mathematics

MDPI

Article

Neural Network Algorithm with Dropout Using Elite Selection

Yong Wang, Kunzhao Wang and Gaige Wang *

School of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; wangyong@ouc.edu.cn (Y.W.); wangkunzhao@stu.ouc.edu.cn (K.W.)
* Correspondence: wgg@ouc.edu.cn

Abstract: A neural network algorithm is a meta-heuristic algorithm inspired by an artificial neural network, which has a strong global search ability and can be used to solve global optimization problems. However, a neural network algorithm sometimes shows the disadvantage of slow convergence speed when solving some complex problems. In order to improve the convergence speed, this paper proposes the neural network algorithm with dropout using elite selection. In the neural network algorithm with dropout using elite selection, the neural network algorithm is viewed from the perspective of an evolutionary algorithm. In the crossover phase, the dropout strategy in the neural network is introduced: a certain proportion of the individuals who do not perform well are dropped and they do not participate in the crossover process to ensure the outstanding performance of the population. Additionally, in the selection stage, a certain proportion of the individuals of the previous generation with the best performance are retained and directly enter the next generation. In order to verify the effectiveness of the improved strategy, the neural network algorithm with dropout using elite selection is used on 18 well-known benchmark functions. The experimental results show that the introduced dropout strategy improves the optimization performance of the neural network algorithm. Moreover, the neural network algorithm with dropout using elite selection is compared with other meta-heuristic algorithms to illustrate it is a powerful algorithm in solving optimization problems.

Keywords: neural network algorithm; meta-heuristics; artificial neural network; global optimization; dropout; elite selection

MSC: 92B20

Citation: Wang, Y.; Wang, K.; Wang, G. Neural Network Algorithm with Dropout Using Elite Selection. *Mathematics* **2022**, *10*, 1827. https://doi.org/10.3390/math10111827

Academic Editor: Ezequiel López-Rubio

Received: 26 April 2022
Accepted: 22 May 2022
Published: 26 May 2022

1. Introduction

Optimization algorithms are applied to many fields to obtain the optimal results to improve performance or reduce cost. Deterministic approaches need to use a large amount of gradient information and are highly dependent on the selected initial point, which is easy to fall into a local minimum [1–3]. However, meta-heuristic algorithms do not rely on gradient information and are not easy to fall into local optimization [3–5], which shows strong search-ability.

The meta-heuristic algorithm is the product of the combination of a random algorithm and a local search algorithm. It mainly solves the global optimization problem by simulating the evolution law of nature or the wisdom of the group [6,7]. To a certain extent, it can search globally and find the approximate solution of the optimal solution. The process of a meta-heuristic algorithm is mainly divided into the following steps [8,9]. (1) Randomly generate candidate solutions as initial values. (2) Calculate the objective function values of the initial values. (3) According to the existing information, update the candidate solutions by crossover, mutation, and other methods to generate a new generation of candidate solutions. (4) The new candidate solutions enter the next iteration until the shutdown criterion is met. It is an iterative generation process. Through random initialization and crossover

and mutation of candidate solutions, the exploration and development of the whole search space can be realized, and the optimal solution can be gradually searched [10,11].

Classical meta-heuristic algorithms include genetic algorithm (GA) [12], simulated annealing (SA) [13], particle swarm optimization (PSO) [14–16], harmony search (HS) [17], differential evolution (DE) [18,19], ant colony optimization (ACO) [20], and artificial bee colony optimization (ABC) [21]. These algorithms follow the principle of a meta-heuristic algorithm. For example, in a genetic algorithm, each independent variable is represented by a gene, and each individual is represented by a chromosome. Starting from an initial population, new chromosomes are generated through the process of chromosome crossover and mutation. After calculating the fitness, the individuals with poor fitness are eliminated, so as to promote the population evolution to produce better and better approximate solutions.

No meta-heuristic algorithm can be suitable for all types of optimization problems, so new meta-heuristic algorithms are constantly proposed, such as the neural network algorithm (NNA) [22], spotted hyena optimizer (SHO) [23], seagull optimization algorithm (SOA) [24], tunicate swarm algorithm (TSA) [25], elephant herding optimization (EHO) [26], sooty tern optimization algorithm (STOA) [27], chaotic neural network algorithm with competitive learning (CCLNNA) [28], monarch butterfly optimization (MBO) [29,30], earthworm optimization algorithm (EWA) [31], and moth search algorithm (MSA) [32]. Furthermore, in the literature [33], the Forest Optimization Algorithm (FOA) is proposed, inspired by nature's process in the forest and it shows quite good accuracy compared with GA and PSO on the path generating four-bar mechanism in [34]. Additionally, the virus optimization algorithm (VOA) is an iteratively population-based algorithm inspired by the behavior of viruses attacking a living cell [35] and it is applied to the identification of elastoplastic properties of materials [36]. All the algorithms provide new solutions for solving different types of optimization problems.

Based on NNA, several improved algorithms are proposed and they are also applied to the practical problems. NNA is applied to improve the overall competitiveness of the single mixed refrigerant (SMR) process for synthetic natural gas (SNG) liquefaction, which saves energy and cost [37] and it is used to optimize parameters of the Fractional-Order-Proportional-Integral-Derivative (FOPID) controller [38]. An effective hybrid method named TLNNA, based on teaching–learning-based optimization (TLBO) is proposed to solve engineering optimization problems [39]. Grey wolf optimization with neural network algorithm (GNNA) is proposed by combining the improved grey wolf optimizer (GWO) and NNA, which significantly improves the performance [40]. A modified neural network algorithm (M-NNA) is adopted as the optimization algorithm in the Complex Fracture Network (CFN) optimization framework, with an optimization searching accuracy far better than the original algorithm [41]. A new methodology based on the combination of symbiosis organism search (SOS) and NNA is proposed for the optimal planning and operation of distributed generations (DGs) and capacitor banks (CBs) in the radial distribution networks (RDNs), with the results obtained helping to improve the annual energy loss mitigation and cost savings [42]. In the literature [43], a quasi-oppositional chaotic neural network algorithm (QOCNNA) is developed by combining NNA with chaotic local search (CLS) and quasi-oppositional-based learning (QOBL) approaches and it is more effective in improving the performance of RDNs.

This paper proposes a global optimization algorithm called the neural network algorithm with dropout using elite selection (DESNNA), which is a variant of NNA. NNA is an algorithm inspired by an artificial neural network. In NNA, each individual is regarded as a pattern solution. Firstly, the initial pattern solution and initial coefficient matrix are generated, and the coefficient matrix is applied to all pattern solutions each time, which is equivalent to the crossover operation between pattern solutions. However, the convergence speed of NNA is slow and sometimes falls into local optimization in complex situations. Therefore, the DESNNA is proposed to improve the convergence speed and performance of NNA. The main contributions of this paper are as follows:

(1) NNA is analyzed from the perspective of an evolutionary algorithm, including crossover, mutation, and selection processes, which correspond to every step of NNA. It shows that NNA belongs to an evolutionary algorithm.
(2) In the crossover stage of the DESNNA, similar to dropout in the neural network, the dropout strategy is applied to NNA: a certain proportion of the individuals are dropped and do not participate in the crossover process, which ensures the superiority of the individuals participating in the crossover process.
(3) In the selection process of the DESNNA, some individuals who performed well in the previous generation are directly retained when updating the population, which increases the optimization ability of the algorithm without losing the diversity of the population.

The rest of this paper is organized as follows. The neural network algorithm is introduced in Section 2. The proposed DESNNA is introduced in detail in Section 3. The experiment and results of the DESNNA on the benchmark functions are presented in Section 4 and the conclusion and future work are stated in Section 5.

