Article

Parallelization of Genetic Algorithms and Propagation Techniques

Vasileios Charilogis[1], Ioannis G. Tsoulos[2,*]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
[*] Correspondence: itsoulos@uoi.gr;
[†] Current address: Department of Informatics and Telecommunications, University of Ioannina, Greece.
[‡] These authors contributed equally to this work.

Abstract: In the field of global optimization, various methods have been developed for locating the global minimum. One of the most popular approaches is optimization using genetic algorithms, which has gained favor due to its exceptional performance in finding effective solutions. However, this method requires computational resources and time, prompting the need for parallel techniques. In the context of this research, we propose innovative changes aiming at the efficient parallelization of genetic algorithms. This novel approach utilizes independent parallel computing units that periodically exchange the best solutions they have identified. The application of this proposed technique to various computational problems has generated enthusiasm, as the results surpass expectations. It has been observed that increasing the number of computational threads, in combination with solution exchange techniques, can significantly reduce the number of calls to the objective function. This constitutes a critical speed criterion for finding the global minimum.

Keywords: Parallel techiniques; Global optimization; Genetic algorithms; Evolutionary techniques

Citation: Charilogis V.; Tsoulos I.G.; Parallelization of Genetic Algorithms and Propagation Techniques. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Many practical problems of the modern world can be thought of either as data fitting problems, as for example, problems that appears in physics [1,2], problems related to chemistry [3,4], economic problms [5,6], medicine problems [7,8], etc. A commonly used machine learning tool to handle problems of this nature, is the Radial Basis Function (RBF) network [9,10]. Usually, an RBF network is expressed using the following equation:

$$y\left(\overrightarrow{x}\right)=\sum_{i=1}^{k}w_i\phi\left(\left\|\overrightarrow{x}-\overrightarrow{c_i}\right\|\right) \tag{1}$$

where the symbols in the equation are defined as follows:

1. The element $\overrightarrow{x}$ represents the input pattern from the dataset describing the problem. For the rest of this paper, the notation d will be used to represent the number of elements in $\overrightarrow{x}$.
2. The parameter k denotes the number of weights used to train the RBF network and the associated vector of weights is denoted as $\overrightarrow{w}$.
3. The vectors $\overrightarrow{c_i},\ i=1,..,k$ stand for the centers of the model.
4. The value $y\left(\overrightarrow{x}\right)$ represents the value of the network for the given pattern $\overrightarrow{x}$.

The $\phi(x)$ function, in most cases represent the Gaussian function given by:

$$\phi(x)=\exp\left(-\frac{(x-c)^2}{\sigma^2}\right) \tag{2}$$

The main advantages of RBF networks are:

1. They have a simpler structure than other models used in machine learning, such as multilayer perceptron neural networks (MLPs)[11], since they have only one processing layer and therefore have faster training techniques and they have faster response times.
2. They can be used to efficiently approximate any continuous function [12].

The RBF networks were applied in a variety of problems, such as problems from physics [13–16], solving differential equations [17–19], robotics [20,21], face recognition [22], digital communications [23,24], chemistry problems [25,26], economic problems [27–29], network security problems [30,31] etc. Also, recently a variety of papers have appeared proposing novel initialization techniques for the network parameters [32–34]. Also, Benoudjit et al [35] discuss the effect of kernel widths on RBF networks. Moreover, Neruda et al [36] presents a comparison of some learning methods for RBF networks. Additionally, a variety of pruning techniques [37–39] have been suggested in the literature for decreasing the number parameters. Due to the widespread usage of RBF networks but also because considerable computing time is often required for their effective training, in recent years a series of techniques have been published [40,41] for the exploitation of parallel computing units to adjust the parameters.

In the same direction of research, other machine learning models have been proposed, such as Support Vector Machines (SVM) [42,43], decision trees [44,45] etc. Also, Wang et al suggested an auto - encoder reduction method, applied on a series of large datasets[46]. Various methods have been proposed in the same direction, such as the work of Agarwal and Bhanot [47] proposed to adapt the RBF parameters, the usage of the ABC algorithm[48], the incorporation of the Firefly algorithm[49]. Furthermore, Gyamfi et al [50] recently proposed a differential RBF network that incorporates partial differential equations, aiming to make the network more robust in the presence of noise data. Also, Li et al [51] proposed a multivariate ensembles-based hierarchical linkage strategy (ME-HL) for system reliability evaluation of aeroengine cooling blades.

The parameters of the RBF network are modified in order to minimize the following loss - function, called training error of the network:

$$E(y(x,g)) = \sum_{i=1}^{m} \left(y\left(\overrightarrow{x}_i, \overrightarrow{g}\right) - t_i\right)^2 \tag{3}$$

Where the parameter m denotes the number of patterns, the t_i represent the expected output for pattern $\overrightarrow{x}_i$. The vector $\overrightarrow{g}$ represents the parameter set of the network.

A common method of calculating the parameters in these neural networks uses a technique to calculate the centers of the functions $\phi(x)$ and then the vector of weights $\overrightarrow{w}$ is calculated as a solution of a linear system of equations. Typically, the method used to calculate the centers is the well - known k-means method [52]. In many cases, this way of estimating the parameters leads to over-fitting of the model so that it cannot generalize satisfactorily to unknown data. Furthermore, since there is no range of values for the parameters, there is the possibility that they will take extremely large or extremely small values, with the result that any generalizability of the model is lost. This work suggests a two phase method to minimize the error of equation (3). During the first phase, an attempt is made to bound the parameter values to intervals at which the training error is likely to be significantly reduced. The identification of the most promising intervals for the parameters is performed using a technique that utilizes Grammatical Evolution[53], that collects information from the training data. During the second phase, the parameters can be trained inside the best located range of the first phase using some global optimization method [54,55]. In the proposed approach, the widely used method of genetic algorithm [56–58] was used for the second phase of the process. The main contributions of the suggested approach are:

1. The first phase procedure seeks to locate a range of values for the parameters, while also reducing the error of the network on the training data set.
2. The rules Grammatical Evolution uses in the first phase are simple and can be generalized to any data set for data classification or fitting.
3. The determination of the value interval is done in such a way that it is faster and more efficient to train the parameters with some optimization method during the second phase.
4. After identifying a promising value interval from the first phase, any global optimization method can be used on that value interval to effectively minimize the network training error.

The rest of this paper is divided in the following sections: in section 2 the proposed method is fully described, the section 3 presents the used datasets and the conducted experiments and finally in section 4 some a discussion on the conducted experiments is made.

2. Method description

This section starts with an exteended presentation of the Grammatical Evolution technique and the grammar that will be used to generate partition rules for the parameter set of RBFs. Afterwards, the first phase of the proposed methodology will be extensively analyzed and then the second phase, where a Genetic Algorithm will be applied to the outcome of the first phase.

