International Journal of Pure and Applied Mathematics
Volume 104 No. 3 2015, 421-436
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
**url:** http://www.ijpam.eu
**doi:** http://dx.doi.org/10.12732/ijpam.v104i3.11

ijpam.eu

# MATRICIAL REPRESENTATION OF INDIVIDUALS IN CONTINUOUS GENETIC ALGORITHMS CAN IMPROVE THE EFFICIENCE IN RADIAL BASIS FUNCTIONS NEURAL NETWORKS TRAINING

J.F. da Mota[1 §], P.H. Siqueira[2], L.V. de Souza[3]

Paraná State University
733 Com. Norberto Marcondes Av.
Campo Mourão, 87303-100, BRAZIL
[2]Federal Parana University
Polytechnic Center
Curitiba, 81531-970, BRAZIL
[3]Federal Parana University
Polytechnic Center
Curitiba, 81531-970, BRAZIL

**Abstract:** In this paper we've compared two GA - Genetic Algorithm - based approaches for computing the weight matrix of a RBFNN - Radial Basis Function Neural Network. The first, named GA, was based in Michalewicz's Operators for Continuous Genetic Algorithms and the other, named modGA, was based in extending these operators to matricial individuals, consequently proposing new operators. The main objective was verifying if the new approach could reduce the number of iterations (generations) necessary to compute the weight matrix. Six datasets was tested and in 50% of them, that hypothesis was confirmed.

**AMS Subject Classification:** 92B20
**Key Words:** RBF neural networks, genetic algorithms, classification, RBF training

Received: July 30, 2015

[§]Correspondence author

## 1. Introduction

For a long time scientists have been trying to develop methods for pattern classification problems wich may help a decision taker in a uncertain scenario. Several mathematical and statistical methods have been developed and this have been caused improvements in existents methods and also the development of new methods wich may reduce the computational effort, offer better results or both.

Every single existent method offer a different option relative to speed and quality of the prediction or classification. The most difficult task, wich every researcher yearns, is developing a method capable of get a high accuracy in a minimal time given the need for speed of the on-line era. By "high accuracy" we mean an error as small as possible and a correct classified percentual as big as possible, considering a pattern classification problem.

A Metaheuristic is a generic heuristic method to solve Operations Research problems. Usually combinatorial, pattern recognition or time series forecasting problems. However, it's also possible to solve function optimization problems using a metaheuristic.

The idea of building up mathematical models that can simulate the human neurons is considered by many scientists as brilliant and useful for decision takers of several knowledge areas. These models are known as the ANN - Artificial Neural Networks. According to Haykin [3], a neural network is a parallel processing machine capable of change experimental knowledge into useful information. This information can be used by decision takers, for instance.

Another biological inspired idea that came up over the time was the idea of transforming Darwinian's evolution into mathematical rules for finding a solution to a problem. This idea culminate in the criation of a technique named "Genetic Algoritms". According to Holland [4], Genetic Algoritm(s) (GA) are basically a set of algorithms based on the principles of evolutionary biology stated by Charles Darwin in his book known worldwide *The Evolution of Species*. The main objective of the GAs is to optimize functions.

Both ANN and GA are metaheuristic techniques, and there are a plenty more of them. However, since the objective of this paper lies on that two (ANN and GA), we are not going to provide any details on another techniques. For a brief description on most of metaheuristic techniques, we recommend [1], [3] and [2].

Getting into a bigger picture, the Computational Intelligence field of study embraces the Metaheuristic subject and it's not a novelty on this field the hybrid-techniques modeling, combining two or more algorithms in order to obtain a more efficient one.

Yu et al. [12] proposed a hybrid Swarm Optimization and Genetic Algorithm optimized Radial Basis Function Neural Network (PSO-GA-RBF) for predicting the annual electricity demand. They concluded that the PSO-GA-RBF generates a simpler network structure or higher estimation precision than other selected ANN - Artificial Neural Networks models. The other conclusion states that the electricity demand will grow rapidly at average rates in the interval 9.7–11.5% a year.

