Confidence Interval Based Crossover Applied to Neural Network Evolution

N. García-Pedrajas, D. Ortiz-Boyer, and C. Hervás-Martínez

Abstract— **In this work we present an application of the *Confidence Interval Based Crossover using L_2 Norm* (CIXL2) to the evolution of neural networks. This crossover operator is based on obtaining the statistical features of the best individuals of the population, these features are used as virtual individuals for the crossover operator parents.**

The codification of the neural networks is made using a non-redundant genetic encoding that avoids the permutation problem. The evolved neural networks are applied to three problems of classification from real-world data sets. We compare our results with a neural network of the same complexity trained by means of a backpropagation algorithm and with a genetic algorithm with the same experimental setup that uses the BLX-α crossover operator.

Palabras clave— **Neural network evolution, genetic algorithms, non-redundant genetic coding, confidence interval based crossover.**

I. Introduction

In the area of neural networks[1] design one of the main problems is finding suitable architectures for solving specific problems. The election of such architecture is very important, as a network smaller than needed would be unable to learn and a network larger than needed would end in over-training.

The problem of finding a suitable architecture and the corresponding weights of the network is a very complex task (for a very interesting review of the matter the reader can consult [2]).

Evolutionary computation[3][4] is a set of global optimization techniques that have been widely used in late years for training and automatically designing neural networks. There have been many applications for parametric learning[5] and for both parametric and structural learning[6] [7] [8] [9] [10] [11] [12] [13] [14] [15].

These works fall in two broad categories of evolutionary computation: genetic algorithms and evolutionary programming. The use of only structural learning in evolutionary computation yields some noise in the fitness function because the evaluation of the fitness can not be done in a neural network without considering the weights, so the network must be trained before evaluating its fitness. From the same representation this training process produces different networks, causing a one-to-many mapping from genotype to phenotype. This one-to-many mapping is the source of the noise in the evaluation of the fitness of a network.

G. F. Miller *et al.*[16] proposed that evolutionary computation is a very good candidate to be used to search the space of topologies because the fitness function associated with that space is complex, noisy, non-differentiable, multi-modal and deceptive. G. F. Miller *et al.* identified two approaches: the *strong specification scheme*, where each connection of the network is specified by its numerical representation, and the *weak specification scheme*, where the exact structure of the network is not explicitly codified but is computed based on the information contained in the chromosome. The former has been more widely used, as the latter introduces more noise in the fitness function and is more deceptive, due to the fact that there is a one-to-many mapping between the space of representation and the solution space. P. Angeline *et al.*[7] have suggested that in this situation neither the representation nor the use of crossover operator is recommendable. Odri *et al.*[8] developed an evolutionary algorithm to change the anatomy of a multilayered neural network. The algorithm only changes the structure, the weights are adjusted by means of a backpropagation algorithm. The neurons that contribute with a nonzero error value to the overall error of the network are subject to cell division; also the connections could be degenerated along evolution. When all the connections of a neuron are degenerated the neuron is removed. The network was only tested in some simple problems (e.g.: the XOR problem). Its major drawback is that it tends to generate networks larger than necessary that usually end over-fitting the data.

G. Bebis *et al.*[13] proposed a model that combined pruning and genetic algorithm and encouraged the evolution of small networks with good generalization. The model was tested using both

Universidad de Córdoba, Dpto. de Informática y Análisis Numérico, E-mail: {npedrajas, dortiz, chervas}@uco.es.

This work was supported in part by the Projects TIC2001-2577 and TIC2002-04036-C05-02 of the Spanish Comisión Interministerial de Ciencia y Tecnología.

artificial and real databases with good performance. Nevertheless, the algorithm is quite complex and some steps of the evolution involve too many *ad hoc* procedures. Also, there is a lot of parameters to be fixed for the execution of the algorithm, and the model is very sensitive to some of these parameters.

X. Yao *et al.*[6] developed an evolutionary programming model, called EPNet, that evolved generalized multilayered perceptrons[17] using the direct encoding scheme. The algorithm is quite complex with several mutations that involved the structure and the weights. It also included hybrid training by means of a backpropagation learning rule. The model is tested against many real problems with a very interesting performance.

