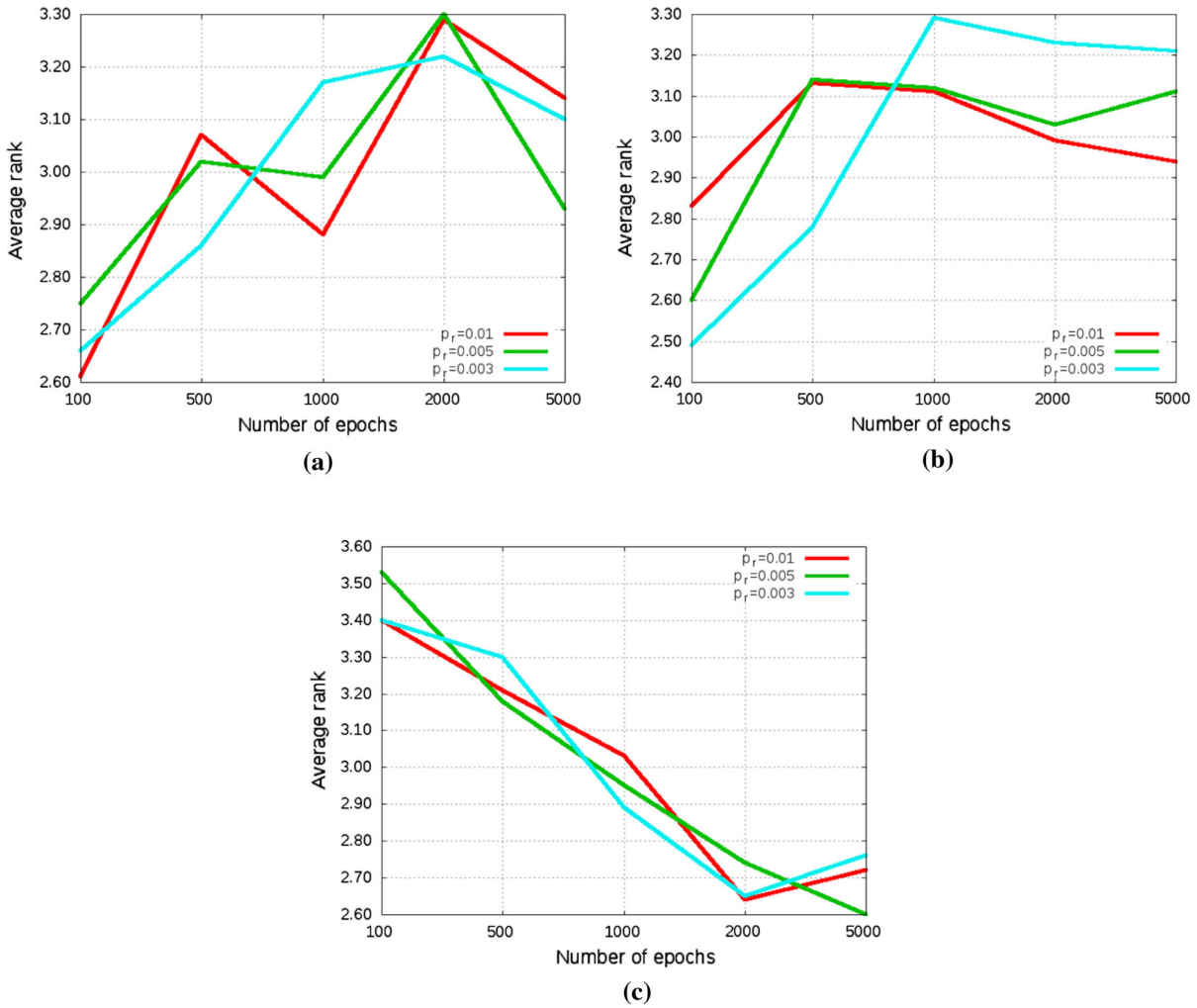


Fig. 3 Influence of the epoch number on the results of the optimization obtained by the nnDE, where each diagram consists of three curves representing average ranks according to the specific probability of regression p_r. **a** $D = 10$, **b** $D = 20$, **c** $D = 30$

4.3 Impact of the fitness function evaluations

One of the more reliable indicators of search stagnation is when the best result is not improved for a long term. Alternatively, this can also mean that the search process gets stuck in a local optimum. In order to detect these undesirable situations during the run of nnDE, the fitness values were monitored at three different phases of the search process, i.e., at 1/25, 1/5, and at the final fitness function evaluation. The results of this test are collated in Tables 3 and 4 for functions of dimension $D = 20$.

As can be seen from Tables 3 and 4, nnjDE successfully progressed toward the global optimum according to all benchmark functions, i.e., no stagnation of the search process is detected.

4.4 Impact of the problem dimensionality

The goal of this experiment is to discover how the quality of the results obtained by the nnDE depends on the dimension of the problem. In line with this, three different dimensions of the benchmark functions $D \in \{10, 20, 30\}$ were taken into account. The results of the tests according to five measures are presented in Tables 5 and 6.

In this experiment, it was expected that the functions of the higher dimensions would be harder to optimize,

Table 3 Results of the nnDE with $p_r = 0.003$ and $epoch = 2000$ showing an impact of the fitness function evaluations measured after $\frac{1}{25}$, $\frac{1}{5}$, and $\frac{1}{1}$ of the maximum fitness function evaluations for dimension $D = 20$—Part 1/2

