ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

NEUROCOMPUTING

Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/neucom

Neural network construction and training using grammatical evolution

Ioannis Tsoulos[a,*], Dimitris Gavrilis[b], Euripidis Glavas[c]

[a]Department of Computer Science, University of Ioannina, Ioannina, Greece
[b]Department of Electrical & Computer Engineering, University of Patras, Greece
[c]Department of Communications, Informatics and Management, Technological Educational Institute of Epirus, Greece

Received 6 March 2007; received in revised form 19 November 2007; accepted 16 January 2008
Communicated by A. Abraham

Abstract

The term neural network evolution usually refers to network topology evolution leaving the network's parameters to be trained using conventional algorithms. In this paper we present a new method for neural network evolution that evolves the network topology along with the network parameters. The proposed method uses grammatical evolution to encode both the network and the parameters space. This allows for a better description of the network using a formal grammar allowing the network architect to shape the resulting search space in order to meet each problem requirement. The proposed method is compared with other three methods for neural network training and is evaluated using 9 known classification problems and 9 known regression problems. In all 18 datasets, the proposed method outperforms its competitors.

Keywords: Grammatical evolution; Neural network; Context-free grammar; Genetic algorithm

1. Introduction

Evolutionary neural networks have been extensively used for various applications. In almost all cases only the topology of the network is evolved and the resulting network is trained using conventional algorithms. In other approaches of neural network evolution, only the weights of the network are evolved and the network topology is predefined. The use of a genetic algorithm to evolve only the topology is considered excessive since a genetic algorithm is inherently used for optimization problems with a vast search space. Even in the case of recurrent neural networks, the maximum number of available topologies per problem is limited. The proposed approach (Fig. 1) encodes both the topology of the neural network and its parameters (input vector, weights, bias) in a genetic algorithm using a context-free grammar (CFG). The combination of a CFG and a genetic algorithm is known as grammatical evolution and in the present case has the benefit of allowing easy shaping of the resulting search space.

Furthermore, recent developments in distributed and quantum computing will present a need in distributed algorithms. This will give an undisputed advantage to evolutionary algorithms over classical approaches (such as gradient-descent-based optimization) since they are natively parallel and can be used in distributed systems with very little modifications.

The proposed method is strongly evaluated on 18 known experimental datasets that are available on the Internet. The experimental datasets refer to both classification and regression problems. In order to reduce random variation of the datasets, 10-fold cross-validation has been applied to all experiments. Furthermore, in order to reduce random variation of the proposed algorithm itself, each experiment has been run 30 times and the mean is presented. The method is tested against neural networks that are trained with various algorithms:

- RPROP [27] which is an improved version of the well-known back-propagation (BP) method local optimization method.

*Corresponding author. Tel.: +30 2651 098835; fax: +30 2651 098890.
E-mail address: itsoulos@cs.uoi.gr (I. Tsoulos).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.01.017

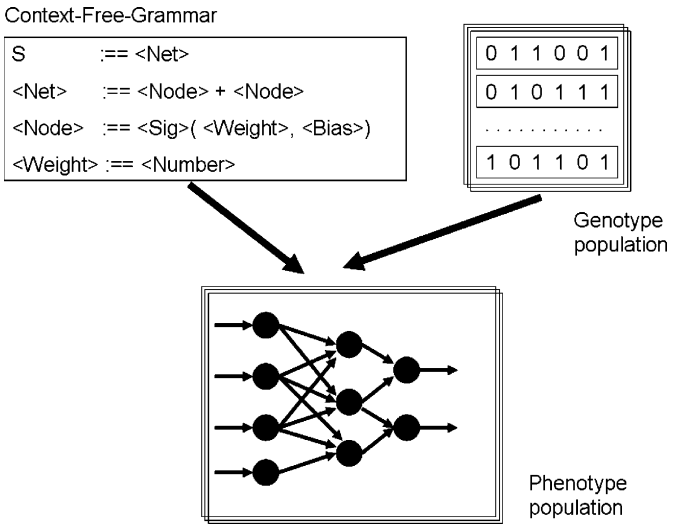


Fig. 1. The proposed method uses a context-free grammar in Backus Naur Form and the grammatical evolution procedure in order to construct and train a population of neural networks (phenotypes) from a genotype population.

- A Powell's variant of the well-known BFGS [25] local optimization method.
- A global optimization method namely MinFinder [35] which is capable under some conditions, to find all the local minimums of a function.
- Using a genetic algorithm to estimate the neural network's initial parameters and the BFGS [5,11] local optimization method for fine tuning using a local search. This method is similar to the one described in [28].
- In all cases, the function to be minimized in the case of neural networks is the train error. In almost all cases, the proposed method outperforms its competitors.

2. Related work

Artificial neural networks are well-established tools [8] used with success in many problems such as pattern recognition [3], solving differential equations [19], regression problems, classification problems, etc. The use of genetic algorithms in the evolution of neural networks has been widely used in literature. However, in most cases limited experimental results are presented. In evolving a neural network using a genetic algorithm, two main approaches are used. In the first approach, the network topology and the network weights are evolved simultaneously [2,16,36], while in the second approach only the network topology is evolved and the parameters are estimated using a standard approach such as a gradient-based optimization method [1,14,37]. Other approaches of defining the network's topology other than using genetic algorithms include pruning algorithms [26], simulated annealing-based algorithms [9] and particle swarm optimization techniques [38].