II. Confidence interval based crossover

In this work we present the application of a new crossover based on statistical features of the best individuals of the population. This crossover has been successfully applied to function optimization with[18] and without constraints[19]. The crossover is based on obtaining the information of the confidence intervals of the genes using the L_2 norm of the best individuals, so it is called *Confidence Interval Based Crossover using L_2 Norm* (CIXL2)[20]. The crossover is described in the rest of this section.

In this study we will work with the i-th gene without loss of generality. Let β be the set of N individuals that forms the population and $\beta^* \subset \beta$ the subset of the n fittest individuals, and q the number of genes on each chromosome. Let us assume that the genes, β_i, of the chromosomes of the individuals in β^* are independent random variables, then we can consider β_i a random variable with a continuous distribution $H(\beta_i)$. With a localization parameter of the form μ_{β_i}, we have the model $\beta_i = \mu_{\beta_i} + e_i$, for each $i = 1, \ldots, q$, being e_1 a random variable.

If we assume that the n fittest individuals form actually a simple random sample $(\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_n})$ of the distribution of the fittest individuals of the population β_i^b, the model can be written:

$$\beta_{ij}^b = \mu_b + e_{ij}, \qquad \text{for j= 1, 2, \ldots, n.} \qquad (1)$$

Let us consider the L_2 norm, defined as $\|\beta_i^b\|_2 = \sqrt{\sum_{j=1}^n (\beta_{ij}^b)^2}$, then we can define the dispersion function, D_2, induced by the L_2 norm as: $D_2(\mu_b) = \sqrt{\sum_{j=1}^n (\beta_{ij}^b - \mu_b)^2}$. The estimator using this dispersion function of the localization parameter is the mean of the distribution β_i^b, that is, $\hat{\mu_b} = \bar{\beta_i^b}$.

The sample mean estimator is a good linear estimator, so it has the properties of unbiasedness and consistency, and when the distribution of the genes is normal, it follows a normal distribution $N(\mu_b, \sigma_b^2/n)$. Under these assumptions we have a bilateral confidence interval for the localization of the genes using the sample mean as localization parameter. This confidence interval, for a confidence coefficient $1-\alpha$, has the form:

$$I_{1-\alpha}(\mu_b) = \left[\bar{\beta_i^b} - t_{n-1,\alpha/2}\frac{S_b}{\sqrt{n}}; \bar{\beta_i^b} + t_{n-1,\alpha/2}\frac{S_b}{\sqrt{n}}\right], \qquad (2)$$

where $S_b = \left[\sum_{j=1}^n (\beta_{ij}^b - \hat{\beta}_i^b)^2/(n-1)\right]^{1/2}$ is the standard deviation, and t_{n-1} is a Student t of $n-1$ degrees of freedom.

A. Crossover operator method

From the confidence interval of equation 2 we build three individuals that are considered the parents of the proposed crossover. These three parents are formed by: all the lower limit values of the confidence intervals of the genes, individual CILL; all the upper limit values of the confidence intervals of the genes, individual CIUL; and all the means of the confidence intervals of the genes, individual CIM. These individuals divide the domain of the gene values into three subintervals: $I_i^L \equiv [a_i, CILL_i)$, $I_i^M \equiv [CILL_i, CIUL_i]$, and $I_i^R \equiv (CIUL_i, b_i]$.

The interval based crossover operator using L_2 norm, CIXL2, creates an offspring β^s, from an individual of the population, $\beta^f = (\beta_1^f, \beta_2^f, \ldots, \beta_p^f)$, and the three individuals CILL, CIUL, and CIM obtained from the confidence interval. We consider these four individuals and their fitness (being $f(\beta)$ the fitness value of individual β) and distinguish three cases depending on the position of β^f in one of the three subintervals defined above. These three cases are:

Case 1: $\beta_i^f \in I_i^L$. If $f(\beta^f) > f(CILL)$ then $\beta_i^s = r(\beta_i^f - CILL_i) + \beta_i^f$ else $\beta_i^s = r(CILL_i - \beta_i^f) + CILL_i$.

Case 2: $\beta_i^f \in I_i^M$. If $f(\beta^f) > f(CIM)$ then $\beta_i^s = r(\beta_i^f - CIM_i) + \beta_i^f$ else $\beta_i^s = r(CIM_i - \beta_i^f) + CIM_i$.