Func.	FEs	Best	Worst	Mean	Median	Std
1	8.00E+003	1.54E+006	6.18E+006	3.71E+006	3.44E+006	1.16E+006
	4.00E+004	8.94E+002	9.90E+003	3.55E+003	2.85E+003	2.14E+003
	2.00E+005	4.69E−012	1.03E−009	1.52E−010	8.89E−011	2.32E−010
2	8.00E+003	1.47E+008	3.67E+008	2.34E+008	2.16E+008	5.95E+007
	4.00E+004	1.57E+001	1.08E+002	4.34E+001	4.62E+001	2.24E+001
	2.00E+005	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
3	8.00E+003	2.93E+003	6.51E+003	4.37E+003	4.24E+003	1.01E+003
	4.00E+004	5.81E−002	4.50E−001	1.75E−001	1.52E−001	1.03E−001
	2.00E+005	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
4	8.00E+003	4.27E+001	9.92E+001	6.81E+001	6.54E+001	1.59E+001
	4.00E+004	1.53E+001	2.32E+001	1.89E+001	1.89E+001	1.73E+000
	2.00E+005	0.00E+000	1.42E+001	7.65E+000	1.11E+001	6.19E+000
5	8.00E+003	2.05E+001	2.09E+001	2.07E+001	2.08E+001	9.06E−002
	4.00E+004	2.02E+001	2.07E+001	2.05E+001	2.05E+001	1.19E−001
	2.00E+005	2.00E+001	2.02E+001	2.01E+001	2.01E+001	4.73E−002
6	8.00E+003	1.08E+001	2.08E+001	1.56E+001	1.56E+001	2.61E+000
	4.00E+004	3.50E−001	4.49E+000	2.53E+000	2.68E+000	1.19E+000
	2.00E+005	0.00E+000	3.69E+000	2.03E+000	2.10E+000	1.07E+000
7	8.00E+003	2.57E+000	5.63E+000	3.64E+000	3.60E+000	7.26E−001
	4.00E+004	1.79E−004	4.94E−001	2.50E−002	1.14E−003	9.81E−002
	2.00E+005	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
8	8.00E+003	8.95E+001	1.38E+002	1.13E+002	1.15E+002	1.32E+001
	4.00E+004	8.01E+000	4.02E+001	1.92E+001	1.80E+001	7.89E+000
	2.00E+005	4.97E+000	3.19E+001	1.23E+001	1.19E+001	5.18E+000
9	8.00E+003	9.90E+001	1.50E+002	1.23E+002	1.24E+002	1.46E+001
	4.00E+004	1.16E+001	4.39E+001	2.48E+001	2.52E+001	9.76E+000
	2.00E+005	7.96E+000	3.88E+001	2.03E+001	1.79E+001	8.25E+000
10	8.00E+003	2.75E+003	4.03E+003	3.50E+003	3.55E+003	3.59E+002
	4.00E+004	2.39E+002	1.40E+003	5.95E+002	5.85E+002	2.75E+002
	2.00E+005	3.07E+001	5.81E+002	2.43E+002	2.67E+002	1.52E+002
11	8.00E+003	2.98E+003	4.31E+003	3.78E+003	3.81E+003	3.80E+002
	4.00E+004	1.96E+002	1.43E+003	9.54E+002	1.03E+003	3.09E+002
	2.00E+005	2.76E+001	1.29E+003	7.89E+002	8.43E+002	3.52E+002
12	8.00E+003	1.58E+000	2.94E+000	2.39E+000	2.38E+000	3.05E−001
	4.00E+004	4.44E−001	1.87E+000	9.91E−001	9.16E−001	4.39E−001
	2.00E+005	2.27E−002	1.44E−001	6.30E−002	5.42E−002	3.07E−002
13	8.00E+003	4.19E−001	8.13E−001	5.81E−001	5.79E−001	8.88E−002
	4.00E+004	2.45E−001	4.29E−001	3.47E−001	3.46E−001	4.94E−002
	2.00E+005	1.21E−001	2.75E−001	1.94E−001	1.87E−001	3.63E−002

Table 3 continued

Func.	FEs	Best	Worst	Mean	Median	Std
14	8.00E+003	3.31E−001	9.62E−001	4.99E−001	4.69E−001	1.42E−001
	4.00E+004	2.20E−001	3.76E−001	3.05E−001	3.06E−001	4.38E−002
	2.00E+005	1.61E−001	2.95E−001	2.19E−001	2.15E−001	3.14E−002
15	8.00E+003	1.21E+001	2.89E+001	1.87E+001	1.81E+001	4.15E+000
	4.00E+004	6.96E+000	1.06E+001	9.16E+000	9.15E+000	9.22E−001
	2.00E+005	1.04E+000	3.06E+000	1.71E+000	1.60E+000	5.15E−001

Table 4 Results of the nnDE with $p_r = 0.003$ and $epoch = 2000$ showing an impact of the fitness function evaluations measured after $\frac{1}{25}$, $\frac{1}{5}$, and $\frac{1}{1}$ of the maximum fitness function evaluations for dimension $D = 20$—Part 2/2

Func.	FEs	Best	Worst	Mean	Median	Std
16	8.00E+003	8.01E+000	8.91E+000	8.53E+000	8.58E+000	2.33E−001
	4.00E+004	5.55E+000	8.00E+000	7.17E+000	7.33E+000	5.72E−001
	2.00E+005	2.74E+000	5.21E+000	4.01E+000	3.94E+000	6.80E−001
17	8.00E+003	1.34E+004	1.22E+005	4.39E+004	3.87E+004	2.55E+004
	4.00E+004	5.36E+002	1.27E+003	1.02E+003	1.03E+003	1.61E+002
	2.00E+005	3.10E+001	6.68E+002	2.85E+002	2.13E+002	2.16E+002
18	8.00E+003	2.26E+003	5.66E+004	2.05E+004	1.81E+004	1.39E+004
	4.00E+004	2.89E+001	4.89E+001	3.99E+001	4.06E+001	5.87E+000
	2.00E+005	5.05E+000	2.69E+001	2.13E+001	2.27E+001	4.82E+000
19	8.00E+003	6.40E+000	1.12E+001	8.48E+000	8.45E+000	9.21E−001
	4.00E+004	3.01E+000	4.72E+000	3.62E+000	3.55E+000	3.75E−001
	2.00E+005	1.37E−001	2.61E+000	1.62E+000	1.80E+000	6.43E−001
20	8.00E+003	1.54E+002	3.67E+002	2.38E+002	2.30E+002	5.81E+001
	4.00E+004	1.26E+001	3.34E+001	2.62E+001	2.69E+001	5.17E+000
	2.00E+005	1.74E+000	1.59E+001	7.41E+000	5.01E+000	4.74E+000
21	8.00E+003	1.94E+003	4.96E+003	3.30E+003	3.44E+003	8.78E+002
	4.00E+004	3.41E+002	6.77E+002	5.23E+002	5.27E+002	7.83E+001
	2.00E+005	8.92E−001	1.19E+002	1.67E+001	9.53E+000	2.59E+001
22	8.00E+003	1.30E+002	3.80E+002	2.62E+002	2.55E+002	7.24E+001
	4.00E+004	2.54E+001	2.80E+002	6.02E+001	3.42E+001	5.57E+001
	2.00E+005	6.17E+000	2.72E+002	5.28E+001	3.08E+001	5.52E+001
23	8.00E+003	3.33E+002	3.37E+002	3.35E+002	3.35E+002	1.06E+000
	4.00E+004	3.30E+002	3.30E+002	3.30E+002	3.30E+002	3.40E−005
	2.00E+005	3.30E+002	3.30E+002	3.30E+002	3.30E+002	5.80E−014
24	8.00E+003	2.18E+002	2.27E+002	2.24E+002	2.24E+002	2.09E+000
	4.00E+004	2.05E+002	2.12E+002	2.10E+002	2.11E+002	1.27E+000
	2.00E+005	2.00E+002	2.11E+002	2.10E+002	2.10E+002	2.06E+000
25	8.00E+003	2.08E+002	2.15E+002	2.10E+002	2.10E+002	1.71E+000
	4.00E+004	2.03E+002	2.04E+002	2.04E+002	2.04E+002	2.19E−001
	2.00E+005	2.03E+002	2.04E+002	2.03E+002	2.04E+002	2.74E−001