The representation of the neural network is another important aspect of the evolutionary approach. Three

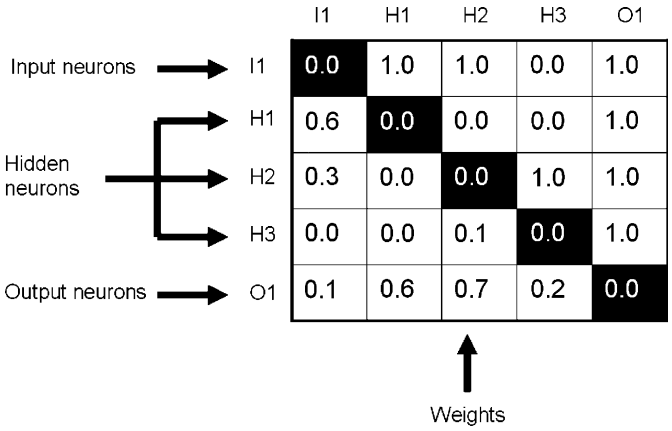


Fig. 2. A neural network representation using a table and evolving both topology and weights.

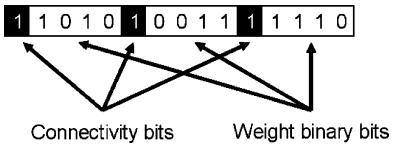


Fig. 3. A binary representation of a neural network topology and weights.

major approaches are adopted in the bibliography. In the first, a binary coding of the topology and weights is used (Fig. 2). In the second approach, a table structure is used to represent the connectivity between the neurons and the network's weights (Fig. 3). The third approach is considered the most sophisticated since it involves using a higher-level language such as grammar to describe the network topology (Fig. 1). The latter approach gives better control to the network architect since he can create node layers and better describe complex topologies.

Evolutionary neural networks have been widely used in solving various problems. The authors in [15] employ a descriptive language to model neural network topology and train the evolved networks using classical algorithms such as RPROP [27]. A modular architecture is used to limit the search space and improve performance. In [2] Ajith Abraham proposes meta-learning evolutionary neural networks and combines both the learning of the weights and topology into his algorithm. In his work, the author presents a large number of experimental results to support his findings. Evolutionary neural networks have also been proposed for a wide variety of problems. The authors in [17] use evolutionary neural networks to construct the controller of a mobile robot. An intrusion detection method is proposed in [14] where evolutionary neural networks are used to find the optimal network topology and weights. The authors extend the evolutionary algorithm to include a BP local search for each evolved network. In this way, the genetic algorithm is actually used to initialize the network's weights and the BP to perform a local search and train the network. A breast cancer diagnosis system is based on evolutionary networks in [1].

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017

The authors propose a hybrid algorithm that uses BP to estimate the network's weights after the network has been constructed and initialized by the genetic algorithm. In [28], the authors employ a genetic algorithm to estimate the weights and the number of neurons in the hidden layer of a radial basis function (RBF) network. The resulting RBF network is then used successfully to predict time series data. The authors in [36] evolve a neural network in order to create an adaptive intrusion detection system for computer network attacks. In [16], the DNA Microarray analysis is performed with high accuracy using evolved neural networks. In [18] the authors propose a Lisp-based genetic programming approach to evolve both the weights and the architecture of a neural network. Also, in [32] a genetic programming approach is used in order to evolve modular neural networks and in [6] a genetic programming technique based graph representation (PDGP) is used in order to evolve the topology and the weights of Neural Networks.

3. Grammatical evolution

The grammatical evolution approach has been introduced by Ryan and O'Neill [22] in 1998 and has been used successfully to solve various problems such as symbolic regression [23], discovery of trigonometric identities [29], robot control [7], caching algorithms [21] and financial prediction [4]. It has also been successfully used for feature selection and construction in [13,34] problem where the authors use their proposed algorithm as a wrapper for traditional neural network training algorithms in order to automatically construct new features for pattern recognition problems. The grammatical evolution approach uses a CFG and a genetic algorithm in order to map each phenotype to sentence produced by the grammar. The CFG $G=(N,T,S,P)$ where N is a set of non-terminal symbols, T is a finite set of terminal symbols where $N\cap T=\varnothing$. $S\in N$ is the starting symbol and P is a finite set of production rules in the form $A\rightarrow a$ or $A\rightarrow aB$, A, $B\in N$, $a\in T$.

The genetic algorithm is a stochastic optimization algorithm based on the theory of biological evolution [20]. It consists of a set of chromosomes (population) and evolves each population to the next generation through a set of genetic operators such as crossover and mutation. The probability of a chromosome to be included in the next generation, mostly (but not solely) depends on its fitness (or evaluation). The genetic algorithm can be described as follows:

1. initialize a random population of chromosomes,
2. evaluate each chromosome,
3. if a member of the population satisfies the termination criterion, terminate,
4. select a pool of chromosomes to be included in the next generation,
5. create the next generation from the pool by applying the crossover and mutation operators,
6. if the maximum number of generations has been reached, terminate.