Case 3: $\beta_i^f \in I_i^R$. If $f(\beta^f) > f(CIUL)$ then $\beta_i^s = r(\beta_i^f - CIUL_i) + \beta_i^f$ else $\beta_i^s = r(CIUL_i - \beta_i^f) + CIUL_i$.

where r is a uniform random number belonging to [0, 1].

The effect of the crossover over a population is exemplified on Figure 1. We can see how the populations is driven towards the distribution of the best individuals after the repeated application of the crossover.

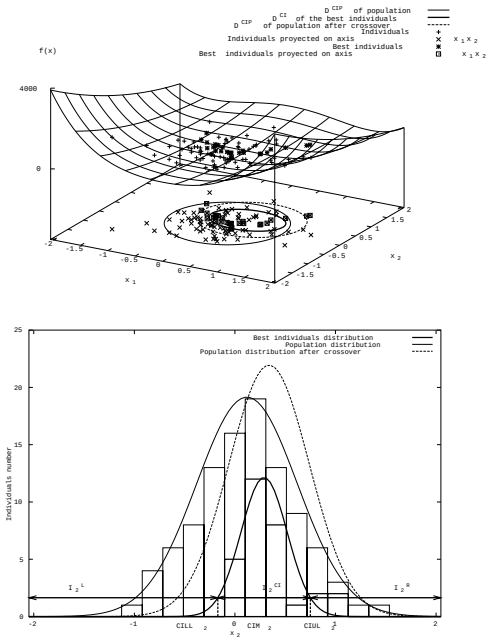

Fig. 1. Effect of the CIXL2 crossover over a population .

III. Non-redundant coding of neural networks

Genetic algorithms are based on a representation independent of the problem, usually the representation is a string of binary, integer or real numbers. This representation (the genotype) codifies a network (the phenotype). This is a dual representation scheme. The interpretation function maps between the elements in the recombination space (on which the search is performed) and the subset of structures that can be evaluated as potential task solutions. The ability to create better solutions in a genetic algorithm relies mainly on the operation of *crossover*. This operator forms offspring by recombining representational components from two members of the population.

The benefits of crossover come from the ability of forming connected substrings of the representation that correspond to above-average solutions[3]. This substrings are called *building blocks*. Crossover is not effective in environments where the fitness of an individual of the population is not correlated with the expected ability of its representational components[21]. Such environments are called *deceptive*[22]. Deception is a very important feature in most representations of neural networks, so crossover should be avoided in evolutionary neural networks[7].

One of the most important forms of deception arises from the many-to-one mapping from genotypes in the representation space to phenotypes in the evaluation space. Two networks which order their hidden nodes differently will have different genotypes in most representation schemes, but these two networks are functionally equivalent. The existence of networks functionally equivalent and with different encodings makes the evolution inefficient, and it is unclear whether crossover operator would produce more fitted individuals from two members of the population. This problem is usually termed as the *permutation problem*[23][24] or the *competing conventions problem*[25].

One of the most common problems of the application of genetic algorithms to neural networks is the *permutation problem*[23][24] or the *competing conventions problem*[25]. It is obvious that we can obtain a number of structurally different networks that implement the same input-output mapping. These structurally different will also have different genetic representations.

To solve this problem D. Thierens[26] proposed a non-redundant genetic coding of neural networks that avoids the permutation problem. The codification is the following:

- To remove the hidden node redundancy we flip the sign of the bias weight, the incoming and the outgoing weights of each hidden node whenever the bias weight is negative, so only hidden neurons with a positive bias are allowed. This way we reduce the 2^n functional equivalent networks to just one representative of the group.
- The second redundancy that we have discussed is at the level of each hidden layer and is caused by the permutation of the hidden nodes of each layer. To eliminate this redundancy we rearrange all hidden neurons in each layer such that the bias weights are sorted in ascending order. This way the $n!$ functional equivalent networks are eliminated and since the sort process does not interfere with the previous sign flipping all the $2^n n!$ equivalent networks are transformed into a *canonical* form.

IV. CIXL2 applied to neural network evolution

An individual of the population of networks is just a string of real number following the codification that we have explained above. With this codification we can apply the CIXL2 crossover to an individual.

