

Using optimization techniques in Grammatical Evolution

Ioannis G. Tsoulos^{1,*}, Alexandros Tzallas², Evangelos Karvounis³

¹ Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr

² Department of Informatics and Telecommunications, University of Ioannina, Greece; tzallas@uoi.gr

³ Department of Informatics and Telecommunications, University of Ioannina, Greece; ekarvounis@uoi.gr

* Correspondence: itsoulos@uoi.gr;

Abstract: The Grammatical Evolution technique has been successfully applied to a wide range of problems in various scientific fields. However, in many cases, techniques that make use of Grammatical Evolution get trapped in local minima of the objective problem and fail to reach the optimal solution. One simple method to tackle such situations is the usage of hybrid techniques, where local minimization algorithms are used in conjunction with the main algorithm. However, Grammatical Evolution is an integer optimization problem and, as a consequence, techniques should be formulated that are applicable to it as well. In current work, a modified version of the Simulated Annealing algorithm is used as a local optimization procedure in Grammatical Evolution. This approach was tested on the Constructed Neural Networks and a remarkable improvement of the experimental results was shown, both in classification data and in data fitting cases.

Keywords: Grammatical Evolution; Optimization techniques; Neural networks; Evolutionary techniques; Stochastic methods.

1. Introduction

Genetic Algorithms belong to the field of evolutionary techniques [1] and were originally formulated by John Hollands and his team [2]. Genetic Algorithms are initiated by generating a series of random candidate solutions to an optimization problem. These candidate solutions are called chromosomes, and they iteratively undergo a series of operations that have their foundation in physical processes, such as selection, crossover, and mutation [3–5]. Grammatical Evolution [6] is considered as a special case of genetic algorithms, where the chromosomes are series of integer numbers. Chromosomes in the Grammatical Evolution process are rules for generating a Backus–Naur form (BNF) grammar [7] and can be used to create functional programs in any programming language.

Grammatical Evolution has been applied on a wide series of real - world applications, such as function approximation [8,9], credit classification [10], network security and prevention of attacks [11], monitoring of water quality [12], modeling glycemia in humans [13], automatic design of Ant algorithms [14], temperature prediction in data centers [15], solving trigonometric equations [16], composing music [17], neural network construction [18,19], producing numeric constants [20], video games [21,22], energy demand estimation [23], combinatorial optimization [24], cryptography [25], evolving of decision trees [26], automatic design of analog electronic circuits [27] etc.

The method of Grammatical Evolution has been extended by various researchers during the last years and some examples of these extensions are the Structured Grammatical Evolution [28,29], that applies a one-to-one mapping between the chromosomes and the non-terminal symbols of the grammar, the π Grammatical Evolution method [30], application of the Particle Swarm Optimization (PSO) [31] to produce programs with grammatical evolution denoted as Grammatical Swarm [32,33], the Probabilistic Grammatical Evolution [34] which introduced a new mapping mechanism for the Grammatical Evolution method, incorporation of parallel programming techniques [35,36], usage of Christiansen grammars [38] etc.

Citation: Tsoulos, I.G.; Tzallas A; Karvounis E; Using optimization techniques in Grammatical Evolution. *Journal Not Specified* **2022**, *1*, 0. <https://doi.org/>

Received:

Accepted:

Published:

Copyright: © 2024 by the authors. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Also, a variety of software has been developed for Grammatical Evolution, such as the GEVA [37] that proposes a GUI application for Grammatical Evolution, the gramEvol software [39] that provides a package in the R programming language, the GeLab [40] that implements a Matlab toolbox for Grammatical Evolution, the GenClass [41] software used to produce classification programs, the QFc software [42] for feature construction etc.

Although the method of Grammatical Evolution has proven to be extremely efficient, it can often get trapped in local minimum of the objective problem and its performance may not be as expected. A common method to get out of such situations is the use of local optimization techniques, which have been applied many times in combination with Genetic Algorithms [43,44]. In the case of Grammatical Evolution and due to the integral representation of the candidate solutions, local optimization techniques are not directly applicable, such as, for example, the BFGS method [45] and hence more suitable methods should be adopted. In this paper, an attempt is made to improve upon the excellent results shown by Grammatical Evolution in the past on a variety of problems by applying local optimization techniques, periodically applied to randomly selected chromosomes of Grammatical Evolution. The current work utilizes a modified version of the Simulated Annealing method [46] as a local search procedure. The method of Simulated Annealing has been utilized in various cases, such as image processing [47], protein structure optimization [48], resource allocation [49], convex optimization [50], deep learning [51] etc. To verify the possibility of the proposed methodology to improve the results of Grammatical Evolution, the Neural Networks construction technique was chosen. This method was initially provided in [52] and it can estimate the topology and the weights of neural networks using the Grammatical Evolution procedure. This method was chosen to test the present methodology because of its many applications. However, the technique proposed in this paper will be able to be applied in the future in other cases of using Grammatical Evolution. To evaluate the performance of the modified technique it was evaluated on a extended series of classification and regression problems found in the relevant literature and the results seem to be promising.

In the same direction, many researchers have published papers in neural network initialization or construction, such as the usage of decision trees to initialize the weights of a neural network [53], initialization of the weights using Cauchy's inequality [54], application of discriminant learning [55]. Also, the issue of constructing the structure of artificial neural networks has been discussed in various papers, such as incorporation of genetic algorithms [56], construction and pruning of the weights [58], application of cellular automata [59] etc.

This paper has the following sections: section 2 the Grammatical Evolution procedure is discussed and the proposed modification is described, the section 3 presents the datasets used in the experiments as well as the results from the conducted experiments and finally the section 4 provides some conclusions and guidelines for future work.

2. The proposed method

This section starts with a brief description of the Grammatical Evolution process and the grammar used, then the modified Simulated Annealing method is presented, and finally the artificial neural network construction algorithm will be presented.

2.1. The Grammatical Evolution method

The chromosomes of Grammatical Evolution stand for production rules of the given BNF grammar. BNF grammars are usually defined as a tuple $G = (N, T, S, P)$, where

- N is the set of non-terminal symbols.
- T is the set of terminal symbols.
- S represents that start symbol of the grammar with $S \in N$.
- P is the production rules of the grammar. Usually these rules are in the form $A \rightarrow a$ or $A \rightarrow aB$, $A, B \in N$, $a \in T$.

Figure 1. The used grammar in Neural Network Construction.

```

S := <sigexpr> (0)
<sigexpr> ::= <Node> (0)
              | <Node> + <sigexpr> (1)
<Node> ::= <number> * sig(<sum> + <number>) (0)
<sum> ::= <number> * <xxlist> (0)
          | <sum> + <sum> (1)
<xxlist> ::= x1 (0)
            | x2 (1)
            | .....
            | xd (d-1)
<number> ::= (<digitlist> . <digitlist>) (0)
            | (-<digitlist> . <digitlist>) (1)
<digitlist> ::= <digit> (0)
                | <digit> <digitlist> (1)
<digit> ::= 0 (0)
            | 1 (1)
            | .....
            | 9 (9)

```

The BNF grammar is extended by using enumeration in the the production rules. The grammar shown in Figure 1 is used to construct artificial neural networks expressed in the form:

$$N(\vec{x}, \vec{w}) = \sum_{i=1}^H w_{(d+2)i-(d+1)} \sigma \left(\sum_{j=1}^d x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i} \right) \quad (1)$$

where the parameter H represents the number of processing units (hidden noes) for the neural network and the value d represents the of the vector \vec{x} . Also, the vector \vec{w} represents the vector of parameters for the neural network. The function $\text{sig}(x)$ represents the sigmoid function $\sigma(x)$ defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (2)$$

The notation $\langle \rangle$ is used for the non - terminal symbols of the grammar. The sequence numbers of the production rules are inside the parentheses of the extended grammar. The constant d stands for the dimension of the provided dataset. The production of valid expressions initiates from the symbol S and using a series of steps, creates a program by replacing non - terminal symbols with the right hand of the selected production rule. The Grammatical Evolution selects production rules using the following scheme:

- Obtain the next element V from the provided chromosome.
- The production rule is selected according to the scheme

$$\text{Rule} = V \bmod \text{NR} \quad (3)$$

where NR is the total number of production rules for the current non – terminal symbol.