A chromosome C is usually represented as binary and it consists of a set of l genes c_i

$$x=(c_1,c_2,\ldots,c_l),\quad c_l\in\{0,1\}$$

l is also the length of the chromosome. In the proposed implementation, in order to increase speed, the chromosomes are represented as sets of integers where each gene is defined as: $c_l\in\{1,2,\ldots,255\}$. The upper limit 255 practically means that set number of production rules for each non-terminal symbol cannot exceed 255. However, this can easily change.

The genetic operators used are crossover and mutation. For the crossover operator, assume that $C_1=(c_1^1,c_2^1,\ldots,c_N^1)$ and $C_2=(c_1^1,c_2^2,\ldots,c_N^2)$ are two chromosomes. In the proposed algorithm, a simple crossover is performed where the resulting new chromosomes NC_1, NC_2 are defined as

$$NC_1=(c_1^1,c_2^1,\ldots,c_r^1,c_{r+1}^2,\ldots,c_N^1),$$
$$NC_2=(c_1^2,c_2^2,\ldots,c_r^2,c_{r+1}^1,\ldots,c_N^1)$$

where $r\in\{1,2,\ldots,N-1\}$ is a randomly chosen position inside the chromosome. The crossover probability in the present implementation is set to 0.95.

In the mutation procedure, every element of each chromosome can be changed to a different random integer in the range $[0,255]$ with probability equal to mutation rate, which is a number in the range $[0,1]$.

The selection of the intermediate pool of chromosomes in the current implementation is performed using the tournament selection process in which N random chromosomes are selected from the population and according to their fitness, 2 are selected. The genetic operators are applied to the 2 selected chromosomes in order to create the 2 new chromosomes that are to be included in the next generation. This approach, compared to other techniques (like the roulette selection), usually avoids converging towards local minima thus giving better results.

An example of grammatical evolution can be seen in Table 1. The grammar used in the example can be seen in Fig. 4.

4. Neural network evolution and training using grammatical evolution

In order to construct the neural network using grammatical evolution, the network itself has to be expressed using the BNF grammar. A two-layer network (see Fig. 5) can be expressed as

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

where $x\in R^d$, $H=nodes/(d+2)$ denotes the number of neurons in the hidden layer and w denotes the weights of

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017

Table 1
Example of the grammatical evolution procedure

String	Chromosome	Operation
⟨*expr*⟩	9, 8, 6, 4, 16, 10, 17, 23, 8, 14	9 mod 3 = 0
(⟨*expr*⟩⟨*op*⟩⟨*expr*⟩)	8, 6, 4, 16, 10, 17, 23, 8, 14	8 mod 3 = 2
(⟨*terminal*⟩⟨*op*⟩⟨*expr*⟩)	6, 4, 16, 10, 17, 23, 8, 14	6 mod 2 = 0
(⟨*xlist*⟩⟨*op*⟩⟨*expr*⟩)	4, 16, 10, 17, 23, 8, 14	4 mod 3 = 1
(*x*2⟨*op*⟩⟨*expr*⟩)	16, 10, 17, 23, 8, 14	16 mod 4 = 0
(*x*2+⟨*expr*⟩)	10, 17, 23, 8, 14	10 mod 3 = 1
(*x*2+⟨*func*⟩(⟨*expr*⟩))	17, 23, 8, 14	17 mod 4 = 1
(*x*2+cos(⟨*expr*⟩))	23, 8, 14	23 mod 3 = 2
(*x*2+cos(⟨*terminal*⟩))	8, 14	8 mod 2 = 0
(*x*2+cos(⟨*xlist*⟩))	14	14 mod 3 = 2
(*x*2+cos(*x*3))		

```
Rule                                                Rule number
S::=<expr>                                          (0)

<expr> ::=  ( <expr> <op> <expr> )                  (0)
          | <func> ( <expr> )                       (1)
          |<terminal>                               (2)

<op>   ::=    + | - | * | /                         (0-3)

<func> ::=  sin | cos | exp | log                   (0-3)

<terminal>::=<xlist>                                (0)
          |<digitlist>.<digitlist>                  (1)

<xlist> ::= x1 | x2 | … | xN                        (0-N-1)

<digitlist>::= <digit>                              (0)
          | <digit><digitlist>                      (1)

<digit>  ::= 0 | 1 | 2 | … | 9                      (0-9)
```

Fig. 4. The BNF grammar used for the grammatical evolution example of Table 1.

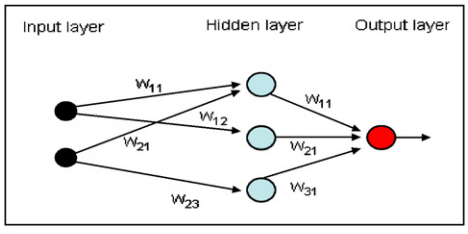


Fig. 5. A graphical representation of a two-layer neural network.

the network. The activation functions of each neuron is the sigmoid function: $\sigma(x) = 1/(1+e^{-x})$.