2. Neural Network Algorithm

2.1. Artificial Neural Network

The artificial neural network (ANN) is a complex structure based on biological neurons. ANN consists of neurons, which are simple processing units and weighted connections between these neurons. A typical structure is a multilayer perceptron (MLP), as shown in Figure 1. ANN receives a data set, starts the training process, and adjusts the connection weights between neurons [44]. The artificial neural network with dropout makes the activation value of a neuron stop working with a certain probability during the forward propagation, which can make the model more generalized because it does not rely too much on some local features [45–47]. The artificial neural network with dropout is shown in Figure 2, in which the dotted circle represents the dropped neurons. ANN has many advantages in the fields of medicine, robot, image processing, and so on [48–51]. The NNA draws on the idea of ANN forward propagation and its weight matrix is updated in each iteration [52,53].

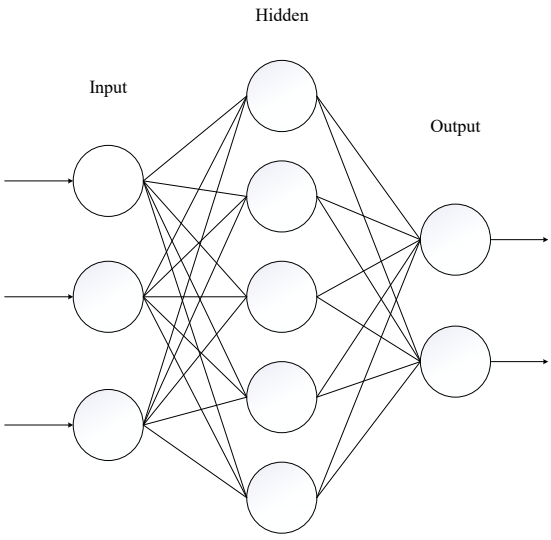

Figure 1. Structure of an Artificial Neural Network.

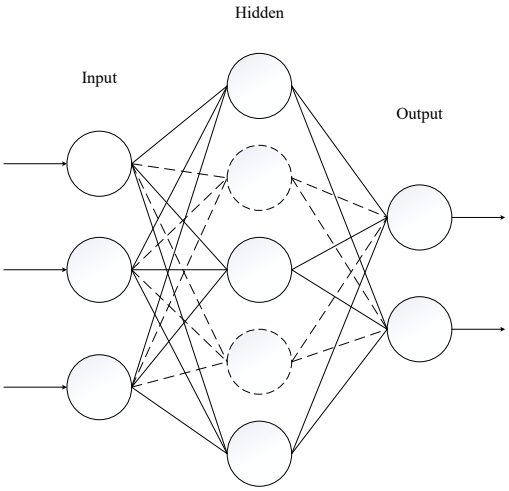

Figure 2. Structure of an Artificial Neural Network with dropout.

2.2. The Introduction of Neural Network Algorithm

According to the NNA [22], a pattern solution is an array of $1 \times D$ defined as $x = [x_1, x_2, \cdots, x_D]$. The population of pattern solutions is a matrix with a size $N \times D$, which can be defined as

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \cdots & x_D^N \end{bmatrix}. \tag{1}$$

The cost of each pattern solution can be obtained by a fitness function. For example, the cost of the ith pattern solution is

$$C_i = f(x_1^i, x_2^i, \cdots, x_D^i). \tag{2}$$

In NNA, weights are a square matrix with the size $N \times N$, defined as

$$W = [W_1, W_2, \cdots, W_{N_{pop}}] = \begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^N \\ w_2^1 & w_2^2 & \cdots & w_2^N \\ \vdots & \vdots & \vdots & \vdots \\ w_N^1 & w_N^2 & \cdots & w_N^N \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{N1} \\ w_{12} & w_{22} & \cdots & w_{N2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1N} & w_{2N} & \cdots & w_{NN} \end{bmatrix}. \tag{3}$$

NNA is described in two stages as the following.

(1) Initialization stage

Firstly, the number of the pattern solutions (N) and the maximum number of iterations are set. Then the initial population containing N pattern solutions is randomly generated between LB and UB. The cost of each individual in the population is obtained by the fitness function. The weight matrix is generated randomly, which satisfies the constraints:

$$\sum_{j=1}^{N} w_{ij} = 1, \ i = 1, 2, \cdots, N. \tag{4}$$

$$w_{ij} \in U(0, 1), \ i, j = 1, 2, \cdots, N. \tag{5}$$

According to the cost, the target solution (X^{Target}) and the corresponding target weight (W^{Target}) should be set.

(2) Cycle stage

Similar to the crossover process, the weight matrix multiplying pattern solutions generates the new pattern solutions using the following equation:

$$X_j^{New}(t+1) = \sum_{i=1}^{N} w_{ij}(t) \times X_i(t),\ j = 1, 2, \cdots, N. \tag{6}$$

$$X_i(t+1) = X_i(t) + X_i^{New}(t+1),\ i = 1, 2, \cdots, N. \tag{7}$$

where t is an iteration index. Then the weight matrix should be updated, following Equation (8):

$$W_i(t+1) = W_i(t) + 2 \times rand \times (W^{Target}(t) - W_i(t)),\ i = 1, 2, \cdots, N. \tag{8}$$

where the weight matrix (W) should always satisfy the constraints (4) and (5).

After the updating process, according to the modification factor β, check the bias condition and choose to either perform the bias operator according to Equations (9) and (10) or perform the transfer function operator by Equation (10). The Equation (9) is given below:

$$X(i, j) = LB + (UB - LB) \times rand,\ i = 1, 2, \cdots, N_b. \tag{9}$$

where N_b = round($D \times \beta$) is the number of biased variables in the population of the new pattern solution and j is a random integer between 0 and D. The Equation (10) is given below:

$$W(i, j) = U(0, 1),\ i = 1, 2, \cdots, N_{wb}. \tag{10}$$

where N_{wb} = round($N \times \beta$) is the number of biased variables in the updated weight matrix. The transfer function operator makes new pattern solutions transfer from their positions to another position to generate better solutions. The transfer function operator on pattern solutions is defined as:

$$X_i^*(t+1) = X_i(t+1) + 2 \times rand \times (X^{Target}(t) - X_i(t)),\ i = 1, 2, \cdots, N. \tag{11}$$

The bias operator is similar to the mutation operator, which can prevent premature convergence.

The cost of every pattern solution for the population is calculated and the minimum is chosen as the optimal value. The target solution (X^{Target}) and the target weight (W^{Target}) corresponding to the optimal value should be updated. Finally, β is reduced according to Equation (12):

$$\beta(t+1) = 0.99 \times \beta(t),\ t = 1, 2, \cdots, Max\_iteration. \tag{12}$$

If the stopping condition is satisfied, the NNA will stop. Otherwise, go back to the beginning of the cycle stage. The process of NNA is as following Algorithm 1:

Algorithm 1. The implementation of the neural network algorithm (NNA).

```
Create random initial population X and weights W with constraints by Equations (4) and (5)
Calculate the cost of every pattern solution and set the target solution and target weight
For i = 1:max_iteration
    Generate new pattern solutions X_{t+1} by Equations (6) and (7)
    Update the weights by Equation (8)
        If rand ≤ β
            Perform the bias operator for pattern solutions X_{t+1} and weights W_{t+1} by Equations (9) and (10)
        Else
            Perform the transfer function operator on X_{t+1} by Equation (11)
        End if
    Calculate the cost of every pattern solution and find the optimal solution and weight
    Reduce the modification factor β by Equation (12)
End for
```

3. The Neural Network Algorithm with Dropout Using Elite Selection

In order to improve the convergence performance of NNA, the DESNNA is proposed. This section is divided into three subsections, including viewing NNA from the perspective of an evolutionary algorithm, the introduced dropout strategy in the DESNNA, and the elite selection in the DESNNA.

3.1. NNA from the Perspective of Evolutionary Algorithm

Firstly, we view NNA from the perspective of an evolutionary algorithm. The main steps of an evolutionary algorithm include initialization, crossover, mutation, and selection. In NNA, each pattern solution is viewed as an individual of the population and, therefore, the pattern solution matrix is seen as a population.