2.1. Grammatical Evolution

Grammatical evolution is a Genetic algorithm, where the chromosomes are integer numbers. Genetic Algorithms was initially proposed by John Holland [59] are biologicaly inspired algorithms. The algorithm starts by formin a population of potential solutions to an optimization problem. These solutions are called chromosomes and they are gradually altered using the genetic operators of selection, crossover and mutation[60]. The chromosomes in the Grammatical Evolution stand for series of production rules of any given BNF (Backus–Naur form) grammar[61]. Grammatical Evolution has been applied with success in a variety of cases, such as function approximation[62,63], solving equations related to trigonometry [64], automatic composition of music[65], construction of neural networks [66,67], producing numeric constraints[68], video games [69,70], estimation of energy demand[71], combinatorial optimization [72], cryptography [73] etc. The BNF grammar can be used to describe the syntax of programming languages and usually it is defined as $G = (N, T, S, P)$ where

- N is a set of the non - terminal symbols. A series of production rules is associated with every non - terminal symbol. The application of these production rules produces series of terminal symbols.
- T stands for the set of terminal symbols.
- S denotes the start symbol of the grammar and $S \in N$.
- P defines the set of production rules. These are rules are following the following notations: $A \to a$ or $A \to aB,\ A, B \in N,\ a \in T$.

The algorithm begins using the symbol S and gradually creates series of terminal symbols with the assistance of the production rules. The production rules are selected through the following procedure:

- Denote with V the next element form the current chromosome.
- The next production rule is calculated as: Rule = V mod R. The number R stands for the total number of production rules for non – terminal symbol which is currently under processing.

The Algorithm 1 shows the BNF grammar used by the proposed method. Each non - terminal symbol of the grammar is enclosed in <> symbol. The numbers that are enclosed in parentheses represent for the sequence numbers of production rules for every non - terminal symbol. Every RBF network with k weights is constructed by the following series of parameters:

1. A series of vectors $\overrightarrow{c_i},\ i = 1, \ldots, k$ that stand for the centers of the model.
2. For every Gaussian unit an additional parameter σ_i is required.
3. The output weight vector $\overrightarrow{w}$.

The number n is the total number of parameters of the problem. In the case of this paper, it is the total number of parameters of the RBF network. For the current work, the number n can be computed using the following formula:

$$n = (d + 2) \times k \tag{4}$$

The number n in the corresponding grammar is computed as follows:

1. For each center $\overrightarrow{c_i},\ i = 1, .., k$ there are d variables. As a consequence, every center required $d \times k$ parameters.
2. Every Gaussian unit requires an additional parameter: $\sigma_i,\ i = 1, .., k$, which means k more parameters.
3. The weight vector $\overrightarrow{w}$ used in the output has k parameters.

As an example of production, the chromosome $x = [9, 8, 6, 4, 15, 9, 16, 23, 8]$ is considered where $d = 2,\ k = 2,\ n = 8$. The steps to produce the final program $p_{\text{test}} = (x7, 0, 1), (x1, 1, 0)$ are outlined in Table 1. Every partition program consists of a series of partition rules. Each partition rule contains three elements:

1. The variable for which its original interval will be partitioned, for example x_7.
2. An integer number with values 0 and 1 at the left margin of the interval. If this value is 1, then the left margin of the corresponding variable's value field will be divided by two, otherwise no change will be made.
3. An integer number with values 0 and 1 at the right end of the range of values of the variable. If this value is 1, then the right end of the corresponding variable's value field will be divided by two, otherwise no change will be made.

Hence, for the example program p_{test} the two partition rules will divide the right end of the variable x_7 and the left end of the variable x_1.

Algorithm 1 The BNF grammar used in the proposed method, to produce intervals for the RBF parameters. By using this grammar in the first phase of the current work, the optimal interval of values for the parameters will be identified.

```
S::=<expr>   (0)
<expr> ::=  (<xlist> , <digit>,<digit>)  (0)
           |<expr>,<expr>                (1)
<xlist>::=x1    (0)
           | x2 (1)
           .........
           | xn (n)
<digit>  ::= 0 (0)
           | 1 (1)
```

Table 1. The series of steps used to computer a valid expression from the BNF grammar for a given chromosome.

Expression	Chromosome	Operation
	9,8,6,4,15,9,16,23,8	9 mod 2=1
<expr>,<expr>	8,6,4,15,9,16,23,8	8 mod 2=0
(<xlist>,<digit>,<digit>),<expr>	6,4,15,9,16,23,8	6 mod 8=6
(x7,<digit>,<digit>),<expr>	4,15,9,16,23,8	4 mod 2=0
(x7,0,<digit>),<expr>	15,9,16,23,8	15 mod 2=1
(x7,0,1),<expr>	9,16,23,8	9 mod 2 =1
(x7,0,1),(<xlist>,<digit>,<digit>)	16,23,8	16 mod 8=0
(x7,0,1),(x1,<digit>,<digit>)	23,8	23 mod 2=1
(x7,0,1),(x1,1,<digit>)	8	8 mod 2=0
(x7,0,1),(x1,1,0)		

2.2. *The first phase of the proposed algorithm*

The purpose of this phase is to initialize the bounds of the RBF model and discover a promising interval for the corresponding values. For this initialization, the K-Means algorithm [52] technique is used, which is also used for the traditional RBF network training technique. A description of this algorithm in a series of steps is shown in Algorithm 2.

Algorithm 2 The K-Means algorithm.

```
1. Repeat
   (a) Define $S_j = \{\}$, $j = 1..k$
   (b) For every pattern $x_i$, $i = 1, ..., m$ do
       i.  Compute $j^* = \min_{i=1}^{k}\left\{D\left(x_i, c_j\right)\right\}$.
       ii. Compute $S_{j^*} = S_{j^*} \cup \{x_i\}$.
   (c) EndFor
   (d) For every center $c_j$, $j = 1..k$ do
       i.  Denote as $M_j$ the number of points in set $S_j$
       ii. Compute $c_j$ as
           $$c_j = \frac{1}{M_j}\sum_{i=1}^{M_j} x_i$$
   (e) EndFor
2. Compute the quantities $s_j$ as
   $$\sigma_j^2 = \frac{\sum_{i=1}^{M_j}\left(x_i - c_j\right)^2}{M_j}$$
3. Stop the algorithm, if centers $c_j$ do not change anymore.
```

Having calculated the centers c_i and the corresponding variances σ_i, the algorithm continues to compute the vectors $\overrightarrow{L}$, $\overrightarrow{R}$ with dimension n, that will be used as the initial bounds of the parameters. The above vectors are calculated through the procedure of the algorithm 3.

Algorithm 3 The proposed algorithm used to locate the vectors $\overrightarrow{L}$, $\overrightarrow{R}$

```
1. Set m=0
2. Define $F > 1$, the scaling factor.
3. Define $B > 0$, the initial upper bound for the weight vector $\overrightarrow{w}$.
4. For $i = 1..k$ do
   (a) For $j = 1..d$ do
       i.  Compute $L_m = -F \times c_{ij}$, $R_m = F \times c_{ij}$
       ii. Compute $m = m + 1$
   (b) EndFor
   (c) Compute $L_m = -F \times \sigma_i$, $R_m = F \times \sigma_i$
   (d) Compute $m = m + 1$
5. EndFor
6. For $j = 1, ..., k$ do
   (a) Compute $L_m = -B$, $R_m = B$
   (b) Compute $m = m + 1$
7. EndFor
```