A CFG in Backus Naur Form that can create such a network can be seen in Fig. 6. The ⟨*sigexpr*⟩ non-terminal denotes the resulting neural network and each neuron is represented by ⟨*Node*⟩. The sigmoid function is denoted by *sig*(·) and the inputs of the network are the ⟨*xxlist*⟩ non-terminal. It can also be seen from Fig. 6 that the parameters of the network (weights, inputs and bias) are also encoded in the grammar and are in the form of ⟨*number*⟩.

```
Rule                                                Rule Number
-------------------------------------------------------------
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)
             | ...                                  (..)
             | Xn                                   (n-1)

<number> ::=  (<digitlist>.<digitlist>)             (0)
            |  (-<digitlist>.<digitlist>)           (1)

<digitlist> ::=  <digit>                            (0)
             |   <digit><digitlist>                 (1)

<digit>  ::= 0                                      (0)
           | 1                                      (1)
           | 2                                      (2)
           .....                                    ...
           | 9                                      (9)
```

Fig. 6. The proposed BNF grammar that can construct a two-layer network.

Each neural network $N_g(x)$ is constructed from a chromosome through the grammar. The fitness of the chromosome is the performance of the neural network on a selected train dataset. More specifically, for each point (x_i, y_i), $i = 1,\ldots,M$ where M is the total number of points in the train set of the train set, the fitness f is calculated as $f = \sum_{i=1}^{M}(N_g(x_i) - y_i)^2$ where y_i is the desired output for the input vector x_i.

The proposed coding schema for representing neural networks overcomes many of the problems that are indicated in [12]. More specifically, it does not allow the genetic algorithm to restrict the network's architecture and it is more accurate than most binary representation schemas while maintaining all the benefits of using a higher description for the network's architecture as mentioned in [15].

5. Dataset description

The datasets used for evaluating the proposed method, are known datasets that are available for download from the Internet and refer to both classification and regression problems. The number of datasets that are presented and evaluated are 9 classification problems and 9 regression problems.

5.1. Classification datasets

- *Wine*. The wine recognition dataset (WINE) contains data from wine chemical analysis. It contains 178 examples of 13 attributes each that are classified into three classes.

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017

- *Glass*. This dataset (GLASS) contains glass component analysis for glass pieces that belong to 6 classes. The dataset contains 214 examples with 10 attributes each.
- *Pima Indians Diabetes*. The PIMA dataset contains 768 examples of 8 attributes each that are classified into two categories: healthy and diabetic.
- *Wisconsin Diagnostic Breast Cancer*. The Wisconsin Diagnostic Breast Cancer dataset (WDBC) contains data for breast tumors. It contains 569 training examples of 30 attributes each that are classified into two categories.
- *Circular Artificial data*. The circular artificial dataset (CIRCULAR) contains 1000 examples that belong to two categories. The data in the first class belong to a circle and the data of the second class belong to a circular disc outside the first circle. Each example vector has two attributes. It is expanded by adding 3 more attributes generated randomly (noise) using a normal distribution.
- *Spiral Artificial data*. The spiral artificial dataset (SPIRAL) contains 1000 examples that belong to two classes (500 examples each). The data in the first class are created using the following formula: $x = 0.5\,t\cos(0.08\,\mathrm{t})$, $y = 0.5\,t\cos(0.08 + \pi/2)$, and the second-class data using: $x = 0.5\,t\cos(0.08\,t + \pi)$, $y = 0.5\,t\cos(0.08\,t + 3\pi/2)$. The original features vector has two attributes (x, y).
- *Spiral Artificial data* 2. The second spiral dataset (SPIRAL2) is created as the first dataset. Its difference is that its primitive set is expanded by adding 3 more noisy attributes using normal distribution.
- *Liverdisorder*. This dataset contains blood analysis data from people with liver disorders. It consists of 345 examples of 6 attributes each.
- *Ionosphere dataset*: The ionosphere dataset contains data from the Johns Hopkins Ionosphere database. It contains 351 examples of 34 attributes each that are split into two classes.

5.2. Regression datasets

- *BL*. This dataset can be downloaded from StatLib (http://stat.cmu.edu/datasets/). It contains data from an experiment on the affects of machine adjustments on the percentage time to count bolts.
- *FA*. The FA dataset contains percentage of body fat, age, weight, height and 10 body circumference measurements. The goal is to fit body fat to the other measurements.
- *LW*. This dataset is produced from a study that was to identify risk factors associated with giving birth to a low-birth-weight baby (weighing less than 2500 g). Data were collected on 189 women, 59 of which had low-birth-weight babies and 130 of which had normal-birth-weight babies.
- *NT*. This dataset contains data from a paper in the Journal of the American Medical Association that examined whether the true mean body temperature is 98.6°F.
- *PO*. This dataset is available from StatLib (http://stat.cmu.edu/datasets/). It contains pollution data.
- *PW*. This dataset contains numeric prediction data using instance-based learning with encoding length selection.
- *SN*. This dataset contains data on a viticultural experiment that was conducted to investigate different methods of trellising and pruning.
- *MB*. This dataset is available from Smoothing Methods in Statistics (http://stat.cmu.edu/datasets/).
- *BK*. This dataset comes from Smoothing Methods in Statistics (http://stat.cmu.edu/datasets/).