Table 4 continued

Func.	FEs	Best	Worst	Mean	Median	Std
26	8.00E+003	1.00E+002	1.01E+002	1.01E+002	1.01E+002	7.53E−002
	4.00E+004	1.00E+002	1.00E+002	1.00E+002	1.00E+002	5.66E−002
	2.00E+005	1.00E+002	1.00E+002	1.00E+002	1.00E+002	4.69E−002
27	8.00E+003	4.32E+002	8.05E+002	6.39E+002	6.75E+002	1.28E+002
	4.00E+004	3.13E+002	4.46E+002	3.77E+002	3.87E+002	3.20E+001
	2.00E+005	3.00E+002	4.19E+002	3.65E+002	3.67E+002	3.62E+001
28	8.00E+003	7.09E+002	1.07E+003	8.29E+002	8.12E+002	9.01E+001
	4.00E+004	5.51E+002	8.27E+002	6.58E+002	6.30E+002	7.92E+001
	2.00E+005	5.34E+002	8.24E+002	6.46E+002	6.22E+002	8.04E+001
29	8.00E+003	1.09E+003	2.14E+004	6.21E+003	3.68E+003	5.49E+003
	4.00E+004	2.22E+002	3.27E+002	2.77E+002	2.74E+002	3.74E+001
	2.00E+005	2.17E+002	2.49E+002	2.37E+002	2.43E+002	1.18E+001
30	8.00E+003	3.39E+003	3.68E+004	9.61E+003	5.95E+003	9.88E+003
	4.00E+004	5.81E+002	1.24E+003	9.31E+002	9.73E+002	1.76E+002
	2.00E+005	4.94E+002	7.20E+002	5.79E+002	5.75E+002	6.82E+001

Table 5 Results of the nnDE with $p_r = 0.003$ and $epoch \in \{2000, 2000, 100\}$ showing an impact of the dimensionality of the problem measured by function dimensions $D \in \{10, 20, 30\}$, respectively—Part 1/2

Func.	Dim.	Best	Worst	Mean	Median	Std
1	10	0.00E+000	2.84E−014	1.31E−014	1.42E−014	9.98E−015
	20	4.69E−012	1.03E−009	1.52E−010	8.89E−011	2.32E−010
	30	2.60E+004	1.69E+005	7.36E+004	6.30E+004	4.48E+004
2	10	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	20	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	30	0.00E+000	2.84E−014	2.27E−015	0.00E+000	7.87E−015
3	10	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	20	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	30	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
4	10	0.00E+000	9.72E+000	6.36E+000	8.16E+000	3.68E+000
	20	0.00E+000	1.42E+001	7.65E+000	1.11E+001	6.19E+000
	30	1.14E−002	5.65E+001	4.57E+000	1.42E−001	1.54E+001
5	10	2.00E+001	2.04E+001	2.01E+001	2.00E+001	1.24E−001
	20	2.00E+001	2.02E+001	2.01E+001	2.01E+001	4.73E−002
	30	2.00E+001	2.01E+001	2.00E+001	2.00E+001	2.81E−002
6	10	0.00E+000	2.66E+000	9.26E−001	8.95E−001	9.10E−001
	20	0.00E+000	3.69E+000	2.03E+000	2.10E+000	1.07E+000
	30	4.59E−001	1.31E+001	6.18E+000	5.31E+000	3.33E+000
7	10	0.00E+000	3.41E−001	6.76E−002	4.18E−002	7.34E−002
	20	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	30	0.00E+000	9.86E−003	6.90E−004	0.00E+000	2.42E−003

Table 5 continued

Func.	Dim.	Best	Worst	Mean	Median	Std
8	10	0.00E+000	8.95E+000	2.11E+000	1.99E+000	1.87E+000
	20	4.97E+000	3.19E+001	1.23E+001	1.19E+001	5.18E+000
	30	1.39E+001	4.68E+001	2.74E+001	2.59E+001	7.99E+000
9	10	1.99E+000	1.89E+001	7.72E+000	6.96E+000	3.80E+000
	20	7.96E+000	3.88E+001	2.03E+001	1.79E+001	8.25E+000
	30	2.09E+001	7.66E+001	4.08E+001	3.98E+001	1.49E+001
10	10	1.55E+001	6.71E+001	4.11E+001	3.88E+001	1.25E+001
	20	3.07E+001	5.81E+002	2.43E+002	2.67E+002	1.52E+002
	30	3.71E+002	1.44E+003	7.70E+002	7.36E+002	3.03E+002
11	10	3.60E+000	8.65E+002	3.59E+002	2.73E+002	2.66E+002
	20	2.76E+001	1.29E+003	7.89E+002	8.43E+002	3.52E+002
	30	1.74E+003	4.37E+003	2.74E+003	2.62E+003	5.99E+002
12	10	2.59E−005	5.77E−002	2.53E−002	2.71E−002	1.55E−002
	20	2.27E−002	1.44E−001	6.30E−002	5.42E−002	3.07E−002
	30	9.61E−003	1.37E−001	4.76E−002	4.00E−002	2.81E−002
13	10	8.73E−002	1.57E−001	1.23E−001	1.22E−001	2.01E−002
	20	1.21E−001	2.75E−001	1.94E−001	1.87E−001	3.63E−002
	30	1.30E−001	3.05E−001	2.13E−001	2.20E−001	4.66E−002
14	10	6.33E−002	2.11E−001	1.34E−001	1.37E−001	3.80E−002
	20	1.61E−001	2.95E−001	2.19E−001	2.15E−001	3.14E−002
	30	1.93E−001	3.40E−001	2.64E−001	2.83E−001	4.02E−002
15	10	2.88E−001	1.67E+000	7.02E−001	6.25E−001	2.97E−001
	20	1.04E+000	3.06E+000	1.71E+000	1.60E+000	5.15E−001
	30	1.94E+000	5.29E+000	3.41E+000	3.07E+000	1.03E+000

and therefore, the obtained results would be worse. As a matter of fact, this assumption holds in general except for functions f_7, f_{11} and f_{12}, where nnDE achieved better results by optimizing the functions of dimension $D = 20$ than by the other dimensions. Interestingly, the results of optimizing the function f_{26} are equal by the all algorithms in test.