2.2. The modified Simulated Annealing algorithm

The current work utilizes a modified version of the Simulated Annealing algorithm as a local search procedure. Simulated Annealing was chosen as a local search method, since it offers the possibility of representing the considered solutions in an integer form, which is critical for the Grammatical Evolution representation of chromosomes. In addition, this method has been distinguished for its easy adaptation to a multitude of problems but also for its ability to find the total minimum of functions through the point acceptance

mechanism at high temperatures. In the present work, the Simulated Annealing will initiate from the current representation of a chromosome and gradually try to find other nearby representations that might lead to lower values of the fitness value. The main steps of this procedure are shown in Algorithm 1.

Algorithm 1 The modified version of the Simulated Annealing algorithm.

procedure siman(x_0)

1. **Set** $k = 0, T_0 > 0, \epsilon > 0, r_T > 0, r_T < 1$.
2. **Set** $N_{eps} > 0$, a positive integer indicated the number of samples drawn in every iteration.
3. **Set** g and R , positive integer values.
4. **Set** $x_b = x_0, f_b = f(x_b)$.
5. **For** $i = 1 \dots N_{eps}$
 - (a) **Set** $y = x_k$
 - (b) **For** $j = 1 \dots g$
 - i. **Set** $p = \text{rand}(1, \text{size}(x_0))$. The variable p indicates a randomly selected position in y .
 - ii. **Set** $y_p = y_p + \text{rand}(-R, R)$
 - (c) **EndFor**
 - (d) **If** $f(y) \leq f(x_k)$ **then** $x_{k+1} = y$
 - (e) **Else Set** $x_{k+1} = y$ with probability $\min\left\{1, \exp\left(-\frac{f(y)-f(x_k)}{T_k}\right)\right\}$
 - (f) **If** $f(y) < f_b$ **then** $x_b = y, f_b = f(y)$.
6. **EndFor**
7. **Set** $T_{k+1} = T_k r_T$
8. **Set** $k = k + 1$.
9. **If** $T_k \leq \epsilon$ **terminate**
10. **Goto** step 5.
11. **Return** x_b

end siman

The method accepts the chromosome x_0 as a starting point and, in each iteration, randomly generates chromosomes around it. The parameter g controls the number of changes that will be made to the chromosome and the parameter R controls the radius of these changes. The parameter T represents the temperature of the algorithm. The temperature starts at extremely high values and progressively decreases linearly. In the early stages and at high temperatures, the algorithm may accept points that may have higher values of the fitness function, but as the temperature decreases, this probability also decreases. The used simulated annealing variant is also shown graphically as a flow chart in Figure 2.

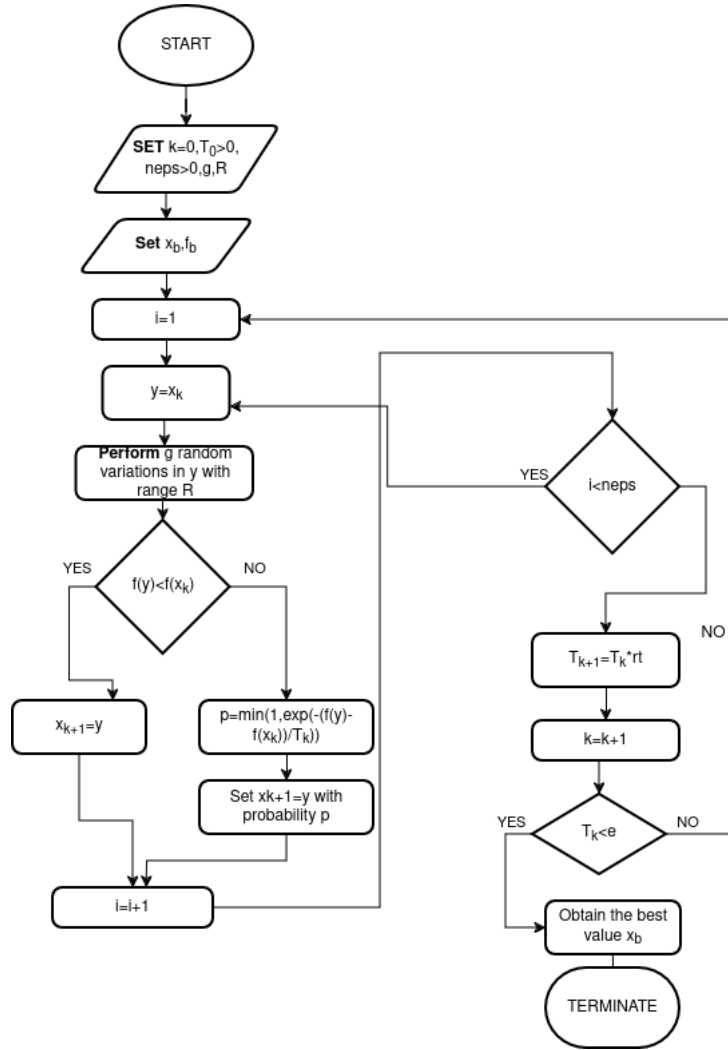


Figure 2. Flowchart of the used Simulated Annealing variant.

2.3. The Neural Network Construction algorithm

The main steps of the algorithm used to construct artificial neural networks with Grammatical Evolution are listed below:

1. Initialization step.

- (a) **Set** N_g as the maximum number of generations allowed.
- (b) **Set** N_c as the number of chromosomes.
- (c) **Set** p_s as the selection rate and p_m as the mutation rate.
- (d) **Define** as L_i the number of generations that should elapse before applying the local optimization technique.
- (e) **Define** as L_c the number of chromosomes involved in the local search procedure.
- (f) **Initialize** the chromosomes. Each chromosome is considered as a series of randomly initialized integers.
- (g) **Set** iter=0

2. Genetic step

- (a) **For** $i = 1, \dots, N_g$ **do**
 - i. **Create** for every chromosome a neural network C_i using the Grammatical Evolution procedure of subsection 2.1 and the associated grammar given in Figure 1.

- ii. **Calculate** the fitness f_i on the train set of the objective problem as:

$$f_i = \sum_{j=1}^M (C_i(\vec{x}_j) - t_j)^2 \quad (4)$$

where the set (\vec{x}_j, t_j) , $j = 1, \dots, M$ is train dataset, with t_j being the actual output for the point \vec{x}_j .

- iii. **Perform** the selection procedure. Initially, the chromosomes are sorted according to their fitness values. The best $(1 - p_s) \times N_c$ chromosomes are transferred to the next generation. The remaining chromosomes will be replaced by offsprings created during the crossover procedure.
- iv. **Perform** the crossover procedure. The crossover procedure produces $p_s \times N_c$ offsprings. For every pair of produced offsprings \tilde{z} and \tilde{w} , two offsprings (z, w) . The selection is performed using tournament selection. The new offsprings are produced using the one - point crossover procedure. An example of the one - point crossover procedure is shown in Figure 3.
- v. **Perform** the mutation procedure. For each element of every chromosome, a random number $r \in [0, 1]$ is drawn. The corresponding element is altered if $r \leq p_m$.