6. Experimental results

In this section, the results from the application of the proposed method against the methods RPROP, BFGS and MinFinder are listed. Each method was tested for different topologies of the resulting neural network (e.g. the number of hidden neurons) and the topology with the best results was selected. The genetic algorithm's parameters have been estimated experimentally after observing that the algorithm converges faster using these parameters. The parameters can be seen in Table 2.

For comparison, the datasets were split into 10 parts and 10-fold cross-validation was used in order to reduce random variation of the data. The experimental results for the 9 classification problems can be seen in Figs. 7 and 8. Figs. 7 and 8 show the test error for each of the 4 compared methods (smaller is better).

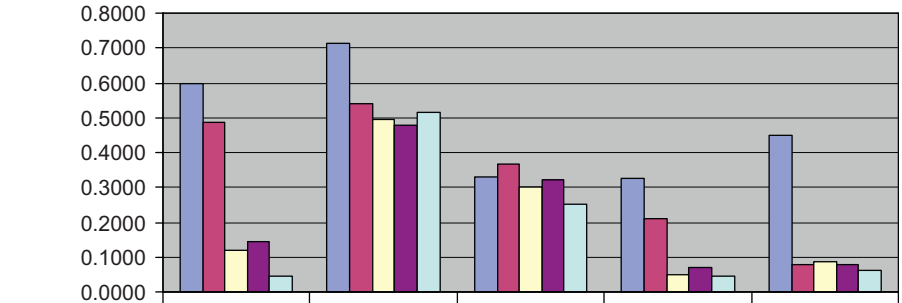
From the experimental results, it can be seen that in all cases except one (SPIRAL2), the NNC performed better, the NNC method is followed by MinFinder and the RPROP and BFGS methods give the worst results. It must also be noted that the MinFinder method is a global optimization method that requires tremendous amounts of time (several orders of magnitude than the proposed one).

The experimental results for the 9 regression problems can be seen in Figs. 9 and 10. The mean square error (MSE) is measured for the RPROP, BFGS, MinFinder, Genetic and the proposed NNC method. As in the classification problems, the NNC method gives the best results.

Table 2
Genetic algorithm parameters used in the experiments

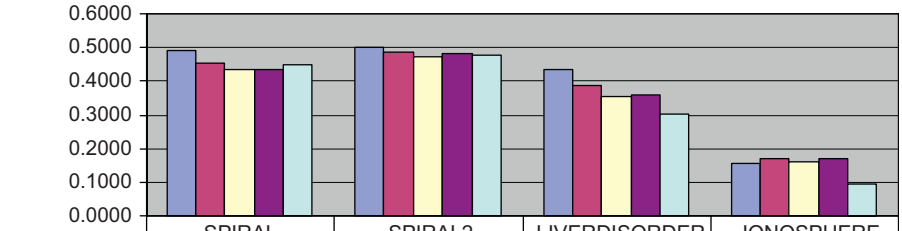
Parameter	Value
Maximum number of generations	500
Population size	500
Chromosome length	100
Crossover rate	0.95
Mutation rate	0.05
Tournament size	10

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017



	WINE	GLASS	PIMA	WDBC	CIRCULAR
RPROP	0.5993	0.7137	0.3319	0.3271	0.4494
BFGS	0.4886	0.5393	0.3656	0.2091	0.0795
MINFINDER	0.1178	0.4941	0.3004	0.0489	0.0857
GENETIC	0.1426	0.4801	0.3216	0.0687	0.0784
NNC	0.0444	0.5140	0.2500	0.0456	0.0620

Fig. 7. Test error (%) for the first 5 classification problems. The RPROP, BFGS and MinFinder methods are compared against the proposed NNC method.



	SPIRAL	SPIRAL2	LIVERDISORDER	IONOSPHERE
RPROP	0.4906	0.5022	0.4356	0.1578
BFGS	0.4531	0.4847	0.3886	0.1708
MINFINDER	0.4343	0.4704	0.3559	0.1627
GENETIC	0.4358	0.4815	0.3584	0.1699
NNC	0.4490	0.4780	0.3006	0.0966

Fig. 8. Test error (%) for the last 4 classification problems. The RPROP, BFGS and MinFinder methods are compared against the proposed NNC method.

In order to better evaluate the proposed method, additional comparisons were made. More specifically, the proposed method was compared to a standard generic algorithm that encodes the topology and weights into a binary chromosome (see Fig. 3). This approach is most widely used in bibliography. The experiments were carried out on all the classification datasets and in order to reduce random variance they ran 30 times and the MSE is presented in Fig. 11. As it can be seen from Fig. 11, the NNC method outperforms the classical approach in 6 out of 8 datasets.