4.5 A comparative study

In order to show that the hybridization with ANN regression improves the results of the original DE and jDE algorithms, a comparative study was performed. In this study, the results of the DE and jDE algorithms were compared with the results of nnDE and nnjDE, i.e., hybridized versions of the DE algorithms with $p_r = 0.003$ and $epoch = 1000$. All the mentioned algorithms used the parameters as reported at the beginning of this section. This comparative study was widened by an additional self-adaptive DE algorithm, i.e., the SaDE using the following parameter settings: F is randomly selected from the normal distribution $N(0.5, 0.3)$, while CR is randomly drawn from a normal distribution $N(CRm_k, 0.1)$. The variable CRm_k denotes an average value of parameter CR for k-th strategy (four strategies were used in the ensemble strategies) during the last $LP = 20$ (i.e., learning rate parameter) generations. The results according to the mean and standard deviation obtained by solving the CEC 2014 benchmark functions of dimension $D = 30$ are illustrated in Table 7. The best results in the table are presented in bold.

As it can be seen from Table 7, the nnjDE outperforms the results of the other observed algorithms twelve times, SaDE eight times, nnDE and DE four times, and jDE once. All algorithms achieved the same results for functions f_{23} and f_{26}. Note that the nnDE

Table 6 Results of the nnDE with $p_r = 0.003$ and $epoch \in \{2000, 2000, 100\}$ showing an impact of the dimensionality of the problem measured by function dimensions $D \in \{10, 20, 30\}$, respectively—Part 2/2

Func.	Dim.	Best	Worst	Mean	Median	Std
16	10	1.44E−001	3.05E+000	1.61E+000	1.51E+000	7.80E−001
	20	2.74E+000	5.21E+000	4.01E+000	3.94E+000	6.80E−001
	30	9.55E+000	1.21E+001	1.08E+001	1.09E+001	6.59E−001
17	10	4.23E−005	1.16E+001	2.20E+000	4.16E−001	4.13E+000
	20	3.10E+001	6.68E+002	2.85E+002	2.13E+002	2.16E+002
	30	7.82E+001	1.26E+003	6.45E+002	5.95E+002	3.05E+002
18	10	1.23E−002	1.28E+000	2.48E−001	2.26E−001	2.58E−001
	20	5.05E+000	2.69E+001	2.13E+001	2.27E+001	4.82E+000
	30	5.48E+000	1.74E+001	9.92E+000	8.80E+000	3.31E+000
19	10	3.65E−002	8.11E−001	3.12E−001	3.27E−001	2.22E−001
	20	1.37E−001	2.61E+000	1.62E+000	1.80E+000	6.43E−001
	30	2.32E+000	7.79E+000	3.57E+000	3.26E+000	1.15E+000
20	10	2.05E−004	4.93E−001	1.06E−001	1.81E−002	1.64E−001
	20	1.74E+000	1.59E+001	7.41E+000	5.01E+000	4.74E+000
	30	4.34E+000	1.78E+001	8.65E+000	8.08E+000	3.39E+000
21	10	2.76E−004	1.12E+000	3.93E−001	3.24E−001	3.47E−001
	20	8.92E−001	1.19E+002	1.67E+001	9.53E+000	2.59E+001
	30	2.25E+000	6.03E+002	2.61E+002	2.36E+002	2.09E+002
22	10	1.52E−002	9.50E−001	2.85E−001	3.29E−001	2.56E−001
	20	6.17E+000	2.72E+002	5.28E+001	3.08E+001	5.52E+001
	30	2.07E+001	6.02E+002	3.12E+002	3.13E+002	1.65E+002
23	10	3.29E+002	3.29E+002	3.29E+002	3.29E+002	2.32E−013
	20	3.30E+002	3.30E+002	3.30E+002	3.30E+002	5.80E−014
	30	3.15E+002	3.15E+002	3.15E+002	3.15E+002	4.67E−002
24	10	1.09E+002	1.26E+002	1.16E+002	1.16E+002	4.54E+000
	20	2.00E+002	2.11E+002	2.10E+002	2.10E+002	2.06E+000
	30	2.21E+002	2.37E+002	2.25E+002	2.24E+002	3.62E+000
25	10	1.14E+002	2.01E+002	1.83E+002	2.01E+002	3.23E+001
	20	2.03E+002	2.04E+002	2.03E+002	2.04E+002	2.74E−001
	30	2.02E+002	2.04E+002	2.03E+002	2.03E+002	2.87E−001
26	10	1.00E+002	1.00E+002	1.00E+002	1.00E+002	3.27E−002
	20	1.00E+002	1.00E+002	1.00E+002	1.00E+002	4.69E−002
	30	1.00E+002	1.00E+002	1.00E+002	1.00E+002	4.99E−002
27	10	1.01E+000	4.00E+002	1.49E+002	2.17E+000	1.86E+002
	20	3.00E+002	4.19E+002	3.65E+002	3.67E+002	3.62E+001
	30	3.00E+002	4.72E+002	3.78E+002	3.87E+002	5.15E+001
28	10	3.60E+002	4.71E+002	3.75E+002	3.69E+002	2.08E+001
	20	5.34E+002	8.24E+002	6.46E+002	6.22E+002	8.04E+001
	30	7.91E+002	8.92E+002	8.41E+002	8.33E+002	2.77E+001