The evolution of the population of networks is the following, once the fitness of each individual has been obtained:

- The best 10% of the population is selected for reproduction. This elitist selection prevent the elimination of the best solutions along the evolutionary process.
- To fill the 60% of the new population we select by means of a roulette algorithm individuals of the population and perform a CIXL2 operator.
- To fill the 10% of the new population we select by means of a roulette algorithm individuals of the population and perform a parametric mutation that consists of a basic back-propagation algorithm.
- To fill the 10% of the new population we select by means of a roulette algorithm individuals of the population and perform a parametric mutation that consists of a random step in the space of network weights. We add to every weight of the network a small value taken form a normal distribution of 0 mean and small variance.
- To fill the 10% of the new population we select by means of a roulette algorithm individuals of the population and perform a structural mutation that consists of one of the two following operations:
  - Removing all the weights of a node. The values of all the connection of a randomly selected node are set to 0.
  - Removing a connection. A connection is randomly selected and its weight is set to 0.

This process is illustrated on Figure 2.

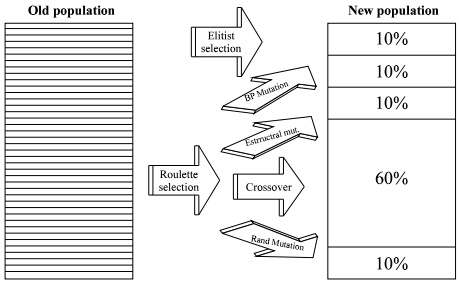

Fig. 2. Generation of the new population of networks.

V. Experimental setup

The tests were conducted following the guidelines of L. Prechelt[27]. Each set of available data was divided into three sets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing the generalization of the individuals.

For the training of the MLP networks with BP we used the method of cross-validation and early-stopping[28]. The networks were trained until the error over the validation set started to grow. The number of hidden nodes was chosen by trial and error.

TABLA I
Parameters of the neural network evolution for the two problems

Parameter	Value
Number of networks	500
Number of hidden nodes	6
Elitist reproduction	10%
Back-Prop mutation	10%
Random mutation	10%
Structural mutation	10%
Crossover	60%
$1-\alpha$ (CIXL2 crossover)	0.7
n (CIXL2 crossover)	30
α (BLX-α crossover)	0.5
Back-prop iterations	500
Back-prop learning coefficient	$\eta = 0.1$

In order to assess the validity of application of the CIXL2 crossover to neural network evolution we compare CIXL2 with BLX-α crossover[29]. The definition of this crossover is the following.

The crossover generates just one offspring $\beta^s = \{\beta_1^s, \beta_2^s, \ldots, \beta_i^s, \ldots, \beta_n^s\}$ from two parents $\beta^{\mathbf{P1}}$, $\beta^{\mathbf{P2}}$, where β_i^s is randomly selected from the interval $[\beta_{min} - I\alpha, \beta_{max} + I\alpha]$, being $\beta_min = \min(\beta_i^{p_1}, \beta_i^{p_2})$, $\beta_max = \max(\beta_i^{p_1}, \beta_i^{p_2})$, and $I = \beta_{max} - \beta_{min}$. The exploration/exploitation compromise of this crossover depends on the value of the α parameter. From the experiments performed in [30] the optimum value obtained of α is $\alpha = 0.5$. This is the value that we have used in our experiments.

The experiments were made with the same parameters for the two crossovers.

For training the networks with a classical back-propagation algorithm we have choose the `quickprop`[31] algorithm, as it is one of the best performing back-propagation algorithms. We use a two-layer perceptron trained using cross-validation and early-stopping[28]. The experiments were made using the simulator Nevada Backpropagation[32].

We have also compared the performance of our model with the C4.5[33] algorithm, using the code implemented by the author of the algorithm, which uses a decision tree, and with a modular neural network using the adaptive mixture of local experts model. For this model we also used cross-validation and early-stopping. The experiments were made using the simulator NeuralWorks Professional II Plus[34].