Similar to the crossover process, when generating new pattern solutions, the weight matrix multiplies pattern solutions by Equation (6), which ties each pattern solution together. For instance, four pattern solutions generating the first new pattern solution can be expressed as:

$$X_1^{New}(t+1) = w_{11}X_1(t) + w_{21}X_2(t) + w_{31}X_3(t) + w_{41}X_4(t). \tag{13}$$

What can be seen from Equation (13) is that when the new pattern solution is generated, the process of linear combination between individuals using the values of the weight matrix is regarded as a crossover process. Figure 3 presents how NNA generates its new population of pattern solutions.

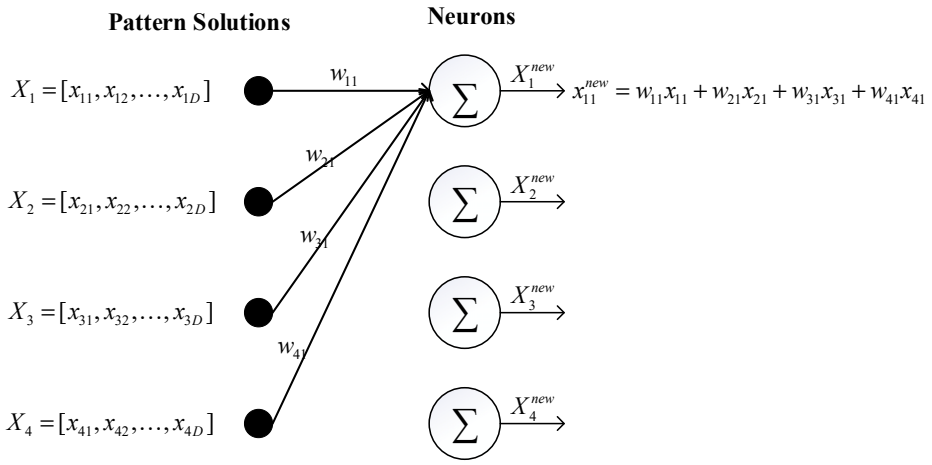

Figure 3. Schematic view of generating new pattern solutions.

Resembling the mutation process, after checking the bias condition, the bias operator for the new pattern solution or updated weight matrix by Equation (9) or Equation (10) and the transfer function operator for pattern solution by Equation (11) perform random offset operation, which is regarded as a mutation process. Similar to the selection process, after calculating the cost of each pattern solution, the minimum is chosen as the optimal value. In conclusion, the whole process of NNA corresponds to the framework of the evolutionary algorithm.

3.2. The Introduced Dropout Strategy in the DESNNA

In order to overcome the low convergence speed problem, the idea of dropout in the neural network is applied to the crossover process. The principle of dropout in deep learning is to set a probability when samples are input into the neural network for training so that each neuron has a certain probability of death and does not participate in the network training. Dropout is similar to sexual reproduction in biological evolution. The power of genes lies in the ability to mix rather than the ability of a single gene. Sexual

reproduction can not only pass down excellent genes but also reduce the joint adaptability between genes.

Inspired by dropout in the neural network, in the crossover process of the DESNNA, when calculating and generating new pattern solutions, the linear combination of different individuals in the population confirms new pattern solutions. Then, 10% of individuals who do not perform well are dropped and do not participate in the process. In order to achieve this, the pattern solutions that do not participate in the crossover process are set to zero. The implementation of concrete details is reflected in the following Equations (14) and (15):

$$X_{drop}(t) = 0. \tag{14}$$

$$X_j^{New}(t+1) = \sum_{i=1}^{N} w_{ij}(t) \times X_i(t),\ j = 1, 2, \cdots, N. \tag{15}$$

where X_1, X_2, ... , X_N includes 10% of X_{drop}. Therefore, after applying the dropout strategy to NNA, the crossover process is improved from Equation (6) to Equations (14) and (15). In this way, the individuals with poor performance corresponding to the pattern solutions are set to zero, which is equivalent to them not participating in the crossover process. This ensures the superiority of individuals in the crossover process and then ensures the superiority of the whole population. Consequently, introducing the dropout strategy can improve the convergence speed of the algorithm.

3.3. The Elite Selection in the DESNNA

To improve the convergence speed, the elite selection strategy is applied to the selection process. When calculating and generating new pattern solutions, the linear combination of different individuals in the population confirms new pattern solutions. Adding new pattern solutions to the old pattern solutions obtains a new population. Then the cost of every pattern solution of the new population is calculated, and then they are sorted. In the selection process, the top 15% of individuals with the highest fitness are saved for the next generation. Therefore, some of the better individuals are preserved in every evolutionary process, which makes the level of the whole population higher. In this way, the convergence performance of the algorithm is better. Meanwhile, the bias operator and the transfer function operator are performed as usual without losing the diversity of the population.

The process of the DESNNA is as following Algorithm 2:

Algorithm 2. The implementation of the neural network algorithm with dropout using elite selection (DES-NNA).

Create random initial population X and weights W with constraints by Equations (4) and (5)
Calculate the cost of every pattern solution and set the target solution and target weight
For i = 1:*max_iteration*
    10% of individuals with the worst fitness corresponding pattern solution X_{worst} is set 0
Generate new pattern solutions X_{t+1} by Equations (14), (15) and (7)
    Update the weights by Equation (8)
        If rand $\leq \beta$
            Perform the bias operator for pattern solutions X_{t+1} and weights W_{t+1} by Equations (9) and (10)
        Else
            Perform the transfer function operator on X_{t+1} by Equation (11)
        End if
    Calculate the cost of every pattern solution and find the optimal solution and weight
    Sort the cost of each pattern solution in the new population
    Save the top 15% of individuals with the highest fitness to the next generation
    Reduce the modification factor β by Equation (12)
End for

4. DESNNA for Global Optimization

In order to verify the performance of the DESNNA in solving numerical optimization, 18 benchmark functions extracted from the literature [54] are used for the experiment. The experimental results from the DESNNA are compared with those from NNA, which reflects the effectiveness of our improved strategy. Additionally, the results are compared with other meta-heuristics optimization algorithms.

4.1. Benchmark Functions

The test functions provided in the literature [54,55] have been applied to the experimental research of meta-heuristic algorithms. The definitions of benchmark functions F1 to F11 are shown in Table 1 and the definitions of hybrid composition functions F12 to F18 are presented in Table 2, which have higher complexity compared to the functions F1 to F11. The detailed procedure used to hybridize the first function with the second function is shown in the literature [54]. The properties of the benchmark functions are shown in Table 3, and the optimal solutions of all benchmark functions are known.

Table 1. Benchmark functions F1 to F11.

Function	Name	Definition
F1	Shifted Sphere Function	$\sum_{i=1}^{D} z_i^2 + f\_bias, z = x - o$
F2	Shifted Schwefel Problem 2.21	$\max_i\{\|z_i\|, 1 \le i \le D\} + f\_bias, z = x - o$
F3	Shifted Rosenbrock's Function	$\sum_{i=1}^{D-1} (100(z_i^2 + z_{i+1})^2 + (z_i - 1)^2) + f\_bias, z = x - o$
F4	Shifted Rastrigin's Function	$\sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias, z = x - o$
F5	Shifted Griewank's Function	$\sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias, z = x - o$
F6	Shifted Ackley's Function	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2} - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i) + 20 + e + f\_bias$
F7	Schwefel's Problem 2.22	$\sum_{i=1}^{D} \|x_i\| + \prod_{i=1}^{D} \|x_i\|$
F8	Schwefel's Problem 1.2	$\sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$
F9	Extended f10	$\left(\sum_{i=1}^{D-1} f_{10}(x_i, x_{i+1})\right) + f_{10}(x_D, x_1)$
		$f_{10} = (x^2 + y^2)^{0.25} \cdot (\sin^2(50 \cdot (x^2 + y^2)^{0.1}) + 1)$
F10	Bohachevsky	$\sum_{i=1}^{D} (x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7$
F11	Schaffffer	$\sum_{i=1}^{D} (x_i^2 + x_{i+1}^2)^{0.25} (\sin^2(50 \cdot (x_i^2 + x_{i+1}^2)^{0.1}) + 1)$

Table 2. Hybrid composition functions.