(b) **EndFor**

3. **Local Search Step**

(a) **If** $\text{iters} \bmod L_i = 0$ **Then**

- i. **Create** a set of randomly L_C chosen chromosomes from the genetic population. Denote this set as L_S .
- ii. **For every** X_k **in** L_S **apply** the modified Simulated Annealing algorithm given in Algorithm 1: $X_k = \text{siman}(X_k)$

4. **set** $\text{iter} = \text{iter} + 1$. **If** $\text{iter} > N_g$ **goto** Evaluation step **else goto** Genetic step.

5. **Evaluation step**

- (a) Get the chromosome with the lowest fitness value and create the associated neural network C^* .
- (b) Evaluate the neural network C^* in the test set of the underlying dataset and report the results.

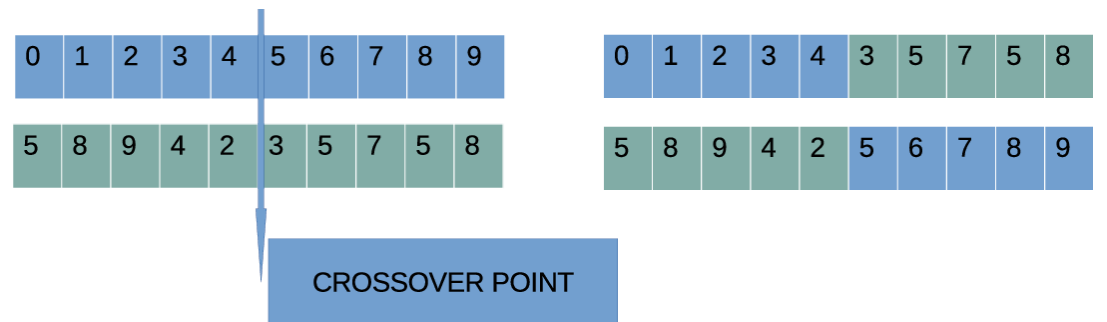


Figure 3. An example of the one - point crossover method, used in the Grammatical Evolution procedure.

3. Results

The current work was evaluated by executing a series of experiments on some classification and regression datasets, commonly used in the relevant literature. The obtained

results were compared with other machine learning techniques. These datasets can be downloaded freely from the following websites:

1. The UCI dataset repository, <https://archive.ics.uci.edu/ml/index.php> (accessed on 20 March 2024) [60]
2. The Keel repository, <https://sci2s.ugr.es/keel/datasets.php> (accessed on 20 March 2024) [61].
3. The Statlib URL <http://lib.stat.cmu.edu/datasets/> (accessed on 20 March 2024).

3.1. Classification datasets

The following series of classification datasets was used in the conducted experiments:

1. **Appendictis** a medical dataset, provided in [62].
2. **Australian** dataset [63], which is related to credit card transactions.
3. **Balance** dataset [64], a dataset related to psychological experiments.
4. **Circular** dataset, an artificial dataset that contains 1000 examples.
5. **Cleveland** dataset, a medical dataset used in a series of papers [65,66].
6. **Dermatology** dataset [67], which is a medical dataset about dermatological deceases.
7. **Ecoli** dataset, a dataset about protein localization sites [68].
8. **Haberman** dataset, related to breast cancer.
9. **Heart** dataset [69], a medical dataset about heart diseases.
10. **Hayes roth** dataset [70], which is a human subjects study.
11. **HouseVotes** dataset [71], related to votes collected from U.S. House of Representatives Congressmen.
12. **Ionosphere** dataset, that was used in experiments related to the ionosphere [72,73].
13. **Liverdisorder** dataset [74], a medical dataset related to liver disorders.
14. **Mammographic** dataset [75], a medical dataset related to breast cancer.
15. **Parkinsons** dataset, a medical dataset related to the Parkinson's disease (PD) [76].
16. **Pima** dataset [77], a medical dataset related to the diabetes presence.
17. **Popfailures** dataset [78], a dataset related to climate measurements.
18. **Regions2** dataset, medical dataset related to hepatitis C [79].
19. **Saheart** dataset [80]. This dataset is used to detect heart diseases.
20. **Segment** dataset [81], related to image processing.
21. **Student** dataset [82], related to data collected in Portuguese schools.
22. **Transfusion** dataset [83], this datasets was taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan.
23. **Wdbc** dataset [84], a medical dataset related cancer detection.
24. **Wine** dataset. This is a dataset used to detect the quality of a series of wines. [85,86].
25. **Eeg** datasets, a dataset related to EEG measurements [87]. From this dataset the following cases were used: Z_F_S, Z_O_N_F_S, ZO_NF_S and ZONF_S.
26. **Zoo** dataset [88], related to animal classification.

3.2. Regression datasets

The following regression datasets were used in the conducted experiments:

1. **Abalone** dataset [89], a dataset related to the prediction of age of abalones.
2. **Airfoil** dataset, a dataset proposed by NASA [90].
3. **Baseball** dataset, related with the income of baseball players.
4. **Concrete** dataset [91], which is a civil engineering dataset.
5. **Dee** dataset. This dataset has measures from the price of electricity.
6. **HO** dataset, downloaded from the STALIB repository.
7. **Housing** dataset, mentioned in [92].
8. **Laser** dataset. This is a dataset related to laser experiments
9. **LW** dataset, related to risk factors associated with low weight babies.
10. **MORTGAGE** dataset, a dataset related to economic measurements from USA.
11. **PL** dataset, provided from the STALIB repository.

12. **SN** dataset, provided from the STALIB repository.
13. **Treasury** dataset, a dataset related to economic measurements from USA.
14. **TZ** dataset, provided from the STALIB repository.

3.3. Experimental results

For the execution of the experiments, code written in ANSI C++ was used and, with the help of the programming environment Optimus. The software is freely available from <https://github.com/itsoulos/OPTIMUS/> (accessed on 20 March 2024). The experiments were conducted 30 times. In every execution different seed was used for the random number generator and the function `drand48()` of the C programming language was used. The validation of the results was performed using the technique of 10 - fold cross validation. The average classification error is reported for the case of classification datasets and the average regression error for the case of regression datasets. These errors are measured on the corresponding test set. The values for the experimental parameters are displayed in Table 1. The experimental results for the classification datasets are outlined in the table 2 and the results for the regression datasets are shown in the table 3. The following applies to the tables with the experimental results:

1. The column DATASET denotes the used dataset.
2. The column ADAM denotes the application of the ADAM optimization method [93] in an artificial neural network with $H = 10$ processing nodes.
3. The column NEAT stands for the usage of NEAT method (NeuroEvolution of Augmenting Topologies) [94].
4. The column MLP stands for the experimental results of an artificial neural network with $H = 10$ processing nodes. The neural network was trained using a genetic algorithm and the BFGS local optimization method [95].
5. The column RBF represents the application of an RBF network with $H = 10$ processing nodes in each dataset.
6. The column NNC denotes the usage of the original Neural Construction technique, that was constructed with Grammatical Evolution.
7. The column NNC-S denotes the usage of the proposed local optimization procedure in the Neural Construction technique.
8. The line AVERAGE denotes the average classification or regression error.

NAME	PURPOSE	VALUE
N_c	Number of chromosomes	500
N_g	Number of generations	200
p_s	Selection rate	0.10
p_m	Mutation rate	0.05
g	Number of random changes	10
R	Range of random changes	10
ϵ	Small value used in comparisons	10^{-5}
N_{eps}	Number of random samples	200
T	Initial temperature	10^8
r_T	Rate of decrease in temperature	0.8

Table 1. The values for the parameters used in the conducted experiments.

Table 2. Experimental results for the series of machine learning methods for the classification datasets. Each number in cells stands for the average classification error as measured in the test set.