In Zhao et al. [13] the PSO - Particle Swarm Optimization was used to obtain an optimum width for the neurons centers. They concluded that the PSO algorithm improved the RBFNN - Radial Basis Function Neural Network accuracy in the classification problem. They also concluded their proposal was a better classifier than a LS-SVM - Least-square Support Vector Machine based approach, wich is a state-of-the-art classifier.

In Mota et al. [9] we can find an application of a GA to obtain the weight matrix of a RBFNN - *Radial Basis Function Neural Network* for pattern classification in four datasets. The operators they used was the Michalewicz's Operators [8] and the GA-RBFNN approach did not overcome the traditional (pseudo-inverse) method in 75% of the datasets.

The researchers Li and Ling [6] applied the idea of getting the weights of the hidden layer of a RBFNN to develop a generalized model predictive control in a power generating unit that had two input and two output variables. The experiment was to compare the performance of the proposed algorithm (called GA-RBF) with the performance of a Multilayer Perceptron using the back-propagation training algorithm. The proposed technique has a mean square error of the order of $10^{-5}$, while the Perceptron tested missed about $10^{-1}$.

The research [5] compared four algorithms to train a RBFNN: Bee Colony, GA, Kalman filter and gradient descent. They tested three databases available in [7] and one database application proposed by the authors. In all of tested datasets the Bee Colony algorithm performances was slightly higher than the GA, both featuring an accuracy above 90% accuracy the test set in the database in [7] and over 70% of the database application in sensors proposed by the authors.

A relatively interesting approach can be found in [11], because in this study all the parameters of RBFNN are converted to individuals (vectors) and then GA is run. Subsequently to the end of each generation, each individual components are assigned to each parameter of a RBFNN taking elements of the vector according to the amount of information necessary for each parameter. The work is related to time series forecasting of electric charge in a city and the largest absolute error in the set of tests was 3.05%.

In our research, we have computationally implemented the same hybrid algorithm you can find on [9], wich used the Michalewicz's Operators for "string" individuals in a GA population and we also proposed an adaptation of these operators considering matricial individuals.

## 2. Methodology

### 2.1. A New Way of Looking at Individuals in a GA Population

Traditionally, we look to an individual as a genetic information vector, a "string" putting side by side genes in a GA population, as we can see in Figure 1. However, not always we have that kind of individual, a vector. Sometimes, we have to deal with matricial individuals, like when we are up to compute the weight matrix of a RBFNN.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1,5 | 9,2 | 31,2 | 6,8 | 0,8 | 7,4 | 1,3 | 5,7 |

Figure 1: Traditional GA individual

That means if we are going to compute a weight matrix of a RBFNN using the GA approach, it would be necessary to transform the matricial individuals in vectors in order to apply the operators and after that make another transformation to calculate the fitnesses of the population.

A consequence of adapting the operators to matricial individuals it's the way we set a GA individual, it also brings advantages when implementing the algorithm, partitioning an individual not in genes anymore, but vertically or horizontally fixed chains of genes (Figure 2). Thus, each row/column represents a chain of genes that is responsible for one attribute of a particular individual.

The main idea behind this formulation is that the attributes of an individual, as the traditional GA suggests, lies on chains of genes. The difference between our approach and the original GA is that an operator can't break these chains of genes as it happens.

One of the reasons for not breaking the chains of genes is to avoid losses of "desirable" characteristics in terms of fitness. Of course it can still happen, but when you have a "string" individual, it's more likely to happen, because the original operators are not designed to matrices. It will not try to evolve the

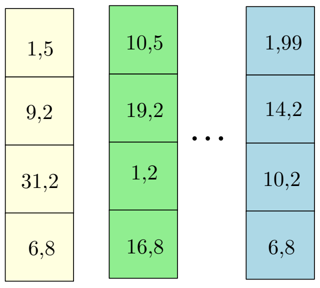

Figure 2: Matricial GA individual

population looking at the chains of genes like if they are "one piece" responsible for one unique attribute.