In order to assess the true difference in performance between CIXL2 and the other methods we conducted three statistical tests (a summary of the results is shown on Table V). First, we corroborated that the distribution of the errors was normal by means of a Kolmogorov - Smyrnov test[35]. Next, we tested the hypothesis that the errors from the experiments with Symbiont and MLP had the same variance performing an F test[36]. Finally we performed a t test that allowed us to asseverate whether the error obtained with Symbiont was significatively different than the error obtained with other methods. All tests were made with the R statistical package[37].

A. Pima Indian data set

This data set is from the UCI machine learning repository. The data set contains data of 768 individuals, all of them females at least 21 years old of Pima Indian heritage. The patterns are divided into two classes. The class of each pattern shows whether the patient shows signs of diabetes according to the World Health Organization criteria.

There are 8 attributes for each pattern, all of them are real valued. Former results can be found in [38]. Following this previous work and the recommendations of L. Prechelt([27] and [39]) we have divided the data set in 384 patterns for training, 192 patterns for validation, and 192 patterns for generalization. The data set contains 500 instances of class 1 and 268 instances of class 2. Three permutations of the data set are used, and ten experiments are carried out on each one.

B. Heart Disease data set

This data set comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The database contains 13 attributes, which have been extracted from a larger set of 75, that correspond to the results of various medical tests carried out on a patient. The goal is the prediction of the presence or absence of heart disease in those patients. The original data set had five classes, considering four degrees of heart disease. The database originally contained 303 examples but six of them had missing values, and 27 of the remaining were retained in case of dispute, leaving a final total of 270 (the problem is described more deeply in [40]).

C. Credit card problem

This data set is also from the UCI Machine Learning Repository. The set contains data from applications to an Australian bank to get a credit card. There are two classes, meaning whether the application was granted (44.5% of the patterns) or denied (55.5%). Each record has 14 attributes, for confidentiality all attributes and values are not explained in the original data set.

This data set is very interesting because it has two important features. First, it contains many missing values (there are missing values in 5% of the records). Second, the attributes are of very different kind: continuous (1, 2, 10, 13, and 14), binary (0, 8, 9, and 11), and nominal (3, 4, 5, 6, and 12). The binary and nominal attributes have been codified using a 1-out-of-n code so the data set used for training the networks had 51 inputs and 2 outputs.

VI. Results

In all the tables we show, for each data set, the averaged error of classification over 30 repetitions on each permutation of the data set, the standard deviation, the best and worst individuals, and the averaged number of nodes and connections of the best networks of each experiment. The measure of the error is the following:

$$E = \frac{1}{P}\sum_{i=1}^{P} e_i, \tag{3}$$

where P is the number of patterns and e_i is 0 if pattern i is correctly classified, and 1 otherwise.

Table II shows the results obtained in classifying the Pima data set. We can see how the CIXL2 crossover is the best performing method. It clearly outperforms the other algorithms. The difference in performance is statistically significant at a confidence level of 10% as it is shown on Table V.

Table III shows the results obtained in classifying the Heart data set. We can see how the CIXL2 crossover is the best performing method together with the MLP trained using a `quickprop` algorithm. They clearly outperform the other algorithms, with a difference that is statistically significant as it is shown on Table V.

Table IV shows the results obtained in classifying the Card data set. We can see how the performance of the classification methods is very similar. Nevertheless, it is interesting to note that CIXL2 performs better than BLX-α crossover. The best performing method for this problem is C4.5. It is a natural result as many of the features of the patterns are not numerical and C4.5 is a hierarchical classification method.

VII. Conclusions

We have successfully introduced the *Confidence Interval Based Crossover* in the field of evo-

TABLA II
ERROR RATES FOR PIMA DATA SET

Model	Training				Generalization			
	Mean	StD	Best	Worst	Mean	StD	Best	Worst
CIXL2	0.2278	0.0127	0.2083	0.2535	0.2090	0.0187	0.1719	0.2604
BLX-α	0.2398	0.0084	0.2188	0.2569	0.2174	0.0172	0.1875	0.2552
`quickprop`	0.2313	0.0069	0.2222	0.2413	0.2219	0.0189	0.1979	0.2604
C4.5	0.1718	–	–	–	0.2552	–	–	–
Mixture of experts	0.2528	0.0135	0.2413	0.2847	0.2485	0.0211	0.2135	0.2865