Function	First Function	Second Function	Weight Factor
F12	F9	F1	0.25
F13	F9	F4	0.25
F14	F5	F1	0.5
F15	F3	F4	0.5
F16	F9	F1	0.75
F17	F9	F3	0.75
F18	F9	F4	0.75

Table 3. Properties of F1 to F18.

Function	Range	Optimum	Unimodal/ Multimodal	Separable	Shifted	*f_bias*
F1	$[-100, 100]^D$	0	U	Y	Y	−450
F2	$[-100, 100]^D$	0	U	N	Y	−450
F3	$[-100, 100]^D$	0	M	Y	Y	390
F4	$[-5, 5]^D$	0	M	Y	Y	−330
F5	$[-600, 600]^D$	0	M	N	Y	−180
F6	$[-32, 32]^D$	0	M	Y	Y	−140
F7	$[-10, 10]^D$	0	U	Y	N	–
F8	$[-65.536, 65.536]^D$	0	U	N	N	–
F9	$[-100, 100]^D$	0	U	N	N	–
F10	$[-15, 15]^D$	0	U	Y	N	–
F11	$[-100, 100]^D$	0	U	Y	N	–
F12	$[-100, 100]^D$	0	U	N	Y	−450
F13	$[-5, 5]^D$	0	M	N	Y	−330
F14	$[-100, 100]^D$	0	U	N	Y	−630
F15	$[-10, 10]^D$	0	M	Y	Y	60
F16	$[-100, 100]^D$	0	U	N	Y	−450
F17	$[-100, 100]^D$	0	M	N	Y	390
F18	$[-5, 5]^D$	0	M	N	Y	−330

In order to verify the performance of the DESNNA, we compare the DESNNA with the other six algorithms on these functions, including the NNA, CCLNNA, TSA, PSO, GA, and HS. The parameters of these applied algorithms are listed in Table 4. For a fair comparison, the dimension of the benchmark function is set to 50 in this experiment. The maximum number of function evaluations (NFES) is used as the shutdown condition, which is set to 5000 times the dimension (D). Each algorithm runs independently of benchmark functions 30 times, taking the best error, the average error, the worst error, and the error standard deviation. The population size is uniformly set to 50. All the optimizers in this paper are coded in MATLAB R2019b.

Table 4. Optimal values of user parameters used in the reported optimizers.

Methods	Parameters	Optimal Values
GA	N	50
	P_c	0.8
	P_m	0.2
PSO	N	50
	C_1, C_2	2
	w	0.9
HS	N	50
	$HMCR$	0.95
	PAR	0.3
CCLNNA	N	50
N	N	50
N	N	50
	$rateOfSelect$	0.15
	$rateOfDropt$	0.10

4.2. Comparison between Improved DESNNA and NNA

In order to verify the effectiveness of the improvement strategy, this section focuses on comparing optimization results between the NNA and DESNNA. The experimental results of the NNA and DESNNA running on benchmark functions are shown in Table 5. The better results are highlighted in bold type. From Table 5, in terms of the best error, the average error, the worst error, and the error standard deviation, the DESNNA is superior to the NNA on all 18 benchmark functions. That is to say, the DESNNA has better search-ability and stability than the NNA, which means the applied improvement strategy improves the performance of the NNA.

Table 5. Experimental results obtained by the NNA and DESNNA.

Function	Methods	Best Error	Average Error	Worst Error	Error Standard Deviation
F1	NNA	5.684×10^{-14}	5.684×10^{-14}	3.411×10^{-13}	1.339×10^{-13}
	DESNNA	**0**	**0**	$\mathbf{1.137 \times 10^{-13}}$	$\mathbf{5.971 \times 10^{-14}}$
F2	NNA	7.209×10^{-2}	3.251×10^{-1}	1.425×10^{0}	2.901×10^{-1}
	DESNNA	$\mathbf{2.626 \times 10^{-2}}$	$\mathbf{3.152 \times 10^{-1}}$	$\mathbf{1.010 \times 10^{0}}$	$\mathbf{2.389 \times 10^{-1}}$
F3	NNA	4.822×10^{1}	5.860×10^{1}	2.094×10^{2}	3.953×10^{1}
	DESNNA	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{3.493 \times 10^{-13}}$
F4	NNA	4.547×10^{-13}	1.825×10^{0}	7.960×10^{0}	2.653×10^{0}
	DESNNA	**0**	$\mathbf{5.978 \times 10^{-1}}$	$\mathbf{4.975 \times 10^{0}}$	$\mathbf{1.470 \times 10^{0}}$
F5	NNA	5.684×10^{-14}	5.754×10^{-4}	9.865×10^{-3}	2.213×10^{-3}
	DESNNA	**0**	**0**	$\mathbf{5.684 \times 10^{-14}}$	$\mathbf{3.077 \times 10^{-14}}$
F6	NNA	2.154×10^{-11}	7.012×10^{-11}	1.863×10^{-10}	3.857×10^{-11}
	DESNNA	$\mathbf{5.684 \times 10^{-14}}$	$\mathbf{2.842 \times 10^{-14}}$	$\mathbf{1.705 \times 10^{-13}}$	$\mathbf{5.853 \times 10^{-14}}$
F7	NNA	3.132×10^{-12}	3.177×10^{-11}	1.101×10^{-10}	3.063×10^{-11}
	DESNNA	$\mathbf{6.335 \times 10^{-15}}$	$\mathbf{6.297 \times 10^{-14}}$	$\mathbf{5.688 \times 10^{-13}}$	$\mathbf{1.072 \times 10^{-13}}$
F8	NNA	9.448×10^{-4}	5.364×10^{-3}	2.151×10^{-2}	5.002×10^{-3}
	DESNNA	$\mathbf{7.039 \times 10^{-5}}$	$\mathbf{1.794 \times 10^{-3}}$	$\mathbf{1.380 \times 10^{-2}}$	$\mathbf{2.701 \times 10^{-3}}$
F9	NNA	3.105×10^{-1}	4.330×10^{0}	2.096×10^{1}	4.678×10^{0}
	DESNNA	$\mathbf{1.418 \times 10^{-2}}$	$\mathbf{3.460 \times 10^{0}}$	$\mathbf{1.369 \times 10^{1}}$	$\mathbf{4.206 \times 10^{0}}$
F10	NNA	0	3.701×10^{-17}	2.220×10^{-16}	8.417×10^{-17}
	DESNNA	**0**	**0**	**0**	**0**
F11	NNA	2.143×10^{-1}	5.928×10^{0}	2.416×10^{1}	6.384×10^{0}
	DESNNA	$\mathbf{1.787 \times 10^{-2}}$	$\mathbf{2.126 \times 10^{0}}$	$\mathbf{1.356 \times 10^{1}}$	$\mathbf{3.941 \times 10^{0}}$
F12	NNA	4.151×10^{-5}	2.217×10^{-2}	6.596×10^{-1}	1.204×10^{-1}
	DESNNA	$\mathbf{5.449 \times 10^{-10}}$	$\mathbf{1.710 \times 10^{-8}}$	$\mathbf{8.094 \times 10^{-8}}$	$\mathbf{1.962 \times 10^{-8}}$
F13	NNA	1.906×10^{-5}	1.727×10^{0}	9.720×10^{0}	2.649×10^{0}
	DESNNA	$\mathbf{2.314 \times 10^{-8}}$	$\mathbf{1.017 \times 10^{0}}$	$\mathbf{7.469 \times 10^{0}}$	$\mathbf{2.367 \times 10^{0}}$
F14	NNA	2.274×10^{-13}	8.102×10^{-3}	2.061×10^{-1}	3.772×10^{-2}
	DESNNA	**0**	$\mathbf{3.196 \times 10^{-3}}$	$\mathbf{5.157 \times 10^{-2}}$	$\mathbf{1.120 \times 10^{-2}}$
F15	NNA	2.346×10^{1}	2.356×10^{1}	2.645×10^{1}	5.449×10^{-1}
	DESNNA	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{5.357 \times 10^{-10}}$
F16	NNA	3.345×10^{-2}	2.657×10^{0}	8.910×10^{0}	2.854×10^{0}
	DESNNA	$\mathbf{1.699 \times 10^{-5}}$	$\mathbf{2.869 \times 10^{-1}}$	$\mathbf{4.255 \times 10^{0}}$	$\mathbf{8.419 \times 10^{-1}}$
F17	NNA	1.096×10^{1}	4.957×10^{1}	2.714×10^{2}	5.833×10^{1}
	DESNNA	$\mathbf{1.061 \times 10^{1}}$	$\mathbf{2.680 \times 10^{1}}$	$\mathbf{1.434 \times 10^{2}}$	$\mathbf{3.733 \times 10^{1}}$
F18	NNA	1.152×10^{-2}	2.101×10^{0}	1.293×10^{1}	3.218×10^{0}
	DESNNA	$\mathbf{7.856 \times 10^{-6}}$	$\mathbf{1.070 \times 10^{0}}$	$\mathbf{1.092 \times 10^{1}}$	$\mathbf{2.881 \times 10^{0}}$