The range of values for the first $(d + 1) \times k$ parameters is estimated by multiplying the parameter F by the values already estimated by the K-Means algorithm. The bounds of the weight vector $\overrightarrow{w}$ are initialized using the value B. Subsequently, genetic algorithm described here is performed to estimate the most a promising range $\left[\overrightarrow{L}, \overrightarrow{R}\right]$ for the RBF parameters:

```
1. **Define** as $N_c$ the number of chromosomes that will participate in the the Grammatical Evolution procedure.
2. **Define** as $k$ the number of processing nodes of the used RBF model.
3. **Define** as $N_g$ the number of allowed generations.
4. **Define** as $p_s$ the used selection rate, with $p_s \leq 1$.
5. **Define** as $p_m$ the used mutation rate, with $p_m \leq 1$.
6. **Define** as $N_s$as the total number of RBF networks that will be created randomly in every fitness calculation.
7. **Initialize** $N_c$ chromosomes as sets of random numbers.
8. **Set** $f^* = [\infty, \infty]$, the fitness of the best chromosome. The fitness function $f_g$ of any provided chromosome $g$ is considered as an interval $f_g = \left[f_{g,\mathbf{low}}, f_{g,\mathbf{upper}}\right]$
9. **Set** iter=0.
10. **For** $i = 1, \ldots, N_c$ **do**
    (a) **Produce** the partition program $p_i$ using the grammar of Figure 1 for the chromosome $i$.
    (b) **Produce** the bounds $\left[\vec{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$ for the partition program $p_i$.
    (c) **Set** $E_{\text{min}} = \infty,\ E_{\text{max}} = -\infty$
    (d) **For** $j = 1, \ldots, N_S$ **do**
        i. **Create** randomly a set of parameters $\overrightarrow{g_j} \in \left[\vec{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$
        ii. **Calculate** the error $E_{\overrightarrow{g_j}} = \sum_{k=1}^{M} \left(y\left(\overrightarrow{x_k}, \overrightarrow{g_j}\right) - t_k\right)^2$
        iii. **If** $E_{\overrightarrow{g_j}} \leq E_{\text{min}}$ **then** $E_{\text{min}} = E_{\overrightarrow{g_j}}$
        iv. **If** $E_{\overrightarrow{g_j}} \geq E_{\text{max}}$ **then** $E_{\text{max}} = E_{\overrightarrow{g_j}}$
    (e) **EndFor**
    (f) **Set** the fitness$f_i = \left[E_{\text{min}}, E_{\text{max}}\right]$
11. **EndFor**
12. **Perform** the procedure of selection: Initially, the chromosomes of the population are sorted according to their fitness values. Since the fitness values are intervals, the $L^*$ operator is defined as:

    $$L^*(f_a, f_b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \quad (5)$$

    As a consuquence, the fitness value $f_a$ is considered smaller than $f_b$ if $L^*(f_a, f_b) =$ TRUE. The first $(1 - p_s) \times N_c$ chromosomes with smaller fitness values are copied without changes to the next generation of the algorithm. The rest of chromosomes are replaced by chromosomes created in the crossover procedure.
13. **Perform** the crossover procedure. The crossover procedure will create new $p_s \times N_c$ chromosomes. For every pair of created offsprings two parents $(z, w)$ are selected from the curent population using the tournament selection. These parent will produce the offsprings $\tilde{z}$ and $\tilde{w}$ using the one - point crossover, shown in Figure 1.
14. **Perform** the mutation procedure. In this process a random number $r \in [0, 1]$ is drawn for every element of each chromosome. The corresponding element is changed randomly if $r \leq p_m$.
15. **Set** iter=iter+1
16. **If** iter$\leq N_g$ goto step 10.
```

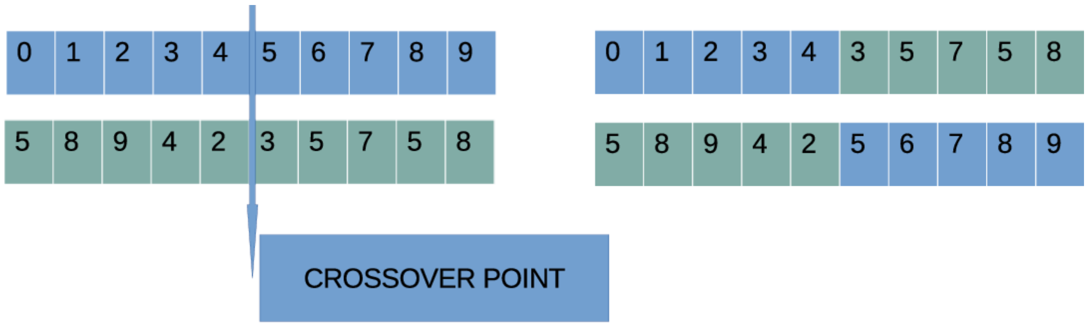

Figure 1. An example of the one point crossover procedure, as used in the Grammatical Evolution.

2.3. *The second phase of the proposed algorithm*

The second phase utilizes a genetic algorithm, to optimize the parameters of the RBF network. The optimization of the parameters uses as bounds the best interval produced in the first phase of the method. The layout of each chromosome is shown in Figure 2.

Figure 2. The layout of chromosomes in the second phase of the proposed algorithm.

c_{11}	c_{12}	...	c_{1d}	σ_1	c_{21}	c_{22}	...	c_{2d}	σ_2	...	c_{k1}	c_{k2}	...	c_{kd}	σ_k	w_1	w_2	$\ldots$	w_k

1. **Initialization Step**
   (a) **Define** as N_c as the number of chromosomes.
   (b) **Define** as N_g the total number of generations.
   (c) **Define** as k the number of processing nodes of the used RBF model.
   (d) **Define** as $S = [L_{\text{best}}, R_{\text{best}}]$ the best located interval of the first stage of the algorithm, of subsection 2.2.
   (e) **Produce** N_C random chromosomes in S.
   (f) **Define** as p_s the used selection rate, with $p_s \leq 1$.
   (g) **Define** as p_m the used mutation rate, with $p_m \leq 1$.
   (h) **Set** iter=0.
2. **Fitness calculation Step**
   (a) **For** $i = 1, \ldots, N_g$ **do**
      i. **Compute** the fitness f_i of each chromosome g_i as $f_i = \sum_{j=1}^{m} \left(y\left(\overrightarrow{x_j}, \overrightarrow{g_i}\right) - t_j\right)^2$
   (b) **EndFor**
3. **Genetic operations step**
   (a) **Selection procedure.** Initially, the population is sorted according to the fitness values. The first $(1-p_s) \times N_c$ chromosomes with the lowest fitness values remain intact. The rest of chromosomes are replaced by offsprings that will be produced during the crossover procedure.
   (b) **Crossover procedure**: For every two new offispirings $(\tilde{z}, \tilde{w})$, there are two parents (z, w) that are selected from the curent population with the selection procedure of tournament selection. The offsprings are prdocuced through the following:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1-a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1-a_i) z_i \end{aligned} \quad (6)$$

   The value a_i is a random number, where $a_i \in [-0.5, 1.5]$ [74].
   (c) **Perform** the mutation procedure. In this process a random number $r \in [0, 1]$ is drawn for every element of each chromosome. The corresponding element is changed randomly if $r \leq p_m$.

4. **Termination Check Step**
   (a) **Set** $iter = iter + 1$
   (b) **If** iter$\leq N_g$ goto step 2.

The steps of the current algorithm are also outlined graphically in Figure 3 using a flowchart.