TABLA III
ERROR RATES FOR HEART DATA SET

Model	Training				Generalization			
	Mean	StD	Best	Worst	Mean	StD	Best	Worst
CIXL2	0.1261	0.0103	0.1089	0.1535	0.1451	0.0255	0.1029	0.2059
BLX-α	0.1584	0.0158	0.1386	0.1881	0.1618	0.0294	0.1029	0.2500
`quickprop`	0.0975	0.0226	0.0495	0.1188	0.1471	0.0196	0.1176	0.1765
C4.5	0.0644	–	–	–	0.2206	–	–	–
Mixture of experts	0.0500	0.0135	0.0347	0.0743	0.1853	0.0210	0.1618	0.2206

TABLA IV
ERROR RATES FOR CARD DATA SET

Model	Training				Generalization			
	Mean	StD	Best	Worst	Mean	StD	Best	Worst
CIXL2	0.1165	0.0058	0.1062	0.1351	0.1401	0.0105	0.1221	0.1686
BLX-α	0.1315	0.0066	0.1216	0.1486	0.1473	0.0208	0.1163	0.1919
`quickprop`	0.0861	0.0112	0.0618	0.1004	0.1407	0.0094	0.1221	0.1512
C4.5	0.0830	–	–	–	0.1337	–	–	–
Mixture of experts	0.1120	0.0097	0.0907	0.1236	0.1401	0.0043	0.1337	0.1453

lutionary neural networks. We have achieved a performance comparable, at least, to other classical methods of neural network training. We have also prove that the CIXL2 crossover performs better than the BLX-α crossover. This results is very important as BLX-α is one of the best operators described in the bibliography of genetic algorithms.

As work for the future we are developing a model to classify the neurons in order to allow the crossover of most similar neurons, to avoid generational gaps.

ACKNOWLEDGEMENTS

The authors would like to acknowledge R. Moya-Sánchez for her helping in the final version of this paper.

BIBLIOGRAPHY

[1] S. Haykin, *Neural Networks – A Comprehensive Foundation*, Prentice – Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[2] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 9, no. 87, pp. 1423–1447, 1999.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley, Reading, MA, 1989.

[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer–Verlag, New York, 1994.

[5] A. J. F. van Rooij, L. C. Jain, and R. P. Johnson, *Neural Networks Training Using Genetic Algorithms*, vol. 26 of *Series in Machine Perception and Artificial Intelligence*, World Scientific, Singapore, 1996.

[6] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, May 1997.

[7] P. J. Angeline, G. M. Saunders, and Jordan B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54 – 65, January 1994.

[8] S. V. Odri, D. P. Petrovacki, and G. A. Krstonosic, "Evolutional development of a multilevel neural network," *Neural Networks*, vol. 6, pp. 583–595, 1993.

TABLA V

STATISTICAL TESTS FOR THE THREE PROBLEMS

Test	CIXL2	BLX-α	quickprop	Mixture
Pima				
K-S	0.5256	0.6718	0.3391	0.7771
F	–	0.6671	0.8960	0.6496
t	–	0.0778	0.0091	0.0000
Heart				
K-S	0.5717	0.3158	0.1822	0.2080
F	–	0.4518	0.1123	0.3199
t	–	0.0225	0.7327	0.0000
Card				
K-S	0.4107	0.6252	0.1494	0.1393
F	–	0.0004	0.4390	0.0000
t	–	0.0990	0.8249	1.0000

[9] R. Smalz and M. Conrad, "Combining evolution with credit apportionment: A new learning algorithm for neural nets," *Neural Networks*, vol. 7, no. 2, pp. 341 – 351, 1994.

[10] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Transactions on neural networks*, vol. 5, no. 1, pp. 39–53, January 1994.

[11] M. V. Borst, *Local structure optimization in evolutionary generated neural networks architectures*, Ph.D. thesis, Leiden University, The Netherlands, August 1994.

[12] B. A. Whitehead and T. D. Choate, "Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, July 1996.

[13] G. Bebis, M. Georgiopoulos, and T. Kasparis, "Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization," *Neurocomputing*, vol. 17, pp. 167–194, 1997.

[14] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 28, no. 3, pp. 417–425, June 1998.