### 2.2. Adapting Operators

#### 2.2.1. Matricial Arithmetical Crossover

This operator is a pure extension of the vectorial case. It consists from individuals $p_1$ and $p_2$, generate,

$$c_1 = \beta p_1 + (1-\beta)\, p_2 \text{ e } c_2 = \beta p_2 + (1-\beta)\, p_1 \text{ with } \beta \sim U(0,1).$$

#### 2.2.2. Heuristic Matricial Crossover

This operator take the individuals $p_1$ and $p_2$, such that $p_1 \leq p_2$, and generate

$$c = p_2 + \beta\,(p_2 - p_1) \text{ with } \beta \sim U(0,1).$$

#### 2.2.3. Total Horizontal Crossover (THC)

This operator assumes that the genetic chains responsible for the individuals attributes are vertically disposed. Thus, draws up a column as crossover point and, from that column to right, gene strings are exchanged between parents, as illustrated in Figure 3.

In other words, it takes $p_1$ and $p_2$, both $m \times n$, draws up an integer $r \in (1, n)$ and from the $r$-th column (genetic chain) to right the parents exchange their genes.

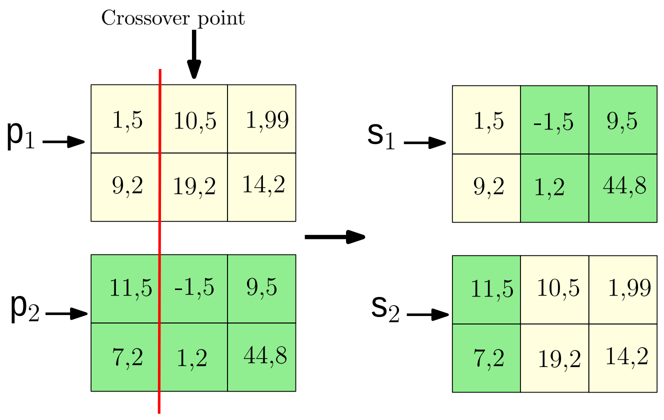

Figure 3: THC Operator

### 2.2.4. Total Vertical Crossover (TVC)

This operator assumes that the genetic chains that define the characteristics an individual are horizontally disposed. Thus, draws up a row as crossover point and, from that row down, gene strings are exchanged between parents, as illustrated in Figure 4.

Similarly to the THC operator, it takes $p_1$ and $p_2$, both $m \times n$, draws up an integer $r \in (1, n)$ and from the $r$-th row (genetic chain) the parents exchange their genes.

### 2.2.5. Partial Horizontal Crossover (PHC)

This operator is a particular case of the THC and also assumes that the genetic chains responsible for the individuals attributes are vertically disposed.

However, this operator exchanges only a part of the genetic chains between parents. Thus, draws up a column and a row of the matrix as a crossing point and, from this gene down and left, parents exchange their genes as illustrated in Figure 5.

This operator takes $p_1$ and $p_2$ both $m \times n$, draws up two integers values $r_1 \in (1, m)$ and $r_2 \in (1, n)$ and from the $r_1$-th row down and left from $r_2$-th column the parents exchange their genes.