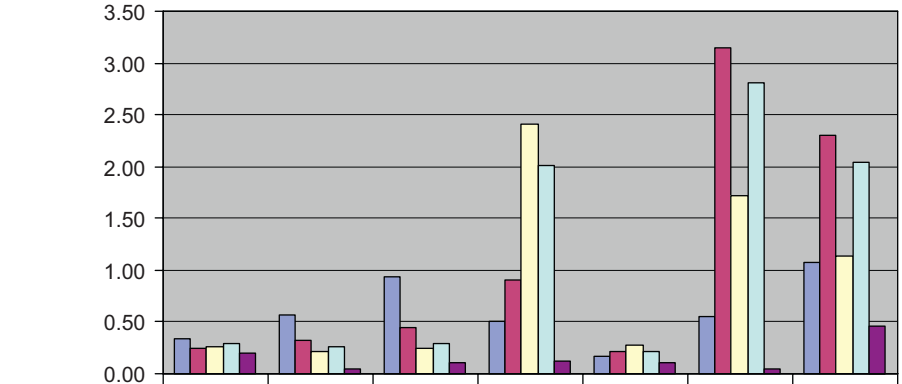
7. Scalability

One of the advantages of genetic programming is that it is inherently scalable. Much work has been done on parallel genetic algorithms [10,24,30,31,33] and all methods can be utilized by the proposed algorithm. A more practical example of a scalable version of the proposed method is the use of the island approach. In this approach, multiple generations are evolved simultaneously (islands) and periodically (or after a predefined number of generations) the best chromosomes of each island are gathered compiled into a new population. The new population is then re-distributed to all islands. The process is repeated until the termination criteria are satisfied. This approach which can be seen in Fig. 12, can be easily implemented using the well-known Message Passing Interface, a well-known technique for parallel programming.

Another approach is to create a scalable version of the proposed method by dividing the search space into smaller subspaces and letting each parallel processing unit to explore each subspace separately. In the present approach, the creation of each subspace can be accomplished by manipulating the grammar. In the present case, each subspace could be mapped onto a subtopology, thus allowing for different topologies to be explored simultaneously.

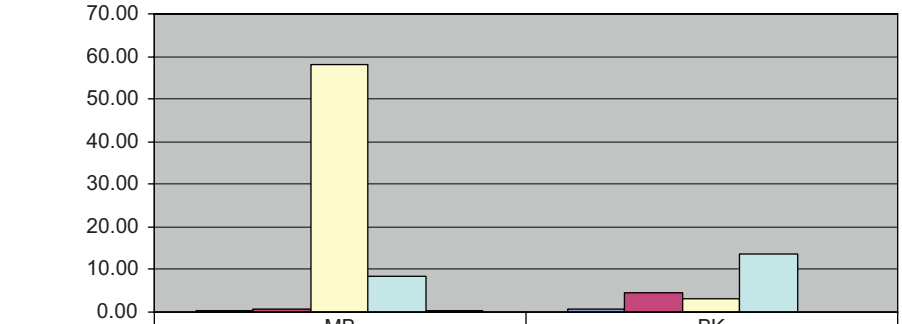
The algorithm's efficiency can also be improved by scaling the fitness function. Since on real-world problems

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017



	PO	BL	FA	LW	PW	NT	SN
RPROP	0.34	0.57	0.94	0.51	0.17	0.56	1.08
BFGS	0.25	0.33	0.44	0.91	0.21	3.14	2.31
MINFINDER	0.26	0.21	0.25	2.41	0.27	1.72	1.13
GENETIC	0.29	0.26	0.29	2.01	0.22	2.81	2.04
NNC	0.2	0.05	0.1	0.13	0.1	0.04	0.46

Fig. 9. MSE error for the first 7 regression problems. The RPROP, BFGS and MinFinder methods are compared against the proposed NNC method.



	MB	BK
RPROP	0.49	0.54
BFGS	0.68	4.44
MINFINDER	57.94	3.06
GENETIC	8.45	13.64
NNC	0.29	0.09

Fig. 10. MSE error for the last 2 regression problems. The RPROP, BFGS and MinFinder methods are compared against the proposed NNC method.

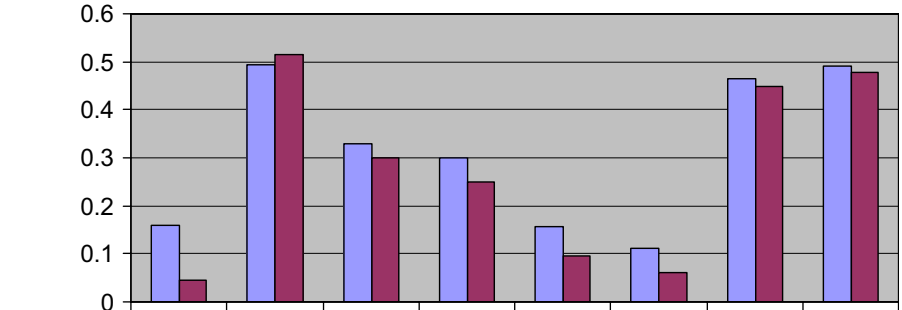
the construction of a neural network, its training and testing is a relatively computationally expensive process, the evaluation of each chromosome could be performed in a distributed manner thus reducing the time needed to evolve a single generation.