Table 6 continued

Func.	Dim.	Best	Worst	Mean	Median	Std
29	10	1.00E+002	2.23E+002	2.17E+002	2.22E+002	2.45E+001
	20	2.17E+002	2.49E+002	2.37E+002	2.43E+002	1.18E+001
	30	2.07E+002	7.62E+002	7.02E+002	7.18E+002	1.04E+002
30	10	4.44E+002	4.99E+002	4.66E+002	4.62E+002	1.23E+001
	20	4.94E+002	7.20E+002	5.79E+002	5.75E+002	6.82E+001
	30	6.37E+002	2.82E+003	1.59E+003	1.45E+003	6.60E+002

Table 7 Comparative study of algorithms DE, nnDE, jDE, nnjDE, and SaDE regarding the mean and standard deviation by dimension of the functions $D = 30$

Func.	Meas.	DE	nnDE	jDE	nnjDE	SaDE
f1	Mean	1.01E+005	7.36E+004	6.12E+004	8.94E+004	**3.73E+003**
	StDev	8.98E+004	4.48E+004	7.64E+004	6.70E+004	3.26E+003
f2	Mean	2.27E−015	2.27E−015	2.27E−015	**1.14E−015**	1.71E−014
	StDev	7.87E−015	7.87E−015	7.87E−015	5.68E−015	1.47E−014
f3	Mean	2.05E−014	**0.00E+000**	4.09E−014	2.27E−015	6.25E−014
	StDev	2.78E−014	0.00E+000	2.60E−014	1.14E−014	1.80E−014
f4	Mean	2.84E+000	4.57E+000	8.53E+000	2.51E+000	**1.53E−013**
	StDev	1.26E+001	1.54E+001	2.16E+001	1.17E+001	7.11E−014
f5	Mean	2.09E+001	2.00E+001	2.03E+001	**2.00E+001**	2.03E+001
	StDev	7.67E−002	2.81E−002	3.26E−002	1.47E−002	4.03E−002
f6	Mean	**4.12E+000**	6.18E+000	5.31E+000	6.90E+000	1.49E+001
	StDev	3.11E+000	3.33E+000	4.04E+000	3.50E+000	9.42E−001
f7	Mean	2.96E−004	6.90E−004	2.96E−004	1.77E−003	**9.09E−014**
	StDev	1.48E−003	2.42E−003	1.48E−003	5.25E−003	4.79E−014
f8	Mean	6.52E+001	2.74E+001	1.19E−001	2.48E+001	**1.02E−013**
	StDev	3.17E+001	7.99E+000	3.30E−001	8.47E+000	3.60E−014
f9	Mean	1.74E+002	4.08E+001	3.81E+001	**3.58E+001**	**3.58E+001**
	StDev	1.08E+001	1.49E+001	5.71E+000	9.11E+000	7.01E+000
f10	Mean	2.14E+003	7.70E+002	3.17E+000	8.55E+002	**1.11E+000**
	StDev	9.80E+002	3.03E+002	3.18E+000	4.02E+002	2.02E+000
f11	Mean	6.70E+003	2.74E+003	2.71E+003	2.76E+003	**2.28E+003**
	StDev	3.24E+002	5.99E+002	2.75E+002	5.20E+002	3.45E+002
f12	Mean	2.40E+000	**4.76E−002**	4.77E−001	5.47E−002	4.59E−001
	StDev	2.97E−001	2.81E−002	5.41E−002	3.15E−002	5.23E−002
f13	Mean	3.18E−001	**2.13E−001**	2.84E−001	2.47E−001	3.02E−001
	StDev	4.30E−002	4.66E−002	3.55E−002	6.40E−002	3.69E−002
f14	Mean	2.73E−001	**2.64E−001**	3.02E−001	2.88E−001	2.68E−001
	StDev	3.06E−002	4.02E−002	4.15E−002	9.92E−002	1.40E−001
f15	Mean	1.48E+001	**3.41E+000**	5.36E+000	3.74E+000	4.86E+000
	StDev	1.13E+000	1.03E+000	7.43E−001	1.23E+000	4.17E−001

Table 7 continued

Func.	Meas.	DE	nnDE	jDE	nnjDE	SaDE
f16	Mean	1.25E+001	1.08E+001	**1.03E+001**	1.08E+001	1.03E+001
	StDev	2.41E−001	6.59E−001	3.23E−001	7.18E−001	3.42E−001
f17	Mean	1.28E+003	**6.45E+002**	1.62E+003	7.09E+002	8.55E+002
	StDev	3.40E+002	3.05E+002	1.49E+003	3.88E+002	2.80E+002
f18	Mean	5.08E+001	**9.92E+000**	1.86E+001	1.33E+001	4.92E+001
	StDev	1.66E+001	3.31E+000	1.04E+001	6.20E+000	2.57E+001
f19	Mean	4.89E+000	**3.57E+000**	4.97E+000	3.99E+000	5.26E+000
	StDev	8.59E−001	1.15E+000	9.61E−001	1.05E+000	1.15E+000
f20	Mean	1.24E+001	**8.65E+000**	1.36E+001	9.19E+000	1.85E+001
	StDev	6.77E+000	3.39E+000	6.64E+000	3.09E+000	4.14E+000
f21	Mean	2.75E+002	**2.61E+002**	2.98E+002	3.47E+002	4.31E+002
	StDev	2.53E+002	2.09E+002	2.25E+002	1.95E+002	1.32E+002
f22	Mean	**1.21E+002**	3.12E+002	1.38E+002	3.57E+002	1.65E+002
	StDev	1.22E+002	1.65E+002	5.38E+001	2.00E+002	7.11E+001
f23	Mean	**3.15E+002**	**3.15E+002**	**3.15E+002**	**3.15E+002**	**3.15E+002**
	StDev	9.28E−014	4.67E−002	0.00E+000	4.10E−002	0.00E+000
f24	Mean	**2.22E+002**	2.25E+002	2.26E+002	2.25E+002	2.25E+002
	StDev	7.06E+000	3.62E+000	3.34E+000	1.96E+000	4.31E+000
f25	Mean	**2.03E+002**	2.03E+002	2.04E+002	2.03E+002	2.03E+002
	StDev	2.19E−001	2.87E−001	8.81E−001	3.48E−001	5.52E−001
f26	Mean	**1.00E+002**	**1.00E+002**	**1.00E+002**	**1.00E+002**	**1.00E+002**
	StDev	4.19E−002	4.99E−002	5.05E−002	6.76E−002	3.55E−002
f27	Mean	3.78E+002	3.78E+002	4.01E+002	**3.68E+002**	5.46E+002
	StDev	8.23E+001	5.15E+001	5.44E+001	4.02E+001	1.11E+002
f28	Mean	8.44E+002	8.41E+002	8.38E+002	8.62E+002	**8.08E+002**
	StDev	4.73E+001	2.77E+001	2.99E+001	4.54E+001	3.78E+001
f29	Mean	6.83E+005	**7.02E+002**	8.66E+002	1.36E+004	8.41E+005
	StDev	2.36E+006	1.04E+002	1.62E+002	4.91E+004	2.66E+006
f30	Mean	1.96E+003	**1.59E+003**	2.79E+003	1.84E+003	2.34E+003
	StDev	1.24E+003	6.60E+002	1.22E+003	1.05E+003	1.38E+003