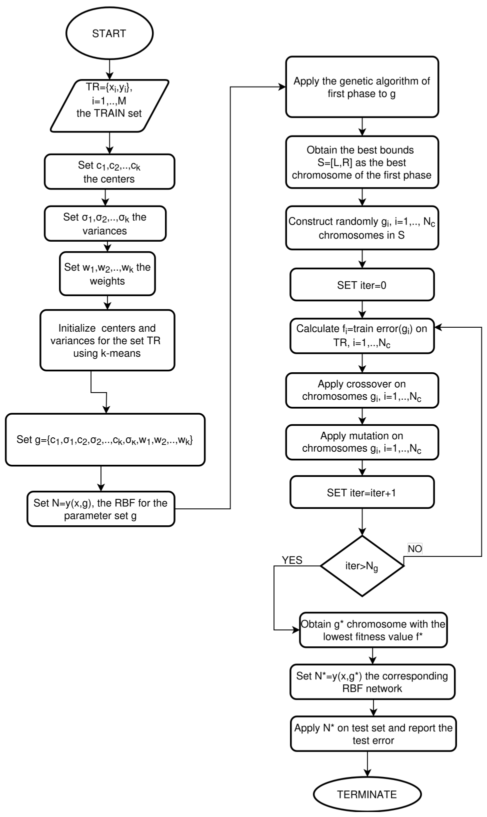
START
TR={x_i,y_i}, i=1,..,M the TRAIN set
Set c_1,c_2,..,c_k the centers
Set σ_1,σ_2,..,σ_k the variances
Set w_1,w_2,..,w_k the weights
Initialize centers and variances for the set TR using k-means
Set g={c_1,σ_1,c_2,σ_2,..,c_k,σ_k,w_1,w_2,..,w_k}
Set N=y(x,g), the RBF for the parameter set g
Apply the genetic algorithm of first phase to g
Obtain the best bounds S=[L,R] as the best chromosome of the first phase
Construct randomly g_i, i=1,.., N_c chromosomes in S
SET iter=0
Calculate f_i=train error(g_i) on TR, i=1,..,N_c
Apply crossover on chromosomes g_i, i=1,..,N_c
Apply mutation on chromosomes g_i, i=1,..,N_c
SET iter=iter+1
iter>N_g
YES
NO
Obtain g^* chromosome with the lowest fitness value f^*
Set N^*=y(x,g^*) the corresponding RBF network
Apply N^* on test set and report the test error
TERMINATE

Figure 3. The flowchart of the proposed algorithm.

3. Experiments

3.1. Experimental datasets

The proposed method was tested on a wide set of classification and regression problems found in the relevant literature. The method was compared against some other well -known machine learning models. The following databases were used to obtain the datasets:

1. The UCI dataset repository, https://archive.ics.uci.edu/ml/index.php(accessed on 5 December 2023)
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 5 December 2023)[75].
3. The Statlib URL http://lib.stat.cmu.edu/datasets/(accessed on 5 December 2023).

The classification datasets are listed in Table 2 and the regression datasets are listed in Table 3.

Table 2. The classification datasets used in the experiments. The column DATASET denotes the number of the dataset, the column CLASSES stands for the number of classes in each dataset and the column REFERENCE points to the bibliography where the use of the particular data set is presented.

DATASET	CLASSES	REFERENCE
APPENDICITIS	2	[76]
AUSTRALIAN	2	[77]
BALANCE	3	[78]
CLEVELAND	5	[79,80]
DERMATOLOGY	6	[81]
HAYES ROTH	3	[82]
HEART	2	[83]
HOUSEVOTES	2	[84]
IONOSPHERE	2	[85,86]
LIVERDISORDER	2	[87]
MAMMOGRAPHIC	2	[88]
PARKINSONS	2	[89]
PIMA	2	[90]
POPFAILURES	2	[91]
SPIRAL	2	[92]
REGIONS2	5	[93]
SAHEART	2	[94]
SEGMENT	7	[95]
WDBC	2	[96]
WINE	3	[97,98]
Z_F_S	3	[99]
ZO_NF_S	3	[99]
ZONF_S	2	[99]
ZOO	7	[100]

Table 3. The regression datasets used in the experiments. The column DATASET denotes the number of the dataset and the column REFERENCE points to the bibliography or URL (KEEL or STATLIB) where the use of the particular data set is presented.

DATASET	REFERENCE
ABALONE	[101]
AIRFOIL	[102]
BASEBALL	STATLIB
BK	[103]
BL	STATLIB
CONCRETE	[104]
DEE	KEEL
DIABETES	KEEL
FA	STATLIB
HOUSING	[105]
MB	[106]
MORTGAGE	KEEL
NT	[107]
PY	[108]
QUAKE	[109]
TREASURY	KEEL
WANKARA	KEEL

3.2. Experimental results

The RBF model used in the experiments was implemented in ANSI C++ with the assistance of the open source Armadillo library [110]. The optimization methods used were also freely available from the OPTIMUS software, available from https://github.com/itsoulos/OPTIMUS/(accessed on 5 December 2023). For validation purposed, the 10 - fold validation technique was used for all datasets and for all methods that participate in the experiments. Also, all the experiments were conducted 30 times and the seed number of the random generator was different in each execution. The average classification error is reported for the classification datasets and the average mean test error for the regression datasets. The machine used in the experiments was an AMD Ryzen 5950X with 128GB of RAM and the operating system was Debian Linux The values of the parameters used in the experiments are shown in Table 4. The experimental results for the classification datasets are outlined in Table 5 and for the regression datasets are listed in Table 6. For the tables with the experimental results, the following applies:

1. The column RPROP represents an artificial neural network [111,112]. This neural network has 10 processing nodes and was trained using the Rprop method [113].
2. The column denoted as ADAM stands the application of the Adam optimizer [114,115] to train an artificial neural network with 10 hidden nodes .
3. The column NEAT (NeuroEvolution of Augmenting Topologies) [116] denotes the application of the NEAT method for neural network training.
4. The RBF-KMEANS column denotes the original two - phase training method for RBF networks.
5. The column GENRBF stands for the RBF training method introduced in [117].
6. The column PROPOSED stands for the results obtained using the proposed method.
7. In the experimental tables an additional row was added with the title AVERAGE. This row contains the average classification or regression error for all datasets.

Table 4. The values used for the experimental parameters.

PARAMETER	**VALUE**
N_c	200
N_g	100
N_s	50
F	10.0
B	100.0
k	10
p_s	0.90
p_m	0.05

Table 5. The first column denotes the name of the classification dataset and the the numbers in cells represent classification error for every method used in the experiments. The last row stands for the average classification error for all datasets.