In addition, the convergence performance between the DESNNA and NNA is compared. Several typical curves of the convergence process on the functions F2, F9, F11, and F14 are provided in Figure 4 to show the convergence performance. From Figure 4, the DESNNA converges faster than the NNA on functions. It can be seen from the characteristics of these curves that in the initial stage, the DESNNA has better convergence performance than the NNA and in fewer iterations, the DESNNA tends to find the value closer to the optimal value. With the increase of iterations, the DESNNA tends to converge to the optimal value. Therefore, the DESNNA has better convergence performance compared to the NNA.

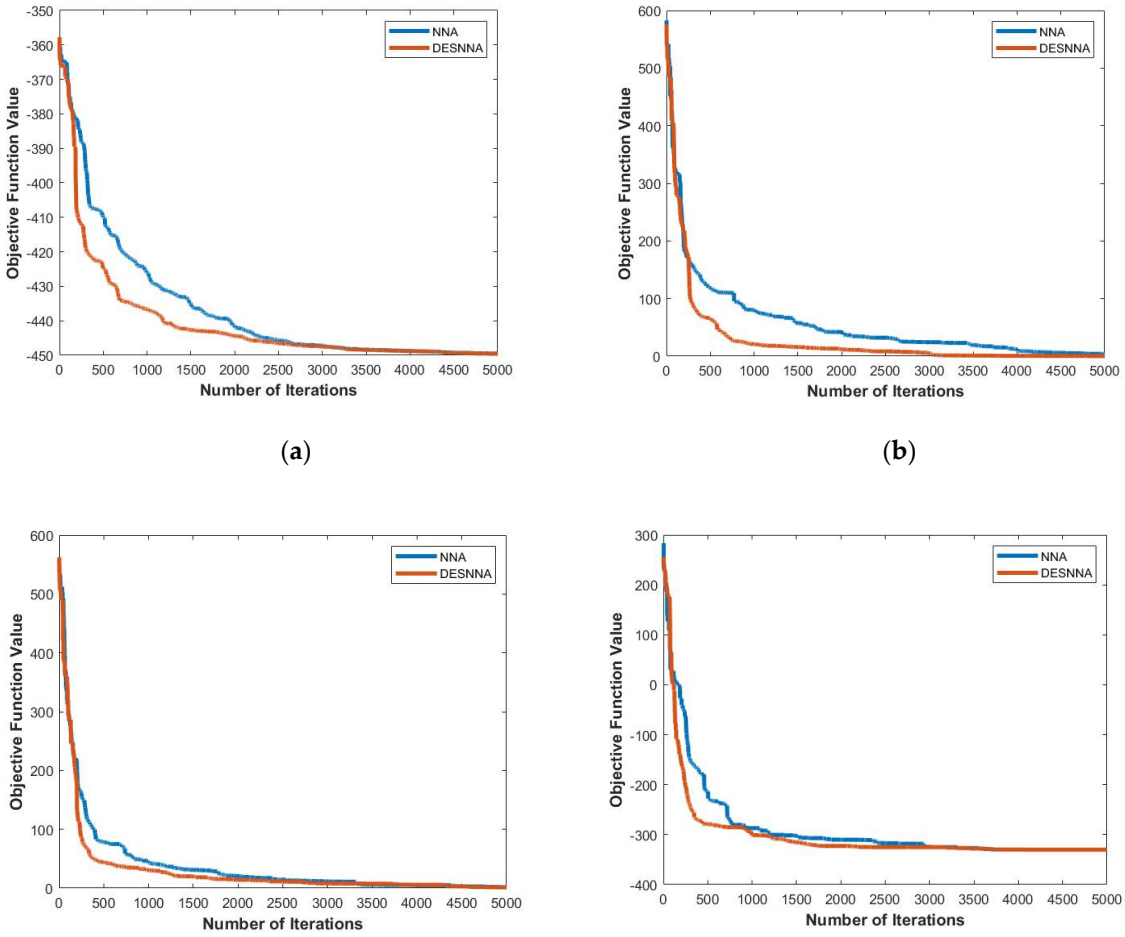

Figure 4. Several typical convergence curves obtained by the NNA and DESNNA. (**a**) F2. (**b**) F9. (**c**) F11. (**d**) F14.

4.3. Comparisons between the Improved DESNNA and Other Algorithms

The optimization performance between the DESNNA and five other algorithms containing the CCLNNA, TSA, PSO, GA, and HS are compared in this section. Table 6 shows the experimental results obtained by the DESNNA and other methods.

From Table 6, in terms of the best error, the DESNNA outperforms CCLNNA on all functions except for F2. The DESNNA is superior to the TSA on functions F1, F3, F4, F5, F6, F9, F10, F11, F13, F14, F15, F16, F17, and F18. The DESNNA beats the PSO, GA, and HS on all functions. From Table 6, in terms of the average error, the DESNNA performs better than the CCLNNA on all functions except for F2, F4, and F13. TSA beats the DESNNA only on functions F2, F7, and F8. The DESNNA is superior to the PSO, GA, and HS on all functions. As for the worst error, the DESNNA outperforms CCLNNA on functions F1, F3, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, and F16 and outperforms TSA on functions F1, F3, F4, F5, F6, F9, F10, F11, F12, F13, F14, F15, F16, F17, and F18. The DESNNA is superior to the PSO, GA, and HS on all functions. In terms of error standard deviation, the DESNNA outperforms CCLNNA on functions F1, F3, F5, F6, F7, F8, F9, F10, F11, F12, F14, F15, and F16. TSA beats the DESNNA only on functions F1, F2, F7, and F8, and GA beats the DESNNA only on F17. The DESNNA is superior to the PSO and HS on all functions. Clearly, the DESNNA shows better performance than other compared methods.

Table 6. Experimental results obtained by the DESNNA and other methods.