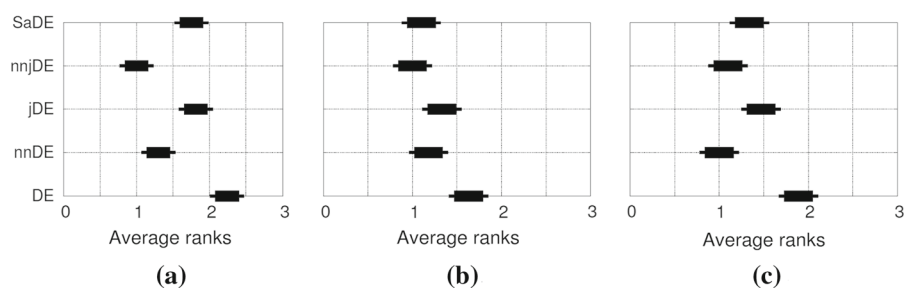
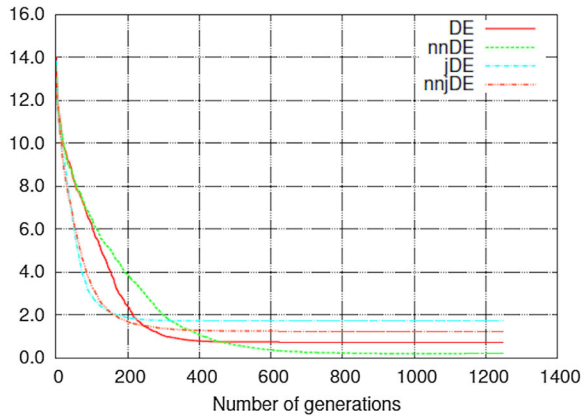
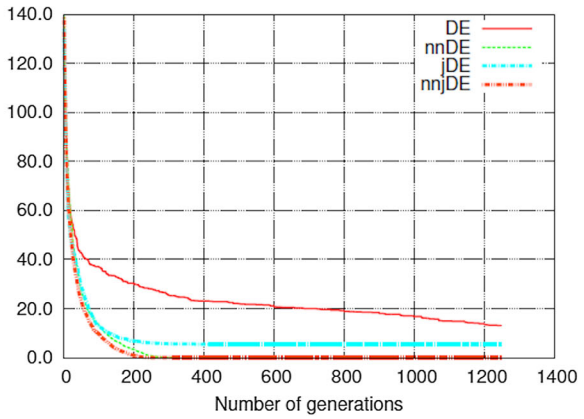


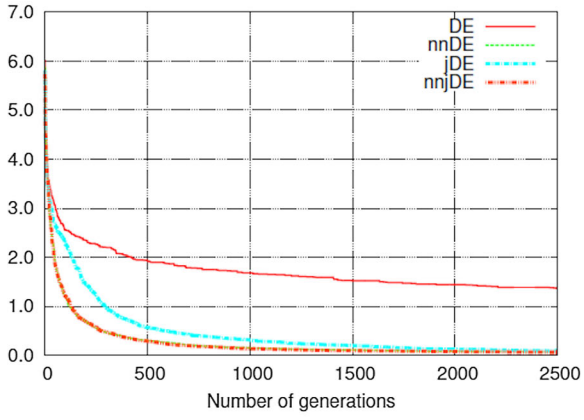
Fig. 4 Results of the Friedman nonparametric test, where each diagram illustrates the normalized average rank of the algorithms in test for the specified dimensions of the benchmark functions. The closer to one the value of the algorithm's rank, the more significant is the specific algorithm. **a** $D = 10$, **b** $D = 20$, **c** $D = 30$



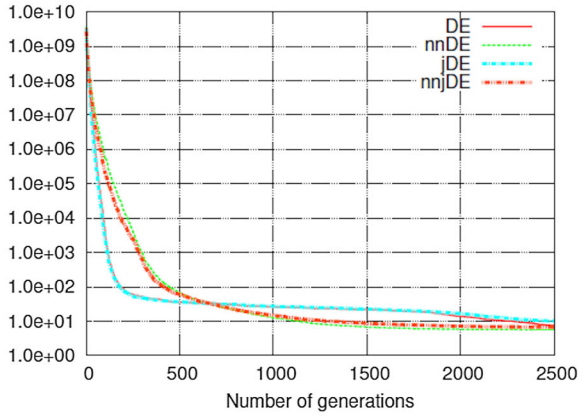
(a)



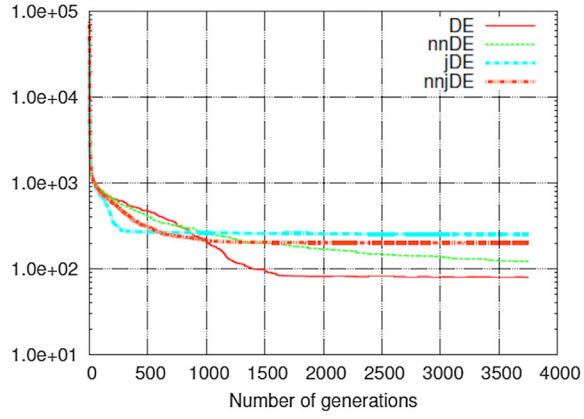
(b)



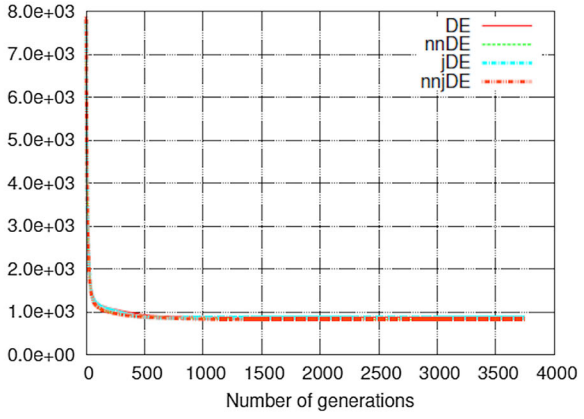
(c)