DATASET	ADAM	NEAT	GENETIC	RBF	NNC	NNC-S
APPENDICITIS	16.50%	17.20%	18.10%	12.23%	14.40%	14.60%
AUSTRALIAN	35.65%	31.98%	32.21%	34.89%	14.46%	14.90%
BALANCE	7.87%	23.84%	8.97%	33.42%	22.13%	7.66%
CIRCULAR	3.94%	34.07%	5.99%	6.30%	14.26%	7.88%
CLEVELAND	67.55%	53.44%	51.60%	67.10%	49.93%	48.59%
DERMATOLOGY	26.14%	32.43%	30.58%	62.34%	24.80%	13.11%
ECOLI	64.43%	43.24%	49.38%	59.50%	48.82%	44.88%
HABERMAN	29.00%	24.04%	28.66%	25.10%	28.33%	28.73%
HAYES ROTH	59.70%	50.15%	56.18%	64.36%	37.23%	28.08%
HEART	38.53%	39.27%	28.34%	31.20%	15.78%	16.00%
HOUSEVOTES	7.48%	10.89%	6.62%	6.13%	3.52%	3.74%
IONOSPHERE	16.64%	19.67%	15.14%	16.22%	11.86%	10.03%
LIVERDISORDER	41.53%	30.67%	31.11%	30.84%	32.97%	32.82%
MAMMOGRAPHIC	46.25%	22.85%	19.88%	21.38%	18.22%	16.58%
PARKINSONS	24.06%	18.56%	18.05%	17.41%	13.21%	12.26%
PIMA	34.85%	34.51%	32.19%	25.78%	28.47%	25.26%
POPFAILURES	5.18%	7.05%	5.94%	7.04%	6.83%	5.52%
REGIONS2	29.85%	33.23%	29.39%	38.29%	25.87%	24.47%
SAHEART	34.04%	34.51%	34.86%	32.19%	30.80%	29.52%
SEGMENT	49.75%	66.72%	57.72%	59.68%	54.89%	39.38%
STUDENT	5.13%	12.50%	5.61%	7.52%	5.70%	4.52%
TRANSFUSION	25.68%	24.87%	25.84%	27.36%	25.30%	24.33%
WDBC	35.35%	12.88%	8.56%	7.27%	7.27%	5.59%
WINE	29.40%	25.43%	19.20%	31.41%	13.53%	11.47%
Z_F_S	47.81%	38.41%	10.73%	13.16%	15.30%	7.93%
Z_O_N_F_S	78.79%	79.08%	64.81%	60.40%	50.48%	40.42%
ZO_NF_S	47.43%	43.75%	8.41%	9.02%	15.22%	6.60%
ZONF_S	11.99%	5.44%	2.60%	4.03%	3.14%	2.36%
ZOO	14.13%	20.27%	16.67%	21.93%	9.10%	7.20%
AVERAGE	32.23%	30.72%	24.94%	28.74%	22.13%	18.43%

Table 3. Experimental results as measured on the regression datasets. Each number in cells denote the average regression error for the associated machine learning method, as measured on the test set.

DATASET	ADAM	NEAT	GENETIC	RBF	NNC	NNC-S
ABALONE	4.30	9.88	7.17	7.37	5.11	4.95
AIRFOIL	0.005	0.067	0.003	0.27	0.003	0.003
BASEBALL	77.90	100.39	103.60	93.02	59.40	57.30
CONCRETE	0.078	0.081	0.0099	0.011	0.008	0.006
DEE	0.63	1.512	1.013	0.17	0.26	0.23
HO	0.035	0.167	2.78	0.03	0.016	0.012
HOUSING	80.20	56.49	43.26	57.68	25.56	18.82
LASER	0.03	0.084	0.59	0.024	0.026	0.015
LW	0.028	0.17	1.90	1.14	0.97	0.038
MORTGAGE	9.24	14.11	2.41	1.45	0.29	0.12
PL	0.117	0.097	0.28	0.083	0.046	0.033
SN	0.026	0.174	2.95	0.90	0.026	0.024
TREASURY	11.16	15.52	2.93	2.02	0.47	0.18
TZ	0.07	0.097	5.38	4.10	0.06	0.028
AVERAGE	13.12	14.20	12.45	12.02	6.60	5.84

The above techniques were used in the experiments as they are widespread in machine learning, such as the Adam method [96,97] and because they have a similar complexity to the present technique, such as Genetic Algorithms. As it is evident, the proposed modification improves the efficiency of the proposed method in the majority of used datasets. This improvement on some datasets could be as much as an 80% percent error reduction on the test set. In all cases, the simple Neural Network Construction technique outperforms other machine learning techniques and this is evident from the average result (the last row in experimental tables). However, the proposed method significantly reduces the classification error or the error in the data fitting sets in most of the cases where it was used. In fact, in order to show in which cases there was a reduction in the error, bold marking was used in the tables of results.

Figure 4 is a scatter plot that provides a detailed comparative analysis of classification error percentages for six distinct machine learning and optimization algorithms: ADAM, NEAT, GENETIC, RBF, NNC, and NNC-S. Each point on the plot represents the outcome of an individual run of a model, thus showcasing the range of variation in the error rates associated with these classification methods. The vertical dispersion of points for each method reflects the spread of error rates, which is critical for evaluating the reliability of each algorithm. The medians of these error rates are indicated by horizontal lines intersecting the clusters of dots, offering a snapshot of each algorithm's central tendency in performance. The asterisk-based notation above the clusters denotes statistical significance levels: one asterisk (*) signifies a p-value less than 0.05, two asterisks (**) denote $p < 0.01$, three asterisks (***) represent $p < 0.001$, and four asterisks (****) indicate an extremely low p-value of less than 0.0001, suggesting strong evidence against the null hypothesis. Of note is the performance of the NNC-S method, which not only shows a significantly lower median error rate when compared to the NNC method but also displays a notably tighter cluster of data points. This implies that the NNC-S method not only tends to be more accurate on average but also provides greater consistency in its error rates across different runs, underscoring its robustness as a classification tool.

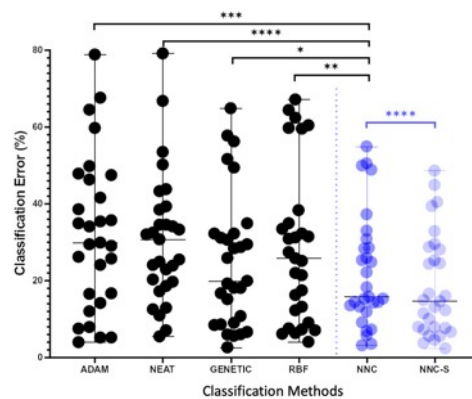


Figure 4. Scatter plot illustrating the variability and median classification error rates for six machine learning algorithms, with statistical significance denoted by asterisks. The NNC-S method demonstrates notably lower error rates, as evidenced by the high statistical significance relative to other methods, which may indicate its superior performance in classification tasks.

Building on the previous analysis of classification methods, Figure 5 extends the evaluation to regression algorithms, providing a comparative study of regression error rates for the same six methods: ADAM, NEAT, GENETIC, RBF, NNC, and NNC-S. Each data point reflects the regression error from a specific trial, with the array of points for each method revealing the range of performance outcomes. The median error rates are again represented by horizontal lines across the clusters of dots, serving as a summary statistic that facilitates a direct comparison of the method's central performance trends. The notation of statistical significance is consistent with the previous figure, where asterisks

convey the p-value levels, identifying statistically meaningful differences in performance between the methods.

This plot reveals that the NNC-S method maintains its superior performance in the context of regression tasks, demonstrating lower median regression errors compared to the other methods. Significantly, it achieves a markedly lower median regression error than its predecessor, NNC, as indicated by the blue dots and supported by the three asterisks (**). This pattern of results underscores the broader applicability of the NNC-S method's local optimization enhancements, not only in classification accuracy but also in reducing regression errors.