Methods	Best Error	Average Error	Worst Error	Error Standard Deviation
		F1		
DESNNA	**0**	**0**	$\mathbf{1.137 \times 10^{-13}}$	5.971×10^{-14}
CCLNNA	7.135×10^{-9}	3.697×10^{-8}	1.215×10^{-7}	2.172×10^{-8}
TSA	5.684×10^{-14}	5.684×10^{-14}	1.137×10^{-13}	$\mathbf{5.382 \times 10^{-14}}$
PSO	1.137×10^{-13}	1.137×10^{-13}	5.116×10^{-13}	2.635×10^{-13}
GA	1.503×10^{-8}	3.612×10^{-8}	1.002×10^{-7}	1.858×10^{-8}
HS	1.694×10^{3}	2.893×10^{3}	3.857×10^{3}	5.336×10^{2}
		F2		
DESNNA	2.626×10^{-2}	3.152×10^{-1}	1.010×10^{0}	2.389×10^{-1}
CCLNNA	8.691×10^{-3}	2.016×10^{-2}	3.789×10^{-2}	6.979×10^{-3}
TSA	$\mathbf{1.070 \times 10^{-8}}$	$\mathbf{1.568 \times 10^{-6}}$	$\mathbf{7.838 \times 10^{-6}}$	$\mathbf{2.323 \times 10^{-6}}$
PSO	2.019×10^{1}	2.019×10^{1}	2.696×10^{1}	3.218×10^{0}
GA	1.367×10^{0}	2.145×10^{0}	3.068×10^{0}	3.789×10^{-1}
HS	4.165×10^{1}	4.724×10^{1}	5.265×10^{1}	2.566×10^{0}
		F3		
DESNNA	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{4.822 \times 10^{1}}$	$\mathbf{3.493 \times 10^{-13}}$
CCLNNA	4.822×10^{1}	5.123×10^{1}	1.308×10^{2}	1.507×10^{1}
TSA	4.841×10^{1}	4.863×10^{1}	4.882×10^{1}	1.484×10^{-1}
PSO	2.430×10^{10}	2.430×10^{10}	3.757×10^{10}	7.454×10^{9}
GA	4.822×10^{1}	4.825×10^{1}	4.861×10^{1}	8.374×10^{-2}
HS	8.008×10^{7}	1.502×10^{8}	2.711×10^{8}	4.366×10^{7}
		F4		
DESNNA	**0**	5.978×10^{-1}	4.975×10^{0}	1.470×10^{0}
CCLNNA	1.020×10^{-8}	$\mathbf{3.317 \times 10^{-2}}$	$\mathbf{9.950 \times 10^{-1}}$	$\mathbf{1.817 \times 10^{-1}}$
TSA	2.270×10^{2}	3.014×10^{2}	3.827×10^{2}	4.299×10^{1}
PSO	5.530×10^{2}	5.530×10^{2}	6.947×10^{2}	5.642×10^{1}
GA	8.955×10^{0}	2.030×10^{1}	5.373×10^{1}	9.087×10^{0}
HS	2.478×10^{2}	2.783×10^{2}	3.106×10^{2}	1.740×10^{1}
		F5		
DESNNA	**0**	**0**	$\mathbf{5.684 \times 10^{-14}}$	$\mathbf{3.077 \times 10^{-14}}$
CCLNNA	3.530×10^{-8}	7.538×10^{-3}	5.867×10^{-2}	1.363×10^{-2}
TSA	2.842×10^{-14}	3.489×10^{-3}	2.495×10^{-2}	6.073×10^{-3}
PSO	2.170×10^{-1}	2.170×10^{-1}	9.131×10^{-1}	3.363×10^{-1}
GA	3.800×10^{-10}	1.802×10^{-3}	3.680×10^{-2}	6.969×10^{-3}
HS	1.897×10^{1}	3.000×10^{1}	3.834×10^{1}	4.877×10^{0}
		F6		
DESNNA	$\mathbf{5.684 \times 10^{-14}}$	$\mathbf{2.842 \times 10^{-14}}$	$\mathbf{1.705 \times 10^{-13}}$	$\mathbf{5.853 \times 10^{-14}}$
CCLNNA	1.605×10^{-5}	3.366×10^{-5}	5.397×10^{-5}	7.631×10^{-6}
TSA	8.527×10^{-14}	1.060×10^{0}	3.385×10^{0}	1.436×10^{0}
PSO	1.919×10^{-7}	1.919×10^{-7}	1.809×10^{-6}	3.368×10^{-7}
GA	9.199×10^{-5}	1.027×10^{0}	1.945×10^{0}	6.774×10^{-1}
HS	7.334×10^{0}	9.207×10^{0}	1.034×10^{1}	6.722×10^{-1}
		F7		
DESNNA	6.335×10^{-15}	6.297×10^{-14}	5.688×10^{-13}	1.072×10^{-13}
CCLNNA	6.837×10^{-5}	1.053×10^{-4}	1.493×10^{-4}	2.521×10^{-5}
TSA	$\mathbf{1.162 \times 10^{-132}}$	$\mathbf{2.264 \times 10^{-127}}$	$\mathbf{6.129 \times 10^{-126}}$	$\mathbf{1.117 \times 10^{-126}}$
PSO	1.097×10^{-10}	1.097×10^{-10}	1.058×10^{-9}	1.998×10^{-10}
GA	4.951×10^{-1}	2.597×10^{0}	4.642×10^{0}	1.114×10^{0}
HS	1.917×10^{1}	2.156×10^{1}	2.353×10^{1}	1.276×10^{0}
		F8		
DESNNA	7.039×10^{-5}	1.794×10^{-3}	1.380×10^{-2}	2.701×10^{-3}
CCLNNA	3.661×10^{-2}	8.557×10^{-2}	1.540×10^{-1}	2.742×10^{-2}
TSA	$\mathbf{6.565 \times 10^{-58}}$	$\mathbf{1.529 \times 10^{-34}}$	$\mathbf{3.362 \times 10^{-33}}$	$\mathbf{6.436 \times 10^{-34}}$
PSO	3.630×10^{4}	3.630×10^{4}	4.899×10^{4}	5.672×10^{3}
GA	9.505×10^{-1}	9.095×10^{0}	4.847×10^{1}	1.405×10^{1}
HS	3.420×10^{4}	5.157×10^{4}	6.727×10^{4}	9.005×10^{3}