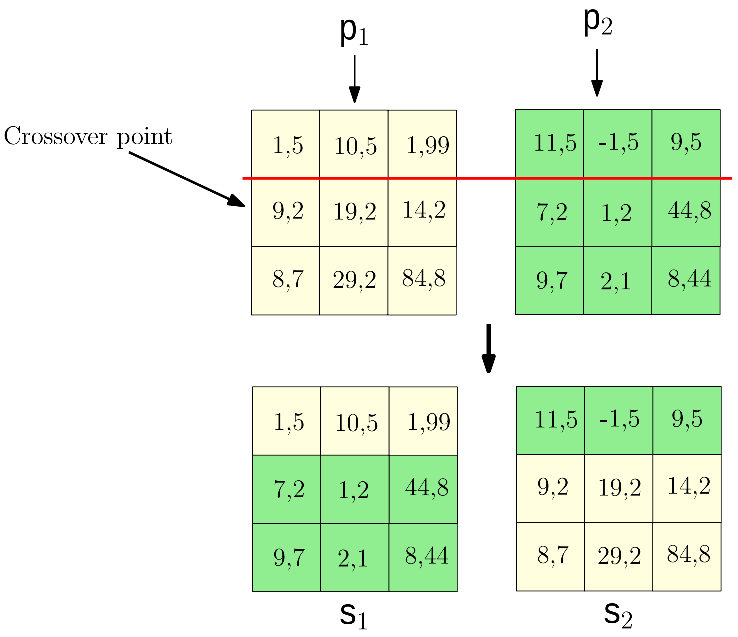

Figure 4: TVC Operator

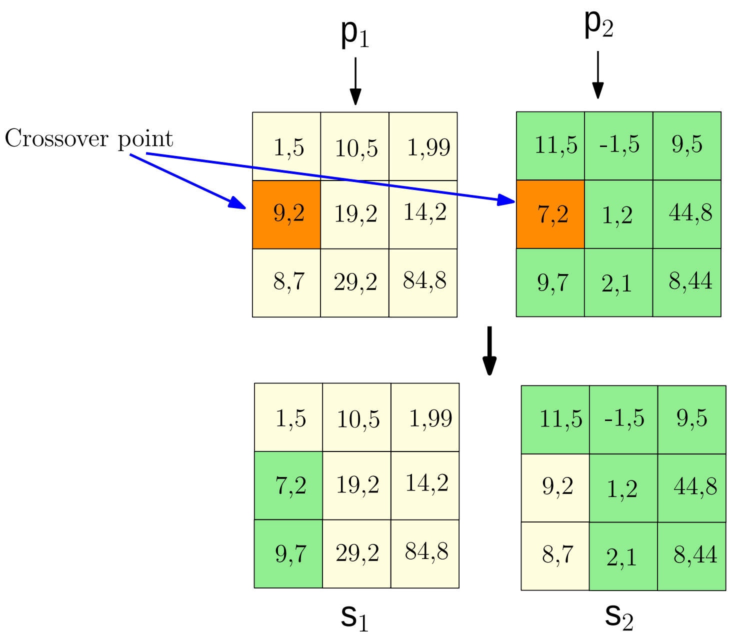

Figure 5: PHC Operator

### 2.2.6. Partial Vertical Crossover (PVC)

This operator is a particular case of the TVC and also assumes that the genetic chains responsible for the individuals attributes are horizontally disposed.

However, this operator exchanges only a part of the genetic chains between parents. Thus, draws up a column and a row of the matrix as a crossing point and, from this gene up and right, parents exchange their genes as illustrated in Figure 6.

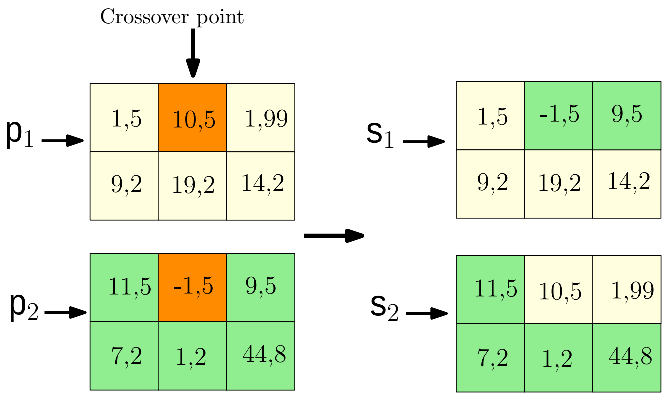

Figure 6: PVC Operator

This operator takes $p_1$ and $p_2$ both $m \times n$, draws up two integers values $r_1 \in (1, m)$ and $r_2 \in (1, n)$ and from the $r_1$-th row up and right from $r_2$-th column the parents exchange their genes.