(d)



(e)



(f)

Fig. 5 Convergence graphs for six selected functions from the benchmark function suite. **a** $f_6\,(D = 10)$, **b** $f_8\,(D = 10)$, **c** $f_{12}\,(D = 20)$, **d** $f_{18}\,(D = 20)$, **e** $f_{22}\,(D = 30)$, **f** $f_{28}\,(D = 30)$

and SaDE obtained the same results for the function f_9. However, the statistical analysis takes into account also the minimum, maximum, median, and standard deviation values. This comparison is therefore more accurate. In summary, the nnDE is thus better for solving problems of higher dimensions (i.e., $D = 30$), while the nnjDE is better for solving the problems of lower dimensions (i.e., $D = 10$ and $D = 20$).

In order to evaluate the quality of the results statistically, Friedman tests [19] were conducted that compare the average ranks of the compared algorithms. Thus, a null hypothesis is placed that states: All algorithms are equivalent, and therefore, their ranks should be equal. When the null hypothesis is rejected, the Nemenyi post hoc test [11] is performed, where the critical difference is calculated between the average ranks for each algorithm.

Three Friedman tests were performed regarding the values of five measures obtained by optimizing 30 functions of three different dimensions. As a result, each algorithm in the tests was compared with respect to 150 variables. The tests were conducted at the significance level 0.05. The results of the Friedman nonparametric test can be seen in Fig. 4, which is divided into three diagrams. Each diagram shows the ranks and confidence intervals (critical differences) for the algorithms under consideration with regard to each problem dimensionality. Note that a significant difference between two algorithms is observed if their confidence intervals denoted as thickened lines in Fig. 4 do not overlap.

Figure 4a–c shows that the original DE algorithm was significantly outperformed by all other algorithms in the test for all problem dimensions. The nnjDE algorithm exhibits the best results in dimensions $D = 10$ (Fig. 4a) and $D = 20$ (Fig. 4b), while nnDE dominates the competitors for $D = 30$ (Fig. 4c). As demonstrated, the proposed local search heuristic significantly improves the results of both original DE and jDE algorithms, with the exception of the case nnjDE vs. jDE, where the advantage of nnjDE is not conclusive. Thereby, the assertion set at the beginning of the section has been successfully confirmed.

4.6 Convergence analysis

Convergence graphs were analyzed for functions f_6 and f_8 of dimension $D = 10$, functions f_{12} and f_{18} of dimension $D = 20$, and functions f_{22} and f_{28} of dimension $D = 30$. The best out of 25 optimization runs was analyzed. Convergence graphs are illustrated in Fig. 5 with two diagrams per problem dimension.

The following observations can be seen from these graphs:

- nnjDE outperforms the results of the original jDE for optimization of all presented functions, except f_{28},
- nnDE outperforms the results of the original DE for optimization of all presented functions, except f_{22} and f_{28},
- all algorithms achieved the similar results for optimization of the function f_{28}.

In summary, the presented results confirmed that hybridizing the original DE and jDE algorithms with the ANN regression can improve the results of both.

5 Conclusion

Recently, hybridizing the nature-inspired algorithms in order to expand its applicability and improve performance has become a popular trend in computational intelligence [15,16]. This paper proposes the hybridization of a DE algorithm with an ANN-based regression as a way to apply the local search heuristic. The ANN functions as a predictor of the best solution from a training set of trial vectors produced by an ensemble of DE strategies.

As a result, two hybrid DE algorithms were developed, i.e., nnDE representing the hybridization of the original DE algorithm with ANN, and nnjDE representing the hybridization of the original jDE algorithm with ANN. The results of experiments conducted on a CEC 2014 test suite consisting of 30 benchmark functions have shown that the proposed hybrids substantially outperform their original predecessors. Moreover, the performances gap broadened when the dimensionality of the problem was increased.

The experiments suggest that the quality of results highly depends on the value of parameter p_r, which determines the probability of local search execution. Lower values of p_r are generally required for higher problem dimensions.

These preliminary results advocate further investigation of the proposed hybridization in the future. As the first next step, however, we would like to expand

our comparative study to other well-known EAs and SI algorithms. Also, adaptation and self-adaptation of the parameter p_r seem very promising idea for the future.

Acknowledgments This research was supported by the Slovenian Research Agency (Grant P5-0027) and by the Deanship of Scientific Research, King Abdulaziz University (Grant 76-130-35-HiCi).