Table 6. *Cont.*

Methods	Best Error	Average Error	Worst Error	Error Standard Deviation
		F9		
DESNNA	$\mathbf{1.418 \times 10^{-2}}$	$\mathbf{3.460 \times 10^{0}}$	$\mathbf{1.369 \times 10^{1}}$	$\mathbf{4.206 \times 10^{0}}$
CCLNNA	7.099×10^{0}	1.795×10^{1}	3.166×10^{1}	4.477×10^{0}
TSA	2.168×10^{1}	5.648×10^{1}	1.543×10^{2}	2.840×10^{1}
PSO	4.944×10^{2}	4.944×10^{2}	5.464×10^{2}	3.333×10^{1}
GA	2.670×10^{1}	3.818×10^{1}	4.814×10^{1}	4.947×10^{0}
HS	1.469×10^{2}	1.863×10^{2}	2.114×10^{2}	1.564×10^{1}
		F10		
DESNNA	**0**	**0**	**0**	**0**
CCLNNA	1.485×10^{-8}	4.157×10^{-8}	1.051×10^{-7}	2.222×10^{-8}
TSA	0	4.246×10^{0}	3.191×10^{1}	9.976×10^{0}
PSO	3.892×10^{3}	3.892×10^{3}	5.628×10^{3}	7.097×10^{2}
GA	5.249×10^{0}	8.968×10^{0}	1.727×10^{1}	2.900×10^{0}
HS	1.519×10^{2}	1.987×10^{2}	2.723×10^{2}	3.068×10^{1}
		F11		
DESNNA	$\mathbf{1.787 \times 10^{-2}}$	$\mathbf{2.126 \times 10^{0}}$	$\mathbf{1.356 \times 10^{1}}$	$\mathbf{3.941 \times 10^{0}}$
CCLNNA	8.918×10^{0}	1.739×10^{1}	3.008×10^{1}	4.385×10^{0}
TSA	1.440×10^{1}	5.089×10^{1}	1.683×10^{2}	3.334×10^{1}
PSO	4.653×10^{2}	4.653×10^{2}	5.113×10^{2}	2.339×10^{1}
GA	2.475×10^{1}	3.564×10^{1}	4.380×10^{1}	4.903×10^{0}
HS	1.603×10^{2}	1.807×10^{2}	2.132×10^{2}	1.307×10^{1}
		F12		
DESNNA	5.449×10^{-10}	$\mathbf{1.710 \times 10^{-8}}$	$\mathbf{8.094 \times 10^{-8}}$	$\mathbf{1.962 \times 10^{-8}}$
CCLNNA	8.100×10^{-2}	2.453×10^{-1}	1.333×10^{0}	2.410×10^{-1}
TSA	$\mathbf{1.137 \times 10^{-13}}$	2.020×10^{0}	1.651×10^{1}	4.570×10^{0}
PSO	3.922×10^{4}	3.922×10^{4}	7.287×10^{4}	1.119×10^{4}
GA	7.491×10^{-2}	4.669×10^{0}	1.343×10^{1}	3.597×10^{0}
HS	7.270×10^{2}	1.231×10^{3}	2.018×10^{3}	3.087×10^{2}
		F13		
DESNNA	$\mathbf{2.314 \times 10^{-8}}$	1.017×10^{0}	7.469×10^{0}	$\mathbf{2.367 \times 10^{0}}$
CCLNNA	1.724×10^{-2}	$\mathbf{3.492 \times 10^{-1}}$	$\mathbf{2.526 \times 10^{0}}$	6.498×10^{-1}
TSA	1.577×10^{2}	2.325×10^{2}	3.012×10^{2}	3.723×10^{1}
PSO	4.077×10^{2}	4.077×10^{2}	5.031×10^{2}	4.723×10^{1}
GA	1.084×10^{1}	2.201×10^{1}	3.664×10^{1}	5.255×10^{0}
HS	1.602×10^{2}	1.989×10^{2}	2.269×10^{2}	1.666×10^{1}
		F14		
DESNNA	**0**	$\mathbf{3.196 \times 10^{-3}}$	5.157×10^{-2}	1.120×10^{-2}
CCLNNA	6.372×10^{-8}	2.275×10^{-2}	9.816×10^{-2}	2.461×10^{-2}
TSA	2.274×10^{-13}	7.897×10^{-3}	$\mathbf{4.276 \times 10^{-2}}$	$\mathbf{1.055 \times 10^{-2}}$
PSO	2.045×10^{4}	2.045×10^{4}	3.515×10^{4}	7.255×10^{3}
GA	2.863×10^{-7}	1.402×10^{-2}	5.987×10^{-2}	1.993×10^{-2}
HS	7.064×10^{1}	1.534×10^{2}	2.943×10^{2}	6.223×10^{1}
		F15		
DESNNA	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{2.346 \times 10^{1}}$	$\mathbf{5.357 \times 10^{-10}}$
CCLNNA	2.346×10^{1}	2.347×10^{1}	2.356×10^{1}	1.750×10^{-2}
TSA	8.121×10^{1}	1.363×10^{2}	2.428×10^{2}	3.715×10^{1}
PSO	6.650×10^{5}	6.650×10^{5}	1.519×10^{6}	3.700×10^{5}
GA	2.546×10^{1}	4.732×10^{1}	1.409×10^{2}	2.594×10^{1}
HS	1.258×10^{3}	1.565×10^{3}	1.953×10^{3}	2.054×10^{2}
		F16		
DESNNA	$\mathbf{1.699 \times 10^{-5}}$	$\mathbf{2.869 \times 10^{-1}}$	$\mathbf{4.255 \times 10^{0}}$	$\mathbf{8.419 \times 10^{-1}}$
CCLNNA	5.691×10^{0}	1.168×10^{1}	2.231×10^{1}	4.565×10^{0}
TSA	8.421×10^{0}	4.459×10^{1}	1.386×10^{2}	2.386×10^{1}
PSO	6.533×10^{3}	6.533×10^{3}	1.763×10^{4}	4.735×10^{3}
GA	1.568×10^{1}	2.582×10^{1}	3.621×10^{1}	5.827×10^{0}
HS	1.653×10^{2}	2.057×10^{2}	2.295×10^{2}	1.711×10^{1}

Table 6. *Cont.*

Methods	Best Error	Average Error	Worst Error	Error Standard Deviation
		F17		
DESNNA	$\mathbf{1.061 \times 10^{1}}$	$\mathbf{2.680 \times 10^{1}}$	1.434×10^{2}	3.733×10^{1}
CCLNNA	1.610×10^{1}	3.373×10^{1}	$\mathbf{8.777 \times 10^{1}}$	1.650×10^{1}
TSA	2.165×10^{1}	6.243×10^{1}	4.272×10^{2}	7.386×10^{1}
PSO	2.160×10^{9}	2.160×10^{9}	9.648×10^{9}	2.406×10^{9}
GA	3.540×10^{1}	5.233×10^{1}	9.555×10^{1}	$\mathbf{1.336 \times 10^{1}}$
HS	5.139×10^{2}	2.355×10^{3}	9.487×10^{3}	2.902×10^{3}
		F18		
DESNNA	$\mathbf{7.856 \times 10^{-6}}$	$\mathbf{1.070 \times 10^{0}}$	1.092×10^{1}	2.881×10^{0}
CCLNNA	5.046×10^{-1}	2.065×10^{0}	$\mathbf{4.341 \times 10^{0}}$	$\mathbf{1.014 \times 10^{0}}$
TSA	8.064×10^{1}	1.093×10^{2}	1.443×10^{2}	1.659×10^{1}
PSO	1.737×10^{2}	1.737×10^{2}	2.260×10^{2}	1.931×10^{1}
GA	2.264×10^{1}	3.461×10^{1}	4.626×10^{1}	6.337×10^{0}
HS	8.716×10^{1}	9.495×10^{1}	1.038×10^{2}	4.256×10^{0}

5. Conclusions and Future Work

In this paper, a new meta-heuristic algorithm called the neural network algorithm with dropout using elite selection strategy (DESNNA) is proposed, which is a new variant of the neural network algorithm (NNA). In the DESNNA, when a new population is generated from the previous population, a certain percentage of individuals who perform the worst are dropped and a certain proportion of the individuals of the previous generation with the best performance are retained and directly enter the next generation to ensure the outstanding performance of the population. In order to verify the effectiveness of the improved strategy, the DESNNA is used on 18 well-known benchmark functions. The experimental results showed that the DESNNA outperforms the NNA on all 18 benchmark functions and the DESNNA beats each other compared algorithm on more than 80% of benchmark functions. Therefore, the introduced dropout and elite selection strategy improved the optimization performance of the NNA, and the DESNNA is a powerful algorithm for solving optimization problems. This work can advance the state-of-the-art. The improved DESNNA can be applied to the single mixed refrigerant process for synthetic natural gas liquefaction or the optimal planning and operation of distributed generations and capacitor banks in the radial distribution networks, which can help to improve the annual energy loss mitigation and cost savings.

As for future research, because the NNA is inspired by the neural network, it can be optimized from the perspective of a neural network. Other new variants of the NNA should be proposed in future research, such as introducing back-propagation and gradient descent to update the weight matrix. The hybridization of the NNA with other meta-heuristic algorithms may form a better optimization algorithm. What is more, the DESNNA can be applied to solve real-world optimization problems in engineering, such as constrained engineering design problems.

Author Contributions: Conceptualization, investigation, G.W. and Y.W.; methodology K.W.; software, Y.W.; validation, G.W. and Y.W.; data curation, K.W.; writing—original draft preparation, K.W.; writing—review and editing, Y.W. and G.W.; supervision, G.W. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are thankful to the anonymous for their valuable suggestions during the review process.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

D	The dimension of optimization problem
N	Population size
LB	The lower limit of variables
UB	The upper limit of variables
Max_iteration	The maximum number of iterations