8. Conclusions and future work

The method proposed in this paper (NNC) uses grammatical evolution in order to construct and train a neural network. The proposed method encodes in the grammar not only the network topology but also the network parameters. The NNC method is evaluated on 9 known classification and 9 known regression problems and compared against the state of the art methods: RPROP, BFGS and MinFinder. An accurate comparison of the four methods is presented that uses 10-fold cross-validation and 30-fold experiment replication. The experimental results show that the proposed NNC method outperforms the other methods. Two major advantages of NNC are that it is significantly faster than MinFinder by several orders of magnitude and is natively parallel. This gives NNC an edge over traditional methods since it can take advantage of distributed computing technologies.

In continuing this work, the proposed algorithm could be extended to include feature selection and/or feature construction (using linear and/or non-linear transformation of the input space). Another aspect of the proposed network that still needs to be explored is the construction of recurrent networks and its evaluation on time series

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017



	WINE	GLASS	LIVER DISOR	PIMA	IONOS PHERE	CIRCU LAR	SPIRAL	SPIRAL 2
Genetic	0.1593	0.495	0.3297	0.2991	0.1563	0.1127	0.4638	0.4921
NNC	0.0444	0.5140	0.3006	0.2500	0.0966	0.0620	0.4490	0.4780

Fig. 11. Comparison of the proposed (NNC) method to the classical (Genetic) approach. The (%) test error is presented.

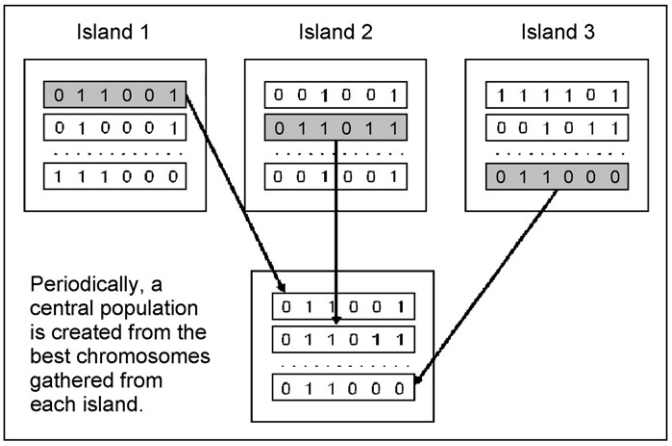


Fig. 12. A classical approach of a parallel genetic algorithm. Multiple generations are evolved simultaneously (in each island) and periodically, a new population is created centrally by gathering the best chromosomes from each island.

data. Finally, a software package will be made available from the authors Internet homepages using which in future versions, could include a distributed version that would run on grids.

References

[1] H.A. Abbass, An evolutionary artificial neural networks approach for breast cancer diagnosis, Artif. Intell. Med. 25 (3) (July 2002) 265–281.
[2] A. Abraham, Meta learning evolutionary artificial neural networks, Neurocomputing 56 (2004) 1–38.
[3] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[4] A. Brabazon, M. O'Neill, in: H.R. Arabnia (Ed.), Proceedings of the International Conference on Artificial Intelligence, vol. II, CSREA Press, 2003, pp. 492–498.
[5] C.G. Broyden, The convergence of a class of double-rank minimization algorithms: Part I and II, J. Inst. Math. Appl. 6 (1970) 76–90.
[6] J. Carlos, F. Pujol, R. Poli, Evolution of the topology and the weights of neural networks using genetic programming with a dual representation, Appl. Intell. 8 (1998) 73–84.
[7] J.J. Collins, C. Ryan, in: Proceedings of AROB 2000, Fifth International Symposium on Artificial Life and Robotics, 2000.
[8] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 303–314.
[9] M.C.P. de Souto, A. Yamazaki, T.B. Ludernir, Optimization of neural network weights and architecture for odor recognition using simulated annealing, in: Proceedings of the 2002 International Joint Conference on Neural Networks, vol. 1, 2002, pp. 547–552.
[10] Dieferson L.A. Araujo, Heitor S. Lopes, Alex A. Freitas, A parallel genetic algorithm for rule discovery in large databases, in: Proceedings of the IEEE Systems, Man and Cybernetics Conference, vol. III, Tokyo, Japan, 12–15 October 1999, pp. 940–945.
[11] R.E. Fletcher, A new approach to variable metric algorithms, Comput. J. 13 (1970) 317–322.
[12] W. Gao, Study on new evolutionary neural network, Int. Conf. Mach. Learn. Cybernet. 2 (Nov 2003) 1287–1292.
[13] D. Gavrilis, I. Tsoulos, G. Georgoulas, E. Glavas, Classification of fetal heart rate using grammatical evolution, in: IEEE 2005 Workshop on Signal Processing Systems.
[14] S.-J. Han, S.-B. Cho, Evolutionary neural networks for anomaly detection based on the behavior of a program, IEEE Trans. Systems, Man Cybernet. 36 (3) (2006).
[15] J.-Y. Jung, J.A. Reggia, Evolutionary design of neural network architectures using a descriptive encoding language, IEEE Trans. Evol. Comput. 10 (6) (2006).
[16] K.-J. Kim, S.-B. Cho, Evolving artificial neural networks for DNA microarray analysis, in: The 2003 Congress on Evolutionary Computation (CEC '03), vol. 4, December 2003, pp. 2370–2377.
[17] K.-J. Kim, S.-B. Cho, Evolved neural networks based on cellular automata for sensory-motor controller, Neurocomputing 69 (2006) 2193–2207.
[18] J.R. Koza, J.P. Rice, Genetic generation of both the weights and architecture for a neural network, in: International Joint Conference on Neural Networks, IJCNN-91, II: pp. 397–404, 8–12, 1991.
[19] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Networks 9 (1998) 987–1000.
[20] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer, Berlin.
[21] M. O'Neill, C. Ryan, in: K. Miettinen, M.M. Mkel, P. Neittaanmki, J. Periaux (Eds.), Evolutionary Algorithms in Engineering and Computer Science, Jyvskyl, Finland, 1999, pp. 127–134.
[22] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. 5 (2001) 349–358.
[23] M. O'Neill, C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in a Arbitary Language, Genetic Programming, vol. 4, Kluwer Academic Publishers, Dordrecht, 2003.