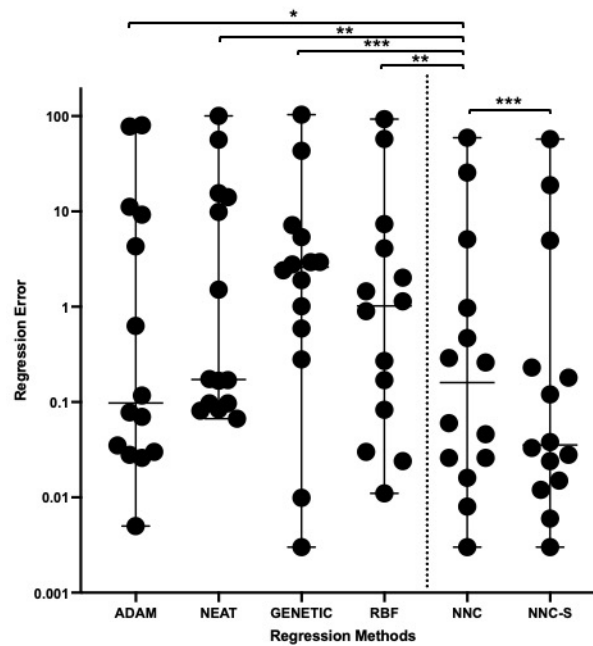


Figure 5. Scatter plot of regression error rates for various regression methods, demonstrating the distribution, median error rates, and statistical significance of differences in performance. The NNC-S method, shows statistically significant improvements in accuracy over the NNC method, underlining the efficacy of local optimization enhancements in regression tasks. The y-axis is presented on a logarithmic scale to clearly visualize the range of error magnitudes across methods.

Additionally, the effectiveness and the robustness of the proposed method was evaluated by performing additional experiments with different values for the critical parameter g of the suggested Simulated Annealing variant. This parameter is used to control the number of random changes performed on any given chromosome. The experimental results in the classification datasets are shown in Table 4. Clearly, no significant variation in the performance of the proposed technique occurs when this critical parameter is varied.

Table 4. Experiments with the parameter g used in the modified Simulated Annealing method.

DATASET	NNC-S($g = 2$)	NNC-S($g = 5$)	NNC-S($g = 10$)	NNC-S($g = 20$)
APPENDICITIS	14.90%	15.00%	14.60%	14.50%
AUSTRALIAN	14.59%	14.85%	14.90%	15.04%
BALANCE	8.53%	7.68%	7.66%	7.56%
CIRCULAR	10.49%	8.81%	7.88%	7.50%
CLEVELAND	48.41%	48.31%	48.59%	48.10%
DERMATOLOGY	15.09%	13.80%	13.11%	13.29%
ECOLI	45.30%	45.12%	44.88%	44.36%
HABERMAN	27.80%	27.97%	28.73%	27.83%
HAYES ROTH	29.85%	28.85%	28.08%	29.15%
HEART	16.11%	15.04%	16.00%	14.78%
HOUSEVOTES	3.70%	4.22%	3.74%	3.70%
IONOSPHERE	10.54%	10.09%	10.03%	10.00%
LIVERDISORDER	31.41%	33.15%	32.82%	33.29%
MAMMOGRAPHIC	17.16%	17.25%	16.58%	16.99%
PARKINSONS	12.32%	12.89%	12.26%	12.11%
PIMA	26.12%	25.96%	25.26%	25.92%
POPFAILURES	5.58%	6.00%	5.52%	5.68%
REGIONS2	24.71%	24.05%	24.47%	24.66%
SAHEART	30.04%	29.67%	29.52%	29.07%
SEGMENT	46.94%	42.37%	39.38%	41.19%
STUDENT	4.60%	4.73%	4.52%	4.48%
TRANSFUSION	24.28%	24.34%	24.33%	24.03%
WDBC	6.23%	6.22%	5.59%	5.68%
WINE	12.59%	11.30%	11.47%	9.24%
Z_F_S	9.57%	9.60%	7.93%	8.10%
Z_O_N_F_S	46.04%	43.36%	40.42%	41.54%
ZO_NF_S	9.69%	8.54%	6.60%	6.44%
ZONF_S	2.58%	2.28%	2.36%	2.36%
ZOO	6.90%	7.00%	7.20%	7.70%
AVERAGE	19.38%	18.91%	18.43%	18.42%

Continuing from the previous analysis, Figure 6 presents the classification error rates for variations of the NNC-S algorithm with differing values of the parameter g . The plot aims to evaluate whether changes in the parameter g led to statistically significant differences in the algorithm's classification performance. The horizontal bars on the plot indicate the median classification error for each variation of the NNC-S method, providing a clear comparison across the different parameter values. The statistical annotations ("ns" for not significant, "***" for $p < 0.01$, and "****" for $p < 0.001$) are used to denote the statistical significance of the differences between the groups. It appears that some variations, particularly between NNC-S ($g=2$) and NNC-S ($g=5$), as well as between NNC-S ($g=10$) and NNC-S ($g=20$), do not show significant differences in performance (denoted by "ns"). In contrast, other comparisons do reveal significant differences, suggesting that certain values of g can indeed impact the classification error rates of the NNC-S algorithm.

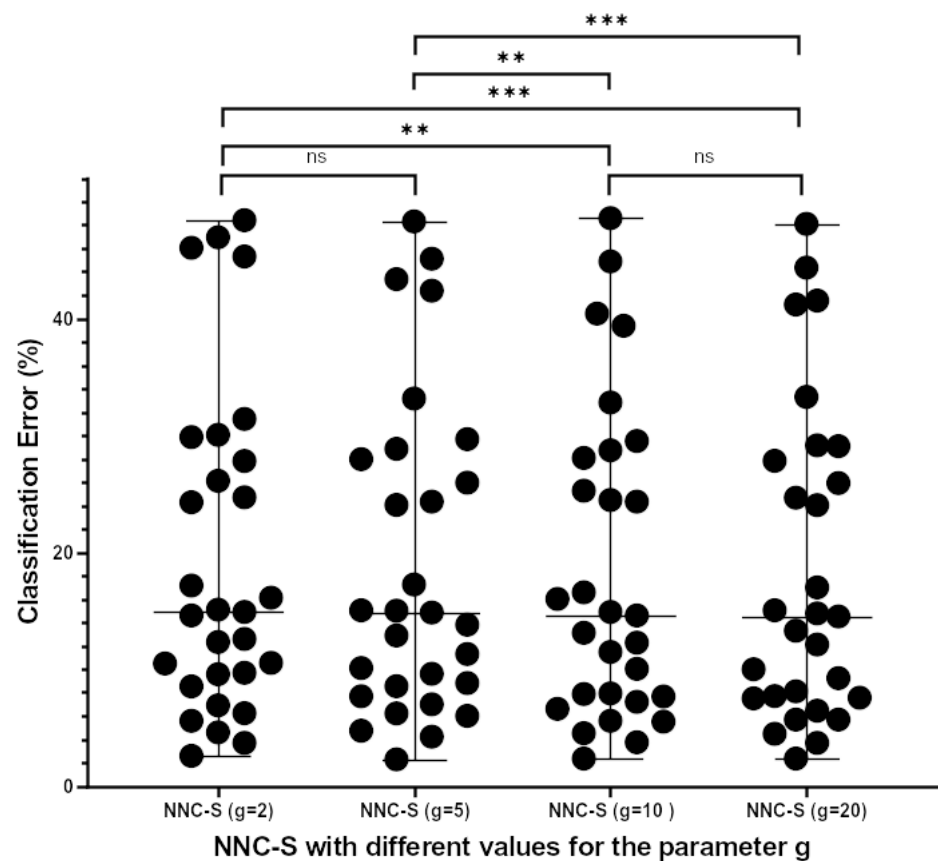


Figure 6. his figure examines the impact of varying the parameter g on the NNC-S algorithm's classification errors, revealing that while some parameter adjustments do not significantly affect performance, others result in noticeable differences, as indicated by the statistical significance annotations.