### 2.2.7. Limit Mutation

Given a $m \times n$ individual, say $p$, draws up two integers $r_1 \in (1, m)$ e $r_2 \in (1, n)$,

$$p(r_1, r_2) = r,$$

where $r \in \{a, b\}$. The values $a$ and $b$ are the feasible interval limits on the space search for $r$.

### 2.2.8. Uniform Mutation

This operator is also a pure extension of the vectorial case. So given an individual $p$, $m \times n$, draws up two integers $r_1 \in (1, m)$ and $r_2 \in (1, n)$,

$$p(r_1, r_2) = r,$$

where $r \sim U(a, b)$. The values $a$ and $b$ are the feasible interval limits on the space search for $r$. In general, there are no boundaries for $a$ and $b$, but we empirically fix these values to help the GA search.

### 2.2.9. (Multiple) Non Uniform Mutation

Given a $m \times n$ individual named $p$, draws up two integers $r_1 \in (1, m)$ e $r_2 \in (1, n)$,

$$p(r_1, r_2) = \begin{cases} p(r_1, r_2) + (b - p(r_1, r_2))f(G), & \text{if } r < 0,5 \\ p(r_1, r_2) - (p(r_1, r_2) - a)f(G) & \text{, otherwise,} \end{cases}$$

com,

$$f(G) = \left[ s \left( 1 - \frac{G}{G_{\max}} \right) \right]^b,$$

where $r, s \sim U(0, 1)$. The values $a$ and $b$ are the feasible interval limits on the space search for $r$, $G$ is the current generation (iteration) and $G_{\max}$ is the maximum of generations.

The multiple non uniform mutation consists of applying the non uniform mutation in two or more genes of an individual even all of its genes.

## 3. Experiment

### 3.1. Datasets and Preparations

In our experiments, we've used the already well-known classification problems - Cancer (cancer), Contraceptive Method Choice (cmc), Car Evaluation (careval), Hepatitis (hepatitis), Iris (iris) and Pageblocks (pageblocks) - data sets that are found in Lichman13 [7]. The characteristics of the sets used in the tests are shown in Table 1, the number of variables in standard input, output and the number of patterns in each set.

In all datasets we've separated 60% of observations for the training set, 20% for the validation set and the remaining 20% for the test set. To avoid any scale problem, the data was normalized according to its means and standard-deviations.

Table 1: Datasets Features

| Dataset | Entries | Outputs | Training | Validation | Test |
|---|---|---|---|---|---|
| cancer | 9 | 2 | 419 | 140 | 140 |
| cmc | 9 | 3 | 884 | 295 | 295 |
| careval | 6 | 4 | 1037 | 346 | 346 |
| hepatitis | 19 | 2 | 93 | 31 | 31 |
| iris | 4 | 3 | 90 | 30 | 30 |
| pageblocks | 10 | 5 | 3284 | 1095 | 1095 |

### 3.2. Describing the Experiment

We developed a MATLAB algorithm to run the experiment as shown in Figure 7. In that routine there are three interest variables named CCP - Correct Classified Percentual, RT - runtime and NI - number of iterations.

```
ALGORITHM: EXPERIMENT ROUTINE
  For (each dataset)
    For (each technique)
      For (each configuration)
        executions = 0
        While (executions < 40)
          Obtain:
            CCP - Correct Classified Percentual of test set
            RT - Runtime
            NI - Number of iterations (when possible)
            executions = executions + 1
        end While
      end For
    end For
  end For
```

Figure 7: Experiment Algorithm

After we ran the experiment, each technique best result was saved in order to verify which approach provided a better result for classification purposes in each dataset. Thus, for both traditional GA and our proposal were calculated the average value and the standard-deviation for each interest variable CCP, RT and NI. The configuration with the greater CCP was considered the best.

Lately, for each dataset, we performed a *Mann-Whitney* test at 5% significance to search for significative difference among the CCP, RT and NI means. Obviously, for CCP we are interested in the higher average value and, for RT and NI, it's the lower average values.

## 4. Results and Discussion

### 4.