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017

[24] R. Poli, Discovery of symbolic, neuro-symbolic and neural networks with parallel distributed genetic programming. In: G.D. Smith, N.C. Steele, R.F. Albrecht (Eds.), Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference, ICANN-GA97, University of East Anglia, Norwich, UK, 1997. Springer, Berlin, 1998.
[25] M.J.D. Powell, A tolerant algorithm for linearly constrained optimization calculations, Mathematical Programming 45 (1989).
[26] R. Reed, Pruning algorithms—a survey, IEEE Trans. Neural Networks 4 (1993) 740–747.
[27] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the {RPROP} algorithm, in: Proceedings of the {IEEE} International Conference on Neural Networks, 1993, pp. 586–591.
[28] V.M. Rivas, J.J. Merelo, P.A. Castillo, M.G. Arenas, J.G. Castellano, Evolving RBF neural networks for time-series forecasting with EvRBF, Inform. Sci. 165 (3-4) (19 Oct 2004) 207–220.
[29] C. Ryan, M. O'Neill, J.J. Collins, in: Proceedings of Mendel 1998: Fourth International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, Brno, Czech Republic, Technical University of Brno, Faculty of Mechanical Engineering, 1998, pp. 111–119.
[30] Sin Man Cheang, Kin Hong Lee, Kwong Sak Leung, Data classification using genetic parallel programming, in: E. Cant'u-Paz, J.A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M.A. Potter, A.C. Schultz, K. Dowsland, N. Jonoska, J. Miller (Eds.), Genetic and Evolutionary Computation—GECCO-2003, vol. 2724 of Lecture Notes in Computer Science, Chicago, 12–16 July 2003, Springer, Berlin, pp. 1918–1919.
[31] Sin Man Cheang, Kin Hong Lee, Kwong Sak Leung, Evolving data classification programs using genetic parallel programming, in: Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, Tom Gedeon (Eds.), Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, 8–12 December 2003, IEEE Press, New York, pp.248–255.
[32] C. Sung-Bae, K. Shimohara, Modular neural networks evolved by genetic programming, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 20–22 May 1996, pp. 681–684.
[33] Sven E. Eklund, Time series forecasting using massively parallel genetic programming, in: Proceedings of Parallel and Distributed Processing International Symposium, 22–26 April 2003, pp. 143–147.
[34] I. Tsoulos, G. Georgoulas, D. Gavrilis, C. Stylios, Introducing Grammatical Evolution for FHR analysis and classification, in: Third IEEE Conference on Intelligent Systems.
[35] I.G. Tsoulos, I.E. Lagaris, MinFinder: locating all the local minima of a function, Comput. Phys. Commun. 174 (2) (January 2006) 166–179.
[36] L. Wang, G. Yu, G. Wang, D. Wang, Method of evolutionary neural network-based intrusion detection, in: International Conferences on Info-tech and Info-net 2001 (ICII 2001), vol. 5, Oct.–Nov. 2001, pp. 13–18.
[37] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.
[38] C. Zhang, H. Shao, Y. Li, Particle swarm optimization for evolving artificial neural networks, IEEE Int. Conf. Systems, Man Cybergenet. (2000) 2487–2490.

Ioannis Tsoulos received his Ph.D. degree from the Department of Computer Science of the University of Ioannina in 2006. He is currently a Visiting Lecturer at the same department. His research interest areas include optimization, genetic programming and neural networks.

Dimitris Gavrilis is currently a researcher at the Digital Curation Unit, Athena Research Centre. He holds a PhD and a Diploma from the Electrical & Computer Engineering Department, University of Patras. His research interests include: feature selection & extraction for classification & regression problems, evolutionary algorithms and neural networks, intrusion detection and prevention algorithms for denial of service and web-based attacks.

Euripidis Glavas received his Ph.D. degree from the University of Sussex in 1988. He is currently an associate professor at the Department of Communications, Informatics and Management of the Technological Educational Institute of Epirus. His research areas include Optical Wave Guide, Computer Architecture and Neural Networks.

Please cite this article as: I. Tsoulos, et al., Neural network construction and training using grammatical evolution, Neurocomputing (2008), doi:10.1016/j.neucom.2008.01.017