In addition, in the graph of the figure 7, a comparison is made for the average execution time of each experiment for the classification datasets. The comparison was made between the initial method (denoted as NNC in the graph) as well as the various cases of the proposed technique by changing the parameter g to 5, 10 and 20. As expected, adding the local minimization technique to the Grammatical Evolution method significantly increases the execution time of the method, however this increase remains almost constant for different values of the critical parameter g . Moreover, by using parallel techniques that appear in the relevant literature this increase in execution time could be reduced.

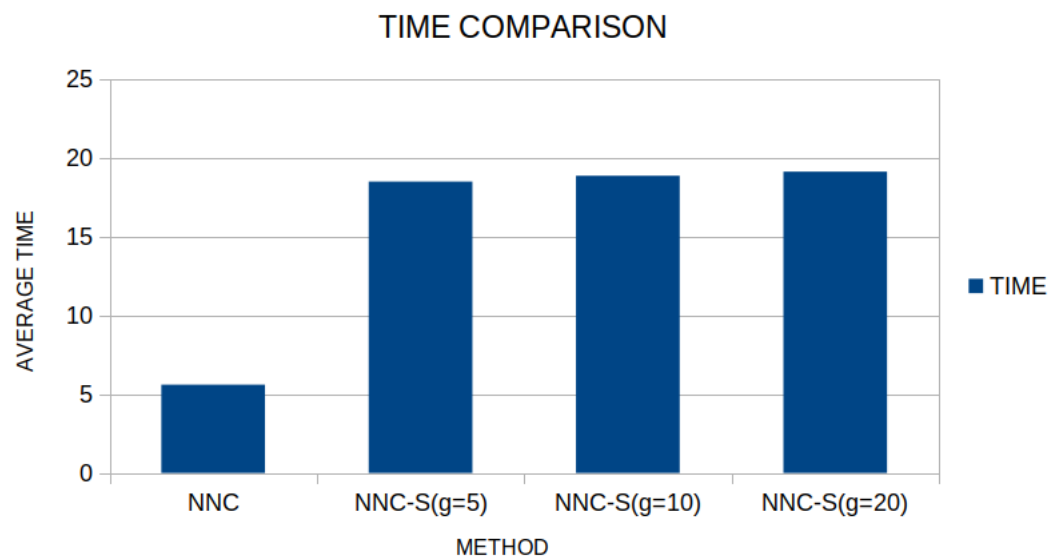


Figure 7. Time comparison between the initial method and the proposed modifications.

Furthermore, another experiment was executed by varying the parameter R of the proposed Simulated Annealing variant. This parameter controls the range of changes performed in any given chromosome. The results for this experiment are shown in Table 5. And this time the method appeared quite robust in its performance, without large variations in the error as measured in the test set.

Table 5. Experiments with the parameter R of the modified Simulated Annealing algorithm. The parameter g was set to 10.

DATASET	NNC-S($R = 2$)	NNC-S($R = 5$)	NNC-S($R = 10$)	NNC-S($R = 20$)
APPENDICITIS	14.40%	14.90%	14.60%	15.20%
AUSTRALIAN	14.77%	14.78%	14.90%	14.70%
BALANCE	7.66%	7.74%	7.66%	7.66%
CIRCULAR	8.39%	8.29%	7.88%	7.78%
CLEVELAND	49.45%	49.28%	48.59%	47.11%
DERMATOLOGY	14.09%	12.54%	13.11%	11.34%
ECOLI	44.24%	46.30%	44.88%	44.48%
HABERMAN	27.33%	28.10%	28.73%	28.04%
HAYES ROTH	29.15%	27.92%	28.08%	26.46%
HEART	15.67%	15.52%	16.00%	15.15%
HOUSEVOTES	4.00%	3.62%	3.74%	4.52%
IONOSPHERE	10.14%	10.03%	10.03%	10.71%
LIVERDISORDER	32.80%	32.12%	32.82%	32.29%
MAMMOGRAPHIC	17.18%	16.78%	16.58%	16.62%
PARKINSONS	12.68%	12.16%	12.26%	11.95%
PIMA	25.72%	25.11%	25.26%	26.33%
POPFAILURES	5.87%	5.72%	5.52%	5.58%
REGIONS2	23.55%	24.04%	24.47%	24.08%
SAHEART	29.48%	28.96%	29.52%	29.24%
SEGMENT	40.32%	40.23%	39.38%	40.82%
STUDENT	4.18%	4.50%	4.52%	4.78%
TRANSFUSION	24.60%	24.12%	24.33%	24.36%
WDBC	6.09%	5.68%	5.59%	5.46%
WINE	11.00%	10.30%	11.47%	9.41%
Z_F_S	8.33%	8.30%	7.93%	8.50%
Z_O_N_F_S	41.70%	43.42%	40.42%	41.44%
ZO_NF_S	7.58%	7.72%	6.60%	7.10%
ZONF_S	2.50%	2.54%	2.36%	3.00%
ZOO	6.80%	6.30%	7.20%	6.10%
AVERAGE	18.61%	18.52%	18.43%	18.28%

As a practical application of the suggested method, consider the dataset proposed in [98], which relates the alcohol consumption and EEG recordings. The proposed method was applied on this dataset and it is compared against the original Neural Network Construction technique and the results are outlined in Figure 8.

337
338
339
340

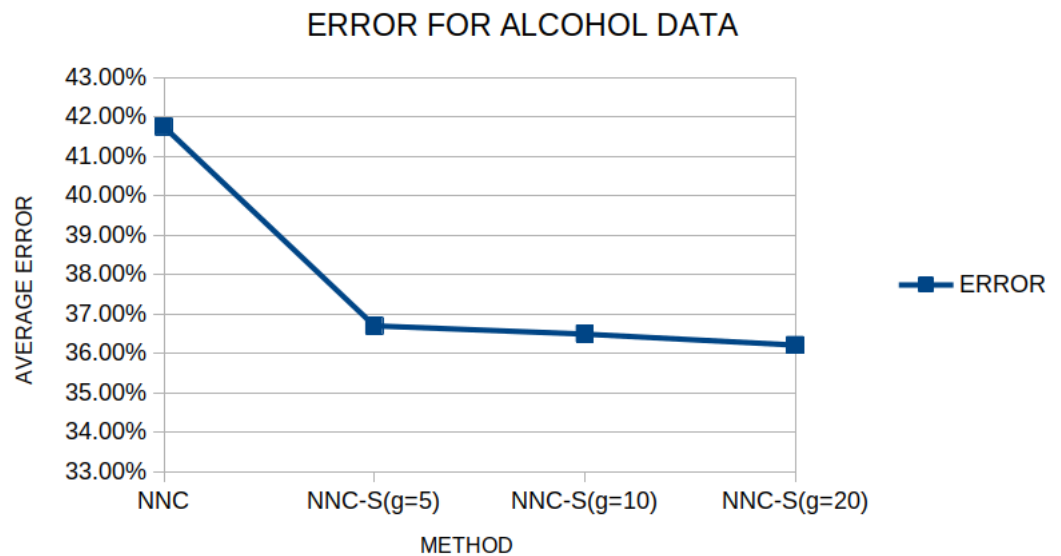


Figure 8. Comparison between the proposed method and the original Neural Construction method for the alcohol consumption dataset.

Once again, the present method outperforms the original artificial neural network construction technique and the performance appears to be stable as the critical parameter g grows.

4. Conclusions

In the current work, an extension of the original Grammatical Evolution has been proposed, which was applied in the Neural Construction method. In this extension, an application of a modified optimization method was suggested, in order to improve the efficiency of the underlying technique. The proposed optimization algorithm was a variant of the Simulated Annealing method, that was applied on a series of chromosomes, that was randomly from the Grammatical Evolution procedure. This method was chosen because of its widespread use in many applications but because of its ability to handle integer representations, since the chromosomes in Grammatical Evolution are vectors of integers. Of course, other local optimization techniques could be incorporated, such as tabu search [99] or hill climbing [100] as feature extensions of the proposed method.