References

1. Sergeyev, Y.D.; Kvasov, D.E. A deterministic global optimization using smooth diagonal auxiliary functions. *Commun. Nonlinear Sci. Numer. Simul.* **2015**, *21*, 99–111. [CrossRef]
2. Magoulas, G.D.; Vrahatis, M.N. Adaptive algorithms for neural network supervised learning: A deterministic optimization approach. *Int. J. Bifurc. Chaos* **2006**, *16*, 1929–1950. [CrossRef]
3. Kvasov, D.E.; Mukhametzhanov, M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Appl. Math. Comput.* **2018**, *318*, 245–259. [CrossRef]
4. Sergeyev, Y.D.; Kvasov, D.E.; Mukhametzhanov, M.S. Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms. *Math. Comput. Simul.* **2017**, *141*, 96–109. [CrossRef]
5. Ma, Y.; Wang, Z.; Yang, H.; Yang, L. Artificial intelligence applications in the development of autonomous vehicles: A survey. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 315–329. [CrossRef]
6. Zhao, Z.; Liu, S.; Zhou, M.; Abusorrah, A. Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem. *IEEE/CAA J. Autom. Sin.* **2020**, *8*, 1199–1209. [CrossRef]
7. Zhang, Z.; Cao, Y.; Cui, Z.; Zhang, W.; Chen, J. A Many-Objective Optimization Based Intelligent Intrusion Detection Algorithm for Enhancing Security of Vehicular Networks in 6G. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5234–5243. [CrossRef]
8. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
9. Wang, G.-G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Trans. Emerg. Top. Comput.* **2020**, *8*, 20–30. [CrossRef]
10. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef]
11. Wang, G.-G.; Gao, D.; Pedrycz, W. Solving multi-objective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Trans. Ind. Inform.* **2022**, 1. [CrossRef]
12. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
13. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
14. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
15. Cui, Z.; Zhang, J.; Wu, D.; Cai, X.; Wang, H.; Zhang, W.; Chen, J. Hybrid many-objective particle swarm optimization algorithm for green coal production problem. *Inf. Sci.* **2020**, *518*, 256–271. [CrossRef]
16. Zhang, W.; Hou, W.; Li, C.; Yang, W.; Gen, M. Multidirection Update-Based Multiobjective Particle Swarm Optimization for Mixed No-Idle Flow-Shop Scheduling Problem. *Complex Syst. Model. Simul.* **2021**, *1*, 176–197. [CrossRef]
17. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
18. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
19. Gao, D.; Wang, G.-G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
20. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **1996**, *26*, 29–41. [CrossRef]
21. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
22. Sadollah, A.; Sayyaadi, H.; Yadav, A. A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Appl. Soft Comput.* **2018**, *71*, 747–782. [CrossRef]
23. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [CrossRef]

24. Dhiman, G.; Kumar, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowl.-Based Syst.* **2019**, *165*, 169–196. [CrossRef]
25. Kaur, S.; Awasthi, L.K.; Sangal, A.; Dhiman, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541. [CrossRef]
26. Wang, G.-G.; Deb, S.; Coelho, L.d.S. Elephant herding optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), Bali, Indonesia, 7–9 December 2015; pp. 1–5.
27. Dhiman, G.; Kaur, A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *82*, 148–174. [CrossRef]
28. Zhang, Y. Chaotic neural network algorithm with competitive learning for global optimization. *Knowl.-Based Syst.* **2021**, *231*, 107405. [CrossRef]
29. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
30. Lakshminarayanan, S.; Abdulgader, M.; Kaur, D. Scheduling energy storage unit with GWO for smart home integrated with renewable energy. *Int. J. Artif. Intell. Soft Comput.* **2020**, *7*, 146–163.
31. Wang, G.-G.; Deb, S.; Coelho, L.D.S. Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems. *Int. J. Bio-Inspired Comput.* **2018**, *12*, 1–22. [CrossRef]
32. Wang, G.-G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
33. Ghaemi, M.; Feizi-Derakhshi, M.-R. Forest optimization algorithm. *Expert Syst. Appl.* **2014**, *41*, 6676–6687. [CrossRef]
34. Grabski, J.K.; Walczak, T.; Buśkiewicz, J.; Michałowska, M. Comparison of some evolutionary algorithms for optimization of the path synthesis problem. In Proceedings of the AIP Conference Proceedings, Lublin, Poland, 13–16 September 2017; p. 020006.
35. Liang, Y.-C.; Cuevas Juarez, J.R. A novel metaheuristic for continuous optimization problems: Virus optimization algorithm. *Eng. Optim.* **2016**, *48*, 73–93. [CrossRef]
36. Grabski, J.K.; Mrozek, A. Identification of elastoplastic properties of rods from torsion test using meshless methods and a metaheuristic. *Comput. Math. Appl.* **2021**, *92*, 149–158. [CrossRef]
37. Qadeer, K.; Ahmad, A.; Naquash, A.; Qyyum, M.A.; Majeed, K.; Zhou, Z.; He, T.; Nizami, A.-S.; Lee, M. Neural network-inspired performance enhancement of synthetic natural gas liquefaction plant with different minimum approach temperatures. *Fuel* **2022**, *308*, 121858. [CrossRef]
38. Bhullar, A.K.; Kaur, R.; Sondhi, S. Design and Comparative Analysis of Optimized Fopid Controller Using Neural Network Algorithm. In Proceedings of the 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), Rupnagar, India, 26–28 November 2020; pp. 91–96.
39. Zhang, Y.; Jin, Z.; Chen, Y. Hybrid teaching–learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowl.-Based Syst.* **2020**, *187*, 104836. [CrossRef]
40. Zhang, Y.; Jin, Z.; Chen, Y. Hybridizing grey wolf optimization with neural network algorithm for global numerical optimization problems. *Neural Comput. Appl.* **2020**, *32*, 10451–10470. [CrossRef]
41. Zhang, H.; Sheng, J.J. Complex fracture network simulation and optimization in naturally fractured shale reservoir based on modified neural network algorithm. *J. Nat. Gas Sci. Eng.* **2021**, *95*, 104232. [CrossRef]
42. Nguyen, T.P.; Nguyen, T.A.; Phan, T.V.-H.; Vo, D.N. A comprehensive analysis for multi-objective distributed generations and capacitor banks placement in radial distribution networks using hybrid neural network algorithm. *Knowl.-Based Syst.* **2021**, *231*, 107387. [CrossRef]
43. Van Tran, T.; Truong, B.-H.; Nguyen, T.P.; Nguyen, T.A.; Duong, T.L.; Vo, D.N. Reconfiguration of Distribution Networks With Distributed Generations Using an Improved Neural Network Algorithm. *IEEE Access* **2021**, *9*, 165618–165647. [CrossRef]
44. Marugán, A.P.; Márquez, F.P.G.; Perez, J.M.P.; Ruiz-Hernández, D. A survey of artificial neural network in wind energy systems. *Appl. Energy* **2018**, *228*, 1822–1836. [CrossRef]
45. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
46. Bhandari, D.; Paul, S.; Narayan, A. Deep neural networks for multimodal data fusion and affect recognition. *Int. J. Artif. Intell. Soft Comput.* **2020**, *7*, 130–145.
47. Agrawal, A.; Barratt, S.; Boyd, S. Learning Convex Optimization Models. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1355–1364. [CrossRef]
48. Hirasawa, T.; Aoyama, K.; Tanimoto, T.; Ishihara, S.; Shichijo, S.; Ozawa, T.; Ohnishi, T.; Fujishiro, M.; Matsuo, K.; Fujisaki, J. Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images. *Gastric Cancer* **2018**, *21*, 653–660. [CrossRef] [PubMed]
49. Paoletti, M.; Haut, J.; Plaza, J.; Plaza, A. A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS J. Photogramm. Remote Sens.* **2018**, *145*, 120–147. [CrossRef]
50. Devin, C.; Gupta, A.; Darrell, T.; Abbeel, P.; Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 2169–2176.
51. Parashar, S.; Senthilnath, J.; Yang, X.-S. A novel bat algorithm fuzzy classifier approach for classification problems. *Int. J. Artif. Intell. Soft Comput.* **2017**, *6*, 108–128. [CrossRef]

52. Laudani, A.; Lozito, G.M.; Riganti Fulginei, F.; Salvini, A. On training efficiency and computational costs of a feed forward neural network: A review. *Comput. Intell. Neurosci.* **2015**, *2015*, 818243. [CrossRef]
53. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.-G.; Chen, J. Detection of malicious code variants based on deep learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
54. Herrera, F.; Lozano, M.; Molina, D. Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and Other Metaheuristics for Large Scale Continuous Optimization Problems. Available online: http://150.214.190.154/sites/default/files/files/TematicWebSites/EAMHCO/functions1-19.pdf (accessed on 25 April 2022).
55. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Comput. Intell. Lab. Zhengzhou Univ. Zhengzhou China Technol. Rep. Nanyang Technol. Univ. Singap.* **2013**, *635*, 490.