DATASET	RPROP	ADAM	NEAT	RBF-KMEANS	GENRBF	PROPOSED
Appendicitis	16.30%	16.50%	17.20%	12.23%	16.83%	15.77%
Australian	36.12%	35.65%	31.98%	34.89%	41.79%	22.40%
Balance	8.81%	7.87%	23.14%	33.42%	38.02%	15.62%
Cleveland	61.41%	67.55%	53.44%	67.10%	67.47%	50.37%
Dermatology	15.12%	26.14%	32.43%	62.34%	61.46%	35.73%
Hayes Roth	37.46%	59.70%	50.15%	64.36%	63.46%	35.33%
Heart	30.51%	38.53%	39.27%	31.20%	28.44%	15.91%
HouseVotes	6.04%	7.48%	10.89%	6.13%	11.99%	3.33%
Ionosphere	13.65%	16.64%	19.67%	16.22%	19.83%	9.30%
Liverdisorder	40.26%	41.53%	30.67%	30.84%	36.97%	28.44%
Mammographic	18.46%	46.25%	22.85%	21.38%	30.41%	17.72%
Parkinsons	22.28%	24.06%	18.56%	17.41%	33.81%	14.53%
Pima	34.27%	34.85%	34.51%	25.78%	27.83%	23.33%
Popfailures	4.81%	5.18%	7.05%	7.04%	7.08%	4.68%
Regions2	27.53%	29.85%	33.23%	38.29%	39.98%	25.18%
Saheart	34.90%	34.04%	34.51%	32.19%	33.90%	29.46%
Segment	52.14%	49.75%	66.72%	59.68%	54.25%	49.22%
Spiral	46.59%	48.90%	50.22%	44.87%	50.02%	23.58%
Wdbc	21.57%	35.35%	12.88%	7.27%	8.82%	5.20%
Wine	30.73%	29.40%	25.43%	31.41%	31.47%	5.63%
Z_F_S	29.28%	47.81%	38.41%	13.16%	23.37%	3.90%
ZO_NF_S	6.43%	47.43%	43.75%	9.02%	22.18%	3.99%
ZONF_S	27.27%	11.99%	5.44%	4.03%	17.41%	1.67%
ZOO	15.47%	14.13%	20.27%	21.93%	33.50%	9.33%
AVERAGE	**26.56%**	**32.36%**	**30.11%**	**28.84%**	**33.35%**	**18.73%**

Table 6. The first column denotes the name of the regression dataset and the the numbers in cells represent regression error for every method used in the experiments. The last row stands for the average regression error for all datasets.

DATASET	RPROP	ADAM	NEAT	RBF-KMEANS	GENRBF	PROPOSED
ABALONE	4.55	4.30	9.88	7.37	9.98	5.16
AIRFOIL	0.002	0.005	0.067	0.27	0.121	0.004
BASEBALL	92.05	77.90	100.39	93.02	98.91	81.26
BK	1.60	0.03	0.15	0.02	0.023	0.025
BL	4.38	0.28	0.05	0.013	0.005	0.0004
CONCRETE	0.009	0.078	0.081	0.011	0.015	0.006
DEE	0.608	0.630	1.512	0.17	0.25	0.16
DIABETES	1.11	3.03	4.25	0.49	2.92	1.74
HOUSING	74.38	80.20	56.49	57.68	95.69	21.11
FA	0.14	0.11	0.19	0.015	0.15	0.033
MB	0.55	0.06	0.061	2.16	0.41	0.19
MORTGAGE	9.19	9.24	14.11	1.45	1.92	0.014
NT	0.04	0.12	0.33	8.14	0.02	0.007
PY	0.039	0.09	0.075	0.012	0.029	0.019
QUAKE	0.041	0.06	0.298	0.07	0.79	0.034
TREASURY	10.88	11.16	15.52	2.02	1.89	0.098
WANKARA	0.0003	0.02	0.005	0.001	0.002	0.003
AVERAGE	**11.71**	**11.02**	**11.97**	**10.17**	**12.54**	**6.46**

On average, the current work appears to be 30-40% more accurate than the immediate best. In many cases, this percentage exceeds 70%. Moreover, in the vast majority of problems, the proposed technique significantly outperforms the next best available method in terms of test error. In order to validate the results, an additional experiment was executed on the classification datasets, where the number of nodes increases from 5 to 20 and the results are graphically outlined in Figure 4. From this experiment, one can draw two conclusions: firstly, the proposed technique has a significant advantage over the others to a large extent in terms of average classification error, and secondly, the proposed method is shown to be robust and not significantly dependent on the increase of processing nodes , since 5–10 processing nodes are enough to achieve low classification errors.

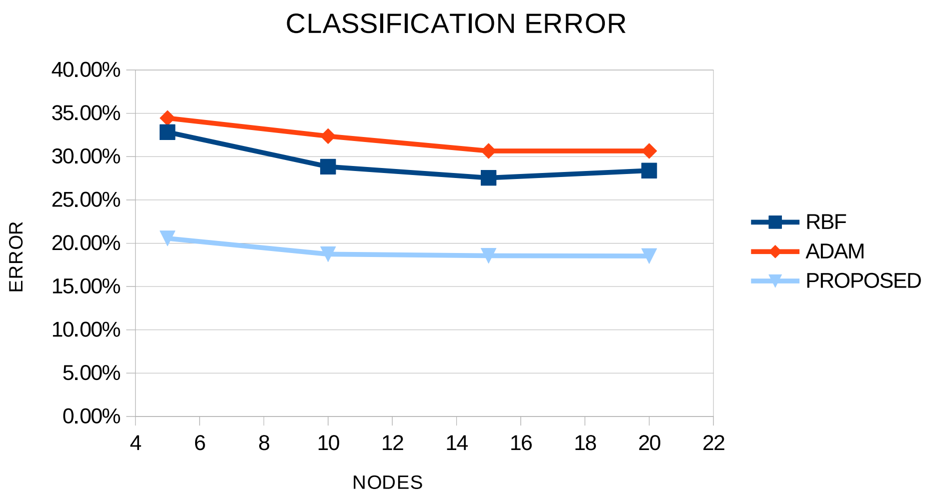

Figure 4. Average classification error for all classification datasets. The number of nodes increases from 5 to 20 and three models were used: the ADAM optimizer to optimize a neural network, the original RBF training method of two phases and the proposed method.

However, the proposed technique consists of two stages and in each of them a genetic algorithm should be executed. This means that it is significantly slower in computing time compared to the rest of the techniques and, of course, it needs more computing resources. This is graphically shown in Figure 5, where the average execution time for the method ADAM and the proposed method is shown for the classification datasets, when the number of processing nodes increases from 5 to 20. As expected, the current work requires significantly more time than a simple optimization technique such as ADAM, since it consists of two sequential genetic algorithms.

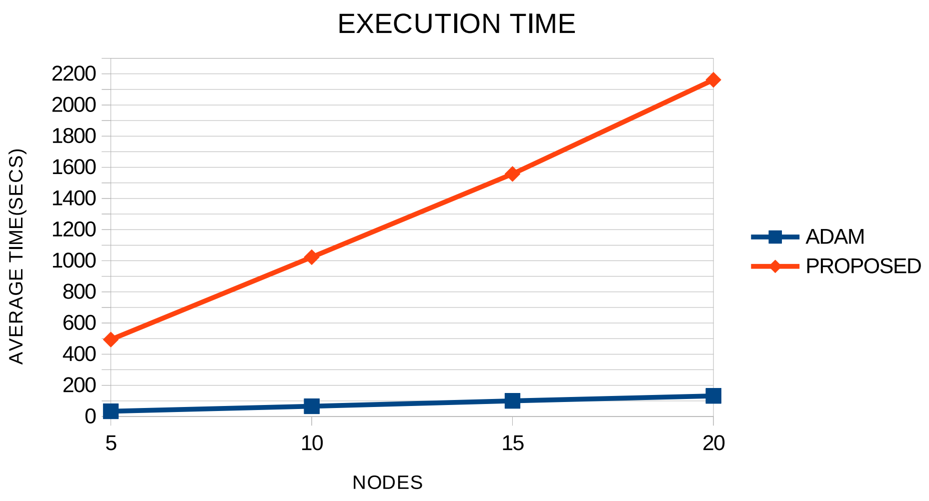

Figure 5. Average execution time for the ADAM optimizer used to train a neural network and the proposed technique.