The proposed modification was applied to the Neural Network Construction method and its efficiency was measured on some classification and regression datasets commonly used. Based on the experimental results, it has become clear that the proposed variant significantly improves the performance of the technique both on classification datasets and on data fitting datasets. Also, the effectiveness and the robustness of the proposed modification were measured using experiments with different values of some critical parameters of the current Simulated Annealing variant. These experiments indicated that the method tends to be robust, since the experimental results does not depend on the selection of any critical parameter of the Simulated Annealing variant.

The present modification could be applied to other techniques that use Grammatical Evolution without significant differences and, moreover, it could also be used to optimize functions either without constraints or with constraints in similar techniques that have been proposed in recent years. However, as was seen from the experimental results, the addition of the Simulated Annealing technique significantly increases the required execution time and it is necessary to use techniques that are not particularly demanding in time or to search for termination techniques of the proposed Simulated Annealing variant that take advantage of its particular characteristics. Also, a field of research could be the search for

more effective techniques to reduce the critical temperature factor used in the Simulated Annealing.

Future work may include application of the local search procedure in other Grammatical Evolution applications, such as feature construction, creation of classification rules etc. Also, the proposed process can be significantly accelerated by the use of parallel optimization techniques [101], which take advantage of modern computational structures.

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. conducted the experiments, employing several datasets, and provided the comparative experiments. A.T. performed the statistical analysis. E.K. and all other authors prepared the manuscript. E.K. and I.G.T. organized the research team and A.T. supervised the project. 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.

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. N. Yusup, A. M. Zain, S. Z. M. Hashim, Evolutionary techniques in optimizing machining parameters: Review and recent applications (2007–2011), *Expert Systems with Applications* **39.10**, pp. 9909–9927, 2012.
2. J.H. Holland, Genetic algorithms. *Scientific american* **267**, pp. 66–73, 1992.
3. J. Stender, *Parallel Genetic Algorithms: Theory & Applications*. Edition: IOS Press, 1993.
4. D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
5. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer - Verlag, Berlin, 1996.
6. M. O'Neill, C. Ryan, Grammatical evolution, *IEEE Trans. Evol. Comput.* **5**, pp. 349–358, 2001.
7. 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.
8. 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.
9. 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.
10. A. Brabazon, M. O'Neill, Credit classification using grammatical evolution, *Informatica* **30.3**, 2006.
11. S. Şen, J.A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks, In: *Proceedings of the second ACM conference on Wireless network security*, 2009.
12. L. Chen, C.H. Tan, S.J. Kao, T.S. Wang, Improvement of remote monitoring on water quality in a subtropical reservoir by incorporating grammatical evolution with parallel genetic algorithms into satellite imagery, *Water Research* **42**, pp. 296–306, 2008.
13. J. I. Hidalgo, J. M. Colmenar, J.L. Risco-Martin, A. Cuesta-Infante, E. Maqueda, M. Botella, J. A. Rubio, Modeling glycemia in humans by means of Grammatical Evolution, *Applied Soft Computing* **20**, pp. 40–53, 2014.
14. J. Tavares, F.B. Pereira, Automatic Design of Ant Algorithms with Grammatical Evolution. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds) *Genetic Programming. EuroGP 2012*. Lecture Notes in Computer Science, vol 7244. Springer, Berlin, Heidelberg, 2012.
15. M. Zapater, J.L. Risco-Martín, P. Arroba, J.L. Ayala, J.M. Moya, R. Hermida, Runtime data center temperature prediction using Grammatical Evolution techniques, *Applied Soft Computing* **49**, pp. 94–107, 2016.
16. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, *proceedings of Mendel*. Vol. 98. 1998.
17. 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.
18. 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.
19. K. Soltanian, A. Ebnehasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, *Evolutionary Computation* **30**, pp 291–327, 2022.

20. I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, *International Journal of Innovative Computing and Applications* **1**, pp 23–38, 2007. 426
21. E. Galván-López, J.M. Swafford, M. O'Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , 427
et al. *Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science*, vol 6024. Springer, 428
Berlin, Heidelberg, 2010. 429
22. N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 430
2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31. 431
23. D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution 432
for energy demand estimation, *Energy Science and Engineering* **8**, pp. 1068-1079, 2020. 433
24. N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems, *IEEE* 434
Transactions on Evolutionary Computation **17**, pp. 840-861, 2013. 435
25. C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with 436
grammatical evolution. *Sci Rep* **12**, 8602, 2022. 437
26. P.J. Pereira, P. Cortez, R. Mendes, Multi-objective Grammatical Evolution of Decision Trees for Mobile Marketing user conversion 438
prediction, *Expert Systems with Applications* **168**, 114287, 2021. 439
27. F. Castejón, E.J. Carmona, Automatic design of analog electronic circuits using grammatical evolution, *Applied Soft Computing* **62**, 440
pp. 1003-1018, 2018. 441
28. N. Lourenço, F.B. Pereira, E. Costa, Unveiling the properties of structured grammatical evolution, *Genetic Programming and* 442
Evolvable Machines **17**, pp. 251-289, 2016. 443
29. N. Lourenço, F. Assunção, F.B. Pereira, E. Costa, P. Machado, Structured grammatical evolution: a dynamic approach, *Handbook* 444
of grammatical evolution, pp. 137-161, 2018. 445
30. M. O'Neill, A. Brabazon, M. Nicolau, S.M. Garraghy, P. Keenan, π Grammatical Evolution. In: Deb, K. (eds) *Genetic and* 446
Evolutionary Computation – GECCO 2004. GECCO 2004. Lecture Notes in Computer Science, vol 3103. Springer, Berlin, 447
Heidelberg, 2004. 448
31. Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, *Swarm Intelligence* **1**, pp 33-57, 449
2007. 450
32. M. O'Neill, A. Brabazon, Grammatical swarm: The generation of programs by social programming. *Natural Computing* **5**, pp. 451
443-462, 2006. 452
33. E. Ferrante, E. Duéñez-Guzmán, A.E. Turgut, T. Wenseleers, GESwarm: Grammatical evolution for the automatic synthesis of 453
collective behaviors in swarm robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 454
pp. 17-24, 2013. 455
34. J. Mégane, N. Lourenço, P. Machado, Probabilistic Grammatical Evolution. In: Hu, T., Lourenço, N., Medvet, E. (eds) *Genetic* 456
Programming. EuroGP 2021. Lecture Notes in Computer Science, vol 12691. Springer, Cham, 2021. 457
35. O. Popelka, P. Osmera, Parallel Grammatical Evolution for Circuit Optimization. In: Hornby, G.S., Sekanina, L., Haddow, P.C. 458
(eds) *Evolvable Systems: From Biology to Hardware. ICES 2008. Lecture Notes in Computer Science*, vol 5216. Springer, Berlin, 459
Heidelberg. https://doi.org/10.1007/978-3-540-85857-7_40 460
36. P. Ošmera, Two level parallel grammatical evolution, *Advances in Computational Algorithms and Data Analysis* pp. 509-525, 461
2009. 462
37. M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, A. Brabazon, GEVA: grammatical evolution in Java. *ACM* 463
SIGEVolution **3**, pp. 17-22, 2008. 464
38. A. Ortega, M. de la Cruz, M. Alfonseca, Christiansen Grammar Evolution: Grammatical Evolution With Semantics, *IEEE* 465
Transactions on Evolutionary Computation **11**, pp. 77-90, Feb. 2007. 466
39. F. Noorian, A.M. de Silva, P.H.W. Leong, gramEvol: Grammatical Evolution in R, *Journal of Statistical Software* **71**, pp. 1–26, 2016. 467
40. M.A. Raja, C. Ryan, GELAB - A Matlab Toolbox for Grammatical Evolution. In: Yin, H., Camacho, D., Novais, P., Tallón-Ballesteros, 468
A. (eds) *Intelligent Data Engineering and Automated Learning – IDEAL 2018. IDEAL 2018. Lecture Notes in Computer Science()*, 469
vol 11315, 2018. Springer, Cham. https://doi.org/10.1007/978-3-030-03496-2_22 470
41. N.Anastasopoulos, I.G. Tsoulos, A. Tzallas, GenClass: A parallel tool for data classification based on Grammatical Evolution, 471
SoftwareX **16**, 100830, 2021. 472
42. I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, *Algorithms* **15**, 295, 2022. 473
43. S. Yang, S. N. Jat, Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling, *IEEE* 474
Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **41**, pp. 93-106, 2011. 475
44. M. Sivaram, K. Batri, A.S. Mohammed, V. Porkodi, Exploiting the Local Optima in Genetic Algorithm using Tabu Search. *Indian* 476
Journal of Science and Technology **12**, pp. 1–13, 2019. 477
45. Y.H. Dai, Convergence Properties of the BFGS Algorithm, *SIAM Journal on Optimization* **13**, pp. 693-701, 2002. 478
46. S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, Optimization by simulated annealing, *Science* **220**, pp. 671-680, 1983. 479
47. M.C. Robini, T. Rastello, I.E. Magnin, Simulated annealing, acceleration techniques, and image restoration, *IEEE Transactions on* 480
Image processing **8**, 10, pp. 1374-1387, 1999. 481
48. L. Zhang, H. Ma, W. Qian, H. Li, Protein structure optimization using improved simulated annealing algorithm on a three- 482
dimensional AB off-lattice model. *Computational Biology and Chemistry* **85**, 107237, 2020. 483