References

1. Adeyemo, J., Otieno, F.: Differential evolution algorithm for solving multi-objective crop planning model. Agric. Water Manag. **97**(6), 848–856 (2010)
2. Bhattacharya, A., Chattopadhyay, P.K.: Solving economic emission load dispatch problems using hybrid differential evolution. Appl. Soft Comput. **11**(2), 2526–2537 (2011)
3. Bigus, J.P.: Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support. McGraw-Hill, Inc., New York (1996)
4. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evolut. Comput. **10**(6), 646–657 (2006)
5. Cai, Y., Wang, J.: Differential evolution with neighborhood and direction information for numerical optimization. IEEE Trans. Cybern. **43**(6), 2202–2215 (2013)
6. Chakraborty, U.K., Das, S., Konar, A.: Differential evolution with local neighborhood. In: IEEE Congress on Evolutionary Computation, pp. 2042–2049. IEEE (2006)
7. Darwin, C.: On the Origin of Species. Harvard University Press, London (1859)
8. Das, S., Suganthan, P.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evolut. Comput. **15**(1), 4–31 (2011)
9. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Trans. Evolut. Comput. **13**(3), 526–553 (2009)
10. Datta, D., Dutta, S.: A binary-real-coded differential evolution for unit commitment problem. Int. J. Electr. Power Energy Syst. **42**(1), 517–524 (2012)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
12. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization problems. Comput. Oper. Res. **38**(12), 1877–1896 (2011)
13. Fan, H.Y., Lampinen, J.: A trigonometric mutation operation to differential evolution. J. Global Optim. **27**(1), 105–129 (2003)
14. Fister, I.J., Suganthan, P.N., Strnad, D., Brest, J., Fister, I.: Artificial neural network regression on ensemble strategies in differential evolution. In: MENDEL 2014, 20th International Conference on Soft Computing. University of Technology, Brno (2014)
15. Fister, I., Fister Jr, I., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evolut. Comput. **13**, 34–46 (2013)
16. Fister, I., Rauter, S., Yang, X.S., Ljubič, K., Fister Jr, I.: Planning the sports training sessions with the bat algorithm. Neurocomputing **149**(Part B), 993–1002 (2015)
17. Fister Jr, I., Fister, D., Fister, I.: Differential evolution strategies with random forest regression in the bat algorithm. In: Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion 2013, pp. 1703–1706. ACM (2013)
18. Fister Jr, I., Fister, D., Yang, X.S.: A hybrid bat algorithm. Elektroteh. vestnik **80**(1–2), 1–7 (2013)
19. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. **11**, 86–92 (1940)
20. Garro, B.A., Sossa, H., Vázquez, R.A.: Design of artificial neural networks using differential evolution algorithm. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds.) Neural Information Processing. Models and Applications, pp. 201–208. Springer, Heidelberg (2010)
21. Gershenson, C.: Artificial Neural Networks for Beginners (2003). arXiv:cs/0308031
22. Han, M.F., Liao, S.H., Chang, J.Y., Lin, C.T.: Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl. Intell. **39**(1), 41–56 (2013)
23. Hecht-Nielsen, R.: Theory of the backpropagation neural network. In: International Joint Conference on Neural Networks, pp. 593–605. IEEE (1989)
24. Holger, H., Thomas, S.: Stochastic Local Search: Foundations & Applications. Morgan Kaufman Inc., Amsterdam (2004)
25. Hozjan, T., Turk, G., Srpčič, S.: Fire analysis of steel frames with the use of artificial neural networks. J. Constr. Steel Res. **63**(10), 1396–1403 (2007)
26. J.J. Liang, B.Y.Q., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Technical Report, Zhengzhou University and Nanyang Technological University (2013)
27. Kartam, N., Flood, I., Garrett, J.H.: Artificial Neural Networks for Civil Engineers: Fundamentals and Applications. American Society of Civil Engineers, New York (1997)
28. Kourentzes, N., Barrow, D.K., Crone, S.F.: Neural network ensemble operators for time series forecasting. Expert Syst. Appl. **41**(9), 4235–4244 (2014)
29. LaTorre, A., Muelas, S., Peña, J.M.: A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. Soft Comput. **15**(11), 2187–2199 (2011)
30. Lee, S., Choeh, J.Y.: Predicting the helpfulness of online reviews using multilayer perceptron neural networks. Expert Syst. Appl. **41**(6), 3041–3046 (2014)
31. Lin, Y.C., Hwang, K.S., Wang, F.S.: Co-evolutionary hybrid differential evolution for mixed-integer optimization problems. Eng. Optim. **33**(6), 663–682 (2001)
32. Mallipeddi, R., Suganthan, P.: Differential evolution algorithm with ensemble of populations for global numerical optimization. Opsearch **46**(2), 184–213 (2009)
33. Mallipeddi, R., Mallipeddi, S., Suganthan, P.N.: Ensemble strategies with adaptive evolutionary programming. Inf. Sci. **180**(9), 1571–1581 (2010)

34. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. **11**(2), 1679–1696 (2011)
35. Mallipeddi, R., Suganthan, P.N.: Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) Swarm, Evolutionary, and Memetic Computing, pp. 71–78. Springer, Heidelberg (2010)
36. Mallipeddi, R., Suganthan, P.N.: Ensemble differential evolution algorithm for CEC2011 problems. In: IEEE Congress on Evolutionary Computation, pp. 1557–1564. IEEE (2011)
37. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943)
38. Mininno, E., Neri, F., Cupertino, F., Naso, D.: Compact differential evolution. IEEE Trans. Evolut. Comput. **15**(1), 32–54 (2011)
39. Neri, F., Iacca, G., Mininno, E.: Disturbed exploitation compact differential evolution for limited memory optimization problems. Inf. Sci. **181**(12), 2469–2487 (2011)
40. Neri, F., Mininno, E.: Memetic compact differential evolution for Cartesian robot control. IEEE Comput. Intell. Mag. **5**(2), 54–65 (2010)
41. Piotrowski, A.P.: Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. Inf. Sci. **241**, 164–194 (2013)
42. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791. IEEE (2005)
43. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Trans. Evolut. Comput. **12**(1), 64–79 (2008)
44. Rocca, P., Oliveri, G., Massa, A.: Differential evolution as applied to electromagnetics. IEEE Antennas Propag. Mag. **53**(1), 38–49 (2011)
45. Rojas, R.: Neutral Networks: A Systematic Introduction. Springer, Berlin (1996)
46. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, Englewood Cliffs (2010)
47. Santos, J., Diéguez, M.: Differential evolution for protein structure prediction using the HP model. In: Ferrández, J.M., Álvarez, J.R., de la Paz, F., Toledo, F.J. (eds.) Foundations on Natural and Artificial Computation, pp. 323–333. Springer, Heidelberg (2011)
48. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
49. Tvrdìk, J.: Competitive differential evolution. In: MENDEL 2006. 12th international conference on soft computing, pp. 7–12. University of Technology, Brno (2006)
50. Tvrdìk, J.: Differential evolution with competitive setting of its control parameters. TASK Q. **11**, 169–179 (2007)
51. Tvrdìk, J.: Adaptation in differential evolution: a numerical comparison. Appl. Soft Comput. **9**, 1149–1155 (2009)
52. Uyar, A.Ş., Türkay, B., Keleş, A.: A novel differential evolution application to short-term electrical power generation scheduling. Int. J. Electr. Power Energy Syst. **33**(6), 1236–1242 (2011)
53. Vrugt, J.A., Robinson, B.A., Hyman, J.M.: Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Trans. Evolut. Comput. **13**(2), 243–259 (2009)
54. Widrow, B., Rumelhart, D.E., Lehr, M.A.: Neural networks: applications in industry, business and science. Commun. ACM **37**(3), 93–105 (1994)
55. Zhang, W.J., Xie, X.F., et al.: DEPSO: hybrid particle swarm with differential evolution operator. IEEE Int. Conf. Syst. Man Cybern. **4**, 3816–3821 (2003)
56. Zobaa, A., Reljin, B.: Neural network applications in electrical engineering. Neurocomputing **70**(16–18), 2613–2614 (2007)