Of course, since we are talking about Genetic Algorithms, the training time required could be significantly reduced by using parallel techniques that take advantage of modern parallel computing structures such as the MPI interface [118] or the OpenMP library [119]. The superiority of the proposed technique is also reinforced by the statistical tests carried out on the experimental results and outlined in figure 6.

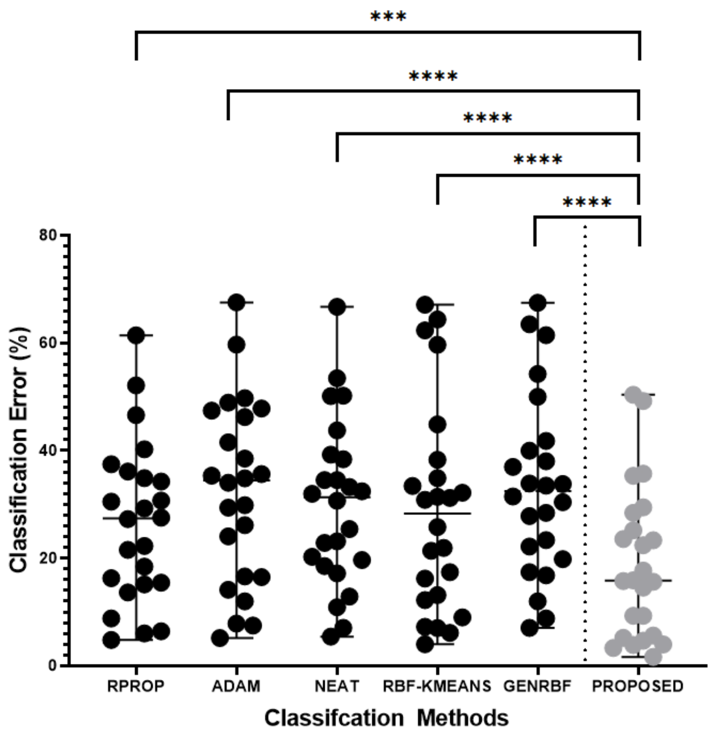

Figure 6. Scatter plot representation and the two-sample paired (Wilcoxon) signed-rank test results of the comparison for each of the five (5) classification methods (RPROP, ADAM, NEAT, RBF-KMEANS, and GENRBF) agains the PROPOSED method regarding the error on the twenty-four (24) classification datasets. The stars only intend to flag significance levels for the two most used groups. A p-value of less than 0.001 is flagged with three stars (**). A p-value of less than 0.0001 is flagged with four stars (***).

In addition, an additional set of experiments was executed on the classification data. In this set of experiments the critical parameter F took the values 3, 5 and 10. The aim of this set of experiments was to establish the sensitivity of the proposed method to changes in its parameters. The experimental results are presented in the table 7 and a statistical test on the results is presented in figure 7. The results and the statistics test indicate that there is no significant difference in the efficiency of the method for different values of the critical parameter F.

Table 7. The following table presents experimental results from the use of the proposed technique in classification problems and for different values of the critical parameter F.

DATASET	$F = 3$	$F = 5$	$F = 10$
Appendicitis	15.57%	16.60%	15.77%
Australian	24.29%	23.94%	22.40%
Balance	17.22%	15.39%	15.62%
Cleveland	52.09%	51.65%	50.37%
Dermatology	37.23%	36.81%	35.73%
Hayes Roth	35.72%	32.31%	35.33%
Heart	16.32%	15.54%	15.91%
HouseVotes	4.35%	3.90%	3.33%
Ionosphere	12.50%	11.44%	9.30%
Liverdisorder	28.08%	28.19%	28.44%
Mammographic	17.49%	17.15%	17.72%
Parkinsons	16.25%	15.17%	14.53%
Pima	23.29%	23.97%	23.33%
Popfailures	5.31%	5.86%	4.68%
Regions2	25.97%	26.29%	25.18%
Saheart	28.52%	28.59%	29.46%
Segment	44.95%	48.77%	49.22%
Spiral	15.49%	18.19%	23.58%
Wdbc	5.43%	5.01%	5.20%
Wine	7.59%	8.39%	5.63%
Z_F_S	4.37%	4.26%	3.90%
ZO_NF_S	3.79%	4.21%	3.99%
ZONF_S	2.34%	2.26%	1.67%
ZOO	11.90%	10.50%	9.33%
AVERAGE	**19.03%**	**18.93%**	**18.73%**

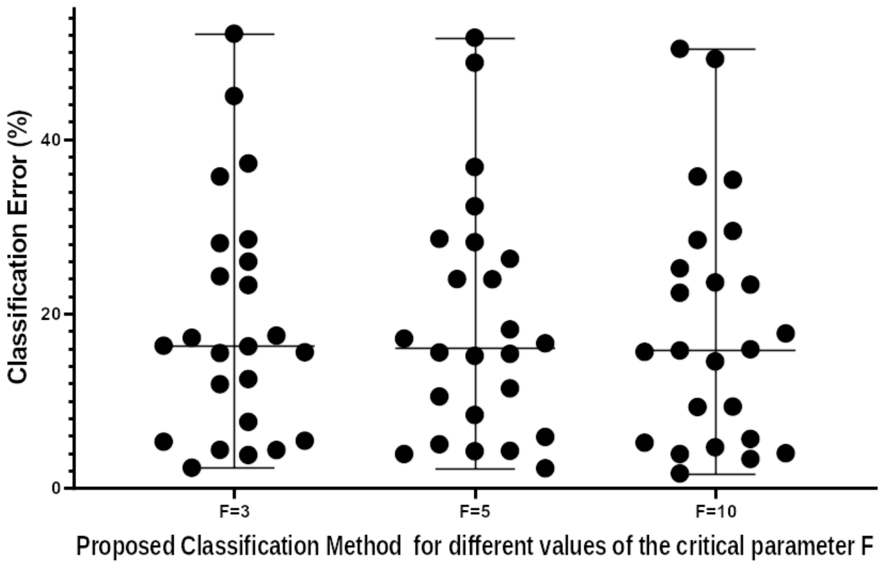

Figure 7. A Friedman test was conducted to find out whether different values of the critical parameter F had a difference or not in the classification error of the proposed method in twenty-four (24) other publicly available classification datasets. The analysis results for three different values of the critical parameter F (F=3, F=5, F=10) indicated no significant difference.

4. Conclusions

In the current work, an innovative two-stage technique was proposed for efficient training of RBF artificial neural networks. In the first stage of the application, using Grammatical Evolution, the interval of values of the neural network parameters is partitioned, so as to find a promising range that may contain low values of the training error. In the second stage, the neural network parameters are trained within the best range of values found in the first stage. The training of the parameters of the second phase is carried out using a Genetic Algorithm. The proposed method was applied on a wide series of well -known datasets from the relevant literature and was tested against a series of machine learning models. The new training technique was compared with the traditional method of training RBF networks but also with other machine learning models and from the experimental results its superiority is evident in percentages that exceed 40%. However, since the proposed technique include two genetic algorithms that are executed sequentially, the execution time required is longer compared to other techniques especially for datasets with many patterns. An immediate solution to increase the speed of the method would be the use of parallel computing techniques, since genetic algorithms can by nature be directly parallelized.