49. J.C.J. H. Aerts, G.B.M. Heuvelink, Using simulated annealing for resource allocation, *International Journal of Geographical Information Science* **16**, pp. 571-587, 2002. 485
50. A.T. Kalai, S. Vempala, Simulated annealing for convex optimization, *Mathematics of Operations Research* **31.2**, pp. 253-266, 2006. 486
51. L.M. Rasdi Rere, M.I. Fanany, A. M. Arymurthy, Simulated Annealing Algorithm for Deep Learning, *Procedia Computer Science* **72**, pp. 137-144, 2015. 487
52. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution. *Neurocomputing* **72**, pp. 269-277, 2008. 488
53. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, *Knowledge-Based Systems* **8**, pp. 333-344, 1995. 489
54. J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feedforward neural network, *Neurocomputing* **30**, pp. 219-232, 2000. 490
55. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, *Neural Networks* **146**, pp. 220-229, 2022. 491
56. F. H. F. Leung, H. K. Lam, S. H. Ling, P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, *IEEE Transactions on Neural Networks* **14**, pp. 79-88, 2003. 492
57. M. O'Neill, C. Ryan, Grammatical evolution, *IEEE Trans. Evol. Comput.* **5**, pp. 349-358, 2001. 493
58. H.G. Han, J.F. Qiao, A structure optimisation algorithm for feedforward neural network construction, *Neurocomputing* **99**, pp. 347-357, 2013. 494
59. K.J. Kim, S.B. Cho, Evolved neural networks based on cellular automata for sensory-motor controller, *Neurocomputing* **69**, pp. 2193-2207, 2006. 495
60. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository. 2023. Available online: <https://archive.ics.uci.edu> (accessed on 18 February 2024). 496
61. 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. 497
62. 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. 498
63. J.R. Quinlan, Simplifying Decision Trees. *International Journal of Man-Machine Studies* **27**, pp. 221-234, 1987. 499
64. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, *Machine Learning* **16**, pp. 59-88, 1994. 500
65. 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. 501
66. R. Setiono, W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, *Applied Intelligence* **12**, pp. 15-25, 2000. 502
67. G. Demiroz, H.A. Govenir, N. Ilter, Learning Differential Diagnosis of Erythematous-Squamous Diseases using Voting Feature Intervals, *Artificial Intelligence in Medicine*. **13**, pp. 147-165, 1998. 503
68. P. Horton, K. Nakai, A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, In: *Proceedings of International Conference on Intelligent Systems for Molecular Biology* **4**, pp. 109-15, 1996. 504
69. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, *Applied Intelligence* **7**, pp. 39-55, 1997. 505
70. 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. 506
71. 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. 507
72. J.G. Dy, C.E. Brodley, Feature Selection for Unsupervised Learning, *The Journal of Machine Learning Research* **5**, pp 845-889, 2004. 508
73. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, *Neural Processing Letters* **10**, pp 243-252, 1999. 509
74. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, *Intell. Data Anal.* **6**, pp. 483-502, 2002. 510
75. 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. 511
76. M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Trans Biomed Eng.* **56**, pp. 1015-1022, 2009. 512
77. 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. 513
78. 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. 514
79. N. Giannakeas, M.G. Tsipouras, A.T. Tzallas, K. Kyriakidi, Z.E. Tsianou, P. Manousou, A. Hall, E.C. Karvounis, V. Tsianos, E. Tsianos, A clustering based method for collagen proportional area extraction in liver biopsy images (2015) *Proceedings of the* 515

- Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100. 544
80. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, *JRSS-C (Applied Statistics)* **36**, pp. 260–276, 1987. 545
81. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, *Data & Knowledge Engineering* **44**, pp. 109–138, 2003. 546
82. P. Cortez, A. M. Gonçalves Silva, Using data mining to predict secondary school student performance, In *Proceedings of 5th Future Business Technology Conference (FUBUTEC 2008)* (pp. 5–12). EUROSIS-ETI, 2008. 547
83. I-Cheng Yeh, King-Jang Yang, Tao-Ming Ting, Knowledge discovery on RFM model using Bernoulli sequence, *Expert Systems with Applications* **36**, pp. 5866-5871, 2009. 548
84. 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. 549
85. 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. 550
86. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, *Optimization Methods and Software* **22**, pp. 225-236, 2007. 551
87. 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. 552
88. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, *The Journal of Machine Learning Research* **5**, pp. 549–573, 2004. 553
89. 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. 554
90. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989. 555
91. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, *Cement and Concrete Research*. **28**, pp. 1797-1808, 1998. 556
92. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, *J. Environ. Economics & Management* **5**, pp. 81-102, 1978. 557
93. 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. 558
94. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, *Evolutionary Computation* **10**, pp. 99-127, 2002. 559
95. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, *Mathematical Programming* **45**, pp. 547-566, 1989. 560
96. D. Yi, J. Ahn, S.Ji, An effective optimization method for machine learning based on ADAM, *Applied Sciences* **10**, 1073, 2020. 561
97. N. Xiao, X. Hu, X. Liu, K.C. Toh, Adam-family methods for nonsmooth optimization with convergence guarantees. *Journal of Machine Learning Research* **25**, pp. 1-53, 2024. 562
98. K.D. Tzimourta, I. Tsoulos, T. Bilero, A.T. Tzallas, M.G. Tsipouras, N. Giannakeas, Direct Assessment of Alcohol Consumption in Mental State Using Brain Computer Interfaces and Grammatical Evolution, *Inventions* **3**, 51, 2018 563
99. F. Glover, Parametric tabu-search for mixed integer programs. *Computers & Operations Research* **33**, pp. 2449-2494, 2006. 564
100. A. Lim, B. Rodrigues, X. Zhang, A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research* **174**, pp.1459-1478, 2006. 565
101. A. Bevilacqua, A methodological approach to parallel simulated annealing on an SMP system. *Journal of Parallel and Distributed Computing* **62**, pp. 1548-1570, 2002. 566