[15] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evolutionary Computation*, vol. 4, no. 5, 1998.

[16] G. F. Miller, P. M. Todd, and S. U. Hedge, "Designing neural networks," *Neural Networks*, vol. 4, pp. 53–60, 1991.

[17] P. J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, Wiley, New York, NY, 1994.

[18] D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas, "Crossover operator effect in function optimization with constraints," in *Parallel Problem Solving from Nature – PPSN VII*, J. Merelo, P. Adamidis, H-G. Beyer, J. Fernández, and H-P. Schwefel, Eds., Granada, september 2002, number 2439 in Lecture Notes in Computer Science, pp. 184–193, Springer-Verlag.

[19] C. Hervás-Martínez, D. Ortiz-Boyer, and N. García-Pedrajas, "Theoretical analysis of the confidence interval based crossover for real-coded genetic algorithms," in *Parallel Problem Solving from Nature – PPSN VII*, J. Merelo, P. Adamidis, H-G. Beyer, J. Fernández, and H-P. Schwefel, Eds., Granada, september 2002, number 2439 in Lecture Notes in Computer Science, pp. 153–161, Springer-Verlag.

[20] D. Ortiz-Boyer, C. Hervás-Martínez, and J. Muñoz-Pérez, "Cixl2: A new crossover operator based on population features," Submitted to IEEE Transactions on Evolutionary Computation.

[21] D. E. Goldberg, "Genetic algorithms and Walsh functions: Part 2, deception and its analysis," *Complex Systems*, vol. 3, pp. 153 – 171, 1989.

[22] D. E. Goldberg, "Genetic algorithms and Walsh functions: Part 1, a gentle introduction," *Complex Systems*, vol. 3, pp. 129 – 152, 1989.

[23] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: Using genetic algorithms with connectionist learning," Tech. Rep. CS90-174, Computer Science Engineering Department, University of California-San Diego, Feb. 1991.

[24] P. J. B. Hancock, "Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification," in *Proc. Int. Workshop of Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, D. Whitley and J. D. Schaffer, Eds., Los Alamitos, CA, 1992, pp. 108–122, IEEE Computer Soc. Press.

[25] J. D. Schaffer, L. D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks*, L. D. Whitley and J. D. Schaffer, Eds., Los Alamitos, CA, 1992, pp. 1–37, IEEE Computer Society Press.

[26] D. Thierens, "Non-redundant genetic coding of neural networks," in *Proc. of the 1996 IEEE International Conference on Evolutionary Computation*, Piscataway, NJ, 1996, pp. 571–575, IEEE Press.

[27] L. Prechelt, "Proben1 – A set of neural network benchmark problems and benchmarking rules," Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, September 1994.

[28] W. Finnoff, F. Hergert, and H. G. Zimmermann, "Improving model selection by nonconvergent methods," *Neural Networks*, vol. 6, pp. 771–783, 1993.

[29] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed., pp. 187–202. Morgan Kaufmann, San Mateo, 1993.

[30] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, pp. 265–319, 1998.

[31] T. J. Sejnowski, G. E. Hinton, and D. S. Touretzky, Eds., *Faster-learning variations on back-propagation: An empirical study*, 1988 Connectionist Models Summer School, San Mateo, CA, 1988. Morgan Kaufmann.

[32] D. Carlson, *NevProp 3 ©User Manual*, University of Nevada, Reno, NE, 1996.

[33] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, 1993.

[34] NeuralWare, *Neural Computing: A Technology Handbook for Professional II/Plus*, NeuralWare Inc., Pittsburgh, PA, 1993.

[35] W. J. Conover, *Practical nonparametric statistics*, John Wiley & Sons, New York, 1971.

[36] T. W. Anderson, *An introduction to multivariate statistical analysis*, Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 2nd edition, 1984.

[37] R Development Core Team, *An introduction to R*, April 2000.

[38] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the Symposium on Com-*

puter Applications and Medical Care. 1988, pp. 261–265, IEEE Computer Society Press.

[39] L. Prechelt, "A quantitative study of experimental evaluation of neural network learning algorithms," *Neural Networks*, vol. 9, pp. 457–462, 1996.

[40] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *American Journal of Cardiology*, vol. 64, pp. 304–310, 1989.