Future improvements to the proposed method may include:

1. The proposed method can be applied to other variants of artificial neural networks.
2. Use of intelligent learning techniques in place of the K-Means technique to initialize the neural network parameters.
3. Using techniques to dynamically determine the number of necessary parameters of the neural network. For the time being, the number of parameters is considered constant, but this has the consequence of observing over-training phenomena in various data sets.
4. Implementation of crossover and mutation techniques that focus more on the existing interval construction technique for the model parameters.
5. Use of efficient termination techniques for Genetic Algorithms, for the most efficient termination of techniques without wasting computing time on unnecessary iterations.
6. Usage of techniques that are based on parallel programming to increase the speed of the method.

Author Contributions: I.G.T., A.T. and E.K. conceived the idea and methodology and supervised the technical part regarding the software. I.G.T. executed the experiments, employing several datasets. A.T. performed the statistical analysis and all authors prepared the manuscript. 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: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name "iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195)

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

References

1. M. Mjahed, The use of clustering techniques for the classification of high energy physics data, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **559**, pp. 199-202, 2006.
2. M Andrews, M Paulini, S Gleyzer, B Poczos, End-to-End Event Classification of High-Energy Physics Data, Journal of Physics: Conference Series **1085**, 2018.
3. P. He, C.J. Xu, Y.Z. Liang, K.T. Fang, Improving the classification accuracy in chemistry via boosting technique, Chemometrics and Intelligent Laboratory Systems 70, pp. 39-46, 2004.
4. J.A. Aguiar, M.L. Gong, T.Tasdizen, Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning, Computational Materials Science **173**, 109409, 2020.
5. I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing **10**, pp. 215-236, 1996.
6. R. Hafezi, J. Shahrabi, E. Hadavandi, A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price, Applied Soft Computing **29**, pp. 196-210, 2015.
7. S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data **6**, 113, 2019.
8. L. Qing, W. Linhong , D. Xuehai, A Novel Neural Network-Based Method for Medical Text Classification, Future Internet **11**, 255, 2019.
9. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.
10. G.A. Montazer, D. Giveki, M. Karami, H. Rastegar, Radial basis function neural networks: A review. Comput. Rev. J **1**, pp. 52-74, 2018.
11. O.I. Abiodun, A. Jantan, A. E. Omolara, K.V. Dada, N. A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: A survey, Heliyon **4**, e00938, 2018.
12. Y. Liao, S. C. Fang, and H. L. W. Nuttle, "Relaxed conditions for radial-basis function networks to be universal approximators," Neural Networks, vol. 16, no. 7, pp. 1019–1028, 2003
13. P. Teng, Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks, Phys. Rev. E **98**, 033305, 2018.
14. R. Jovanović, A. Sretenovic, Ensemble of radial basis neural networks with K-means clustering for heating energy consumption prediction, FME Transactions **45**, pp. 51-57, 2017.
15. V.I. Gorbachenko, M.V. Zhukov, Solving boundary value problems of mathematical physics using radial basis function networks. Comput. Math. and Math. Phys. **57**, pp. 145–155, 2017.
16. J. Määttä, V. Bazaliy, J. Kimari, F. Djurabekova, K. Nordlund, T. Roos, Gradient-based training and pruning of radial basis function networks with an application in materials physics, Neural Networks **133**, pp. 123-131, 2021.
17. Nam Mai-Duy, Thanh Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Networks 14, pp. 185-199, 2001.
18. N. Mai-Duy, Solving high order ordinary differential equations with radial basis function networks. Int. J. Numer. Meth. Engng. **62**, pp. 824-852, 2005.
19. S.A. Sarra, Adaptive radial basis function methods for time dependent partial differential equations, Applied Numerical Mathematics **54**, pp. 79-94, 2005.

20. R. -J. Lian, Adaptive Self-Organizing Fuzzy Sliding-Mode Radial Basis-Function Neural-Network Controller for Robotic Systems, IEEE Transactions on Industrial Electronics **61**, pp. 1493-1503, 2014.
21. M. Vijay, D. Jena, Backstepping terminal sliding mode control of robot manipulator using radial basis functional neural networks. Computers & Electrical Engineering **67**, pp. 690-707, 2018.
22. M.J. Er, S. Wu, J. Lu, H.L. Toh, Face recognition with radial basis function (RBF) neural networks, IEEE Transactions on Neural Networks **13**, pp. 697-710, 2002.
23. C. Laoudias, P. Kemppi and C. G. Panayiotou, Localization Using Radial Basis Function Networks and Signal Strength Fingerprints in WLAN, GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, Honolulu, HI, 2009, pp. 1-6, 2009.
24. M. Azarbad, S. Hakimi, A. Ebrahimzadeh, Automatic recognition of digital communication signal, International journal of energy, information and communications **3**, pp. 21-33, 2012.
25. D.L. Yu, J.B. Gomm, D. Williams, Sensor fault diagnosis in a chemical process via RBF neural networks, Control Engineering Practice **7**, pp. 49-55, 1999.
26. V. Shankar, G.B. Wright, A.L. Fogelson, R.M. Kirby, A radial basis function (RBF) finite difference method for the simulation of reaction–diffusion equations on stationary platelets within the augmented forcing method, Int. J. Numer. Meth. Fluids **75**, pp. 1-22, 2014.
27. W. Shen, X. Guo, C. Wu, D. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, Knowledge-Based Systems 24, pp. 378-385, 2011.
28. J. A. Momoh, S. S. Reddy, Combined Economic and Emission Dispatch using Radial Basis Function, 2014 IEEE PES General Meeting | Conference & Exposition, National Harbor, MD, pp. 1-5, 2014.
29. P. Sohrabi, B. Jodeiri Shokri, H. Dehghani, Predicting coal price using time series methods and combination of radial basis function (RBF) neural network with time series. Miner Econ 2021.
30. U. Ravale, N. Marathe, P. Padiya, Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel Function, Procedia Computer Science **45**, pp. 428-435, 2015.
31. M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, B. Carro, Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning, IEEE Access **9**, pp. 153153-153170, 2021.
32. L.I. Kuncheva, Initializing of an RBF network by a genetic algorithm, Neurocomputing **14**, pp. 273-288, 1997.
33. F. Ros, M. Pintore, A. Deman, J.R. Chrétien, Automatical initialization of RBF neural networks, Chemometrics and Intelligent Laboratory Systems **87**, pp. 26-32, 2007.
34. D. Wang, X.J. Zeng, J.A. Keane, A clustering algorithm for radial basis function neural network initialization, Neurocomputing **77**, pp. 144-155, 2012.
35. N. Benoudjit, M. Verleysen, On the Kernel Widths in Radial-Basis Function Networks, Neural Processing Letters **18**, pp. 139–154, 2003.
36. R. Neruda, P. Kudova, Learning methods for radial basis function networks, Future Generation Computer Systems **21**, pp. 1131–1142, 2005.
37. E. Ricci, R. Perfetti, Improved pruning strategy for radial basis function networks with dynamic decay adjustment, Neurocomputing **69**, pp. 1728-1732, 2006.
38. Guang-Bin Huang, P. Saratchandran and N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Transactions on Neural Networks **16**, pp. 57-67, 2005.
39. M. Bortman and M. Aladjem, A Growing and Pruning Method for Radial Basis Function Networks, IEEE Transactions on Neural Networks **20**, pp. 1039-1045, 2009.
40. R. Yokota, L.A. Barba, M. G. Knepley, PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians, Computer Methods in Applied Mechanics and Engineering **199**, pp. 1793-1804, 2010.
41. C. Lu, N. Ma, Z. Wang, Fault detection for hydraulic pump based on chaotic parallel RBF network, EURASIP J. Adv. Signal Process. **2011**, 49, 2011.
42. A. Iranmehr, H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive support vector machines, Neurocomputing **343**, pp. 50-64, 2019.
43. J. Cervantes, F.G. Lamont, L.R. Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing **408**, pp. 189-215, 2020.
44. S.B. Kotsiantis, Decision trees: a recent overview, Artif Intell Rev **39**, pp. 261–283, 2013.
45. D. Bertsimas, J. Dunn, Optimal classification trees, Mach Learn **106**, pp. 1039–1082, 2017.
46. Y. Wang, H. Yao, S. Zhao, Auto-encoder based dimensionality reduction, Neurocomputing **184**, pp. 232-242, 2016.
47. V. Agarwal, S. Bhanot, Radial basis function neural network-based face recognition using firefly algorithm, Neural Comput & Applic **30**, pp. 2643–2660, 2018.
48. S. Jiang et al., Prediction of Ecological Pressure on Resource-Based Cities Based on an RBF Neural Network Optimized by an Improved ABC Algorithm, IEEE Access. **7**, pp. 47423-47436, 2019.
49. I.U. Khan,N. Aslam, R. Alshehri, S. Alzahrani, M. Alghamdi, A. Almalki, M. Balabeed, Cervical Cancer Diagnosis Model Using Extreme Gradient Boosting and Bioinspired Firefly Optimization, Scientific Programming **2021**, Article ID 5540024, 2021.
50. K.S. Gyamfi, J. Brusey, E. Gaura, Differential radial basis function network for sequence modelling, Expert Systems with Applications **189**, 115982, 2022.

51. X.Q. Li , L.K. Song, Y.S. Choy, G.C. Bai, Multivariate ensembles-based hierarchical linkage strategy for system reliability evaluation of aeroengine cooling blades, Aerospace Science and Technology **138**, 108325, 2023.
52. J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, No. 14, pp. 281-297, 1967.
53. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5**,pp. 349–358, 2001.
54. H.Q. Wang, D.S. Huang, B. Wang, Optimisation of radial basis function classifiers using simulated annealing algorithm for cancer classification. electronics letters **41**, pp. 630-632, 2005.
55. V. Fathi, G.A. Montazer, An improvement in RBF learning algorithm based on PSO for real time applications, Neurocomputing **111**, pp. 169-176, 2013.
56. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
57. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
58. S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.
59. J.H. Holland, Genetic algorithms. Scientific american 267, pp. 66-73, 1992.
60. J. Stender, Parallel Genetic Algorithms:Theory & Applications. Edition: IOS Press, 1993.
61. J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, pp.125-132.
62. C. Ryan, J. Collins, M. O'Neill, Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391. Springer, Berlin, Heidelberg, 1998.
63. M. O'Neill, M., C. Ryan, Evolving Multi-line Compilable C Programs. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1999. Lecture Notes in Computer Science, vol 1598. Springer, Berlin, Heidelberg, 1999.
64. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
65. A.O. Puente, R. S. Alfonso, M. A. Moreno, Automatic composition of music by means of grammatical evolution, In: APL '02: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications July 2002 Pages 148–155.
66. Lídio Mauro Limade Campo, R. Célio Limã Oliveira,Mauro Roisenberg, Optimization of neural networks through grammatical evolution and a genetic algorithm, Expert Systems with Applications **56**, pp. 368-384, 2016.
67. K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.
68. I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.
69. E. Galván-López, J.M. Swafford, M. O'Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , et al. Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science, vol 6024. Springer, Berlin, Heidelberg, 2010.
70. N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31.
71. D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution for energy demand estimation, Energy Science and Engineering **8**, pp. 1068-1079, 2020.
72. N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems, IEEE Transactions on Evolutionary Computation **17**, pp. 840-861, 2013.
73. C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with grammatical evolution. Sci Rep **12**, 8602, 2022.
74. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
75. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17, pp. 255-287, 2011.
76. Weiss, Sholom M. and Kulikowski, Casimir A., Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann Publishers Inc, 1991.
77. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
78. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
79. Z.H. Zhou,Y. Jiang, NeC4.5: neural ensemble based C4.5," in IEEE Transactions on Knowledge and Data Engineering **16**, pp. 770-773, 2004.
80. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
81. G. Demiroz, H.A. Govenir, N. Ilter, Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals, Artificial Intelligence in Medicine. **13**, pp. 147–165, 1998.

82. B. Hayes-Roth, B., F. Hayes-Roth. Concept learning and the recognition and classification of exemplars. Journal of Verbal Learning and Verbal Behavior **16**, pp. 321-338, 1977.
83. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
84. R.M. French, N. Chater, Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting, Neural Comput. **14**, pp. 1755-1769, 2002.
85. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
86. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
87. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
88. M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med Phys. **34**, pp. 4164-72, 2007.
89. Little MA, McSharry PE, Hunter EJ, Spielman J, Ramig LO. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. 2009;56(4):1015. doi:10.1109/TBME.2008.2005954
90. J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, R.S. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, In: Proceedings of the Symposium on Computer Applications and Medical Care IEEE Computer Society Press, pp.261-265, 1988.
91. D.D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, Y. Zhang, Failure analysis of parameter-induced simulation crashes in climate models, Geoscientific Model Development **6**, pp. 1157-1171, 2013.
92. D. Gavrilis, I.G. Tsoulos, E. Dermatas, Selecting and constructing features using grammatical evolution, Pattern Recognition Letters **29**, pp. 1358-1365, 2008.
93. Giannakeas, N., Tsipouras, M.G., Tzallas, A.T., Kyriakidi, K., Tsianou, Z.E., Manousou, P., Hall, A., Karvounis, E.C., Tsianos, V., Tsianos, E. A clustering based method for collagen proportional area extraction in liver biopsy images (2015) Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
94. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
95. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
96. W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc Natl Acad Sci U S A. **87**, pp. 9193–9196, 1990.
97. M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.
98. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
99. R.G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, Phys. Rev. E **64**, pp. 1-8, 2001.
100. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
101. W. J Nash, T.L. Sellers, S.R. Talbot, A.J. Cawthor, W.B. Ford, The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait, Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288), 1994.
102. T.F. Brooks, D.S. Pope, and A.M. Marcolini. Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
103. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
104. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
105. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
106. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
107. Mackowiak, P.A., Wasserman, S.S., Levine, M.M., 1992. A critical appraisal of 98.6 degrees f, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich. J. Amer. Med. Assoc. 268, 1578–1580
108. R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.
109. M. Sikora, L. Wrobel, Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines, Archives of Mining Sciences **55**, pp. 91-114, 2010.
110. C. Sanderson, R. Curtin, Armadillo: a template-based C++ library for linear algebra, Journal of Open Source Software **1**, pp. 26, 2016.

111. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
112. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.
113. M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591, 1993.
114. D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.
115. Y. Xue, Y. Tong, F. Neri, An ensemble of differential evolution and Adam for training feed-forward neural networks. Information Sciences **608**, pp. 453-471, 2022.
116. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.
117. S. Ding, L. Xu, C. Su et al, An optimizing method of RBF neural network based on genetic algorithm. Neural Comput & Applic 21, pp. 333–336, 2012.
118. W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing **22**, pp. 789-828, 1996.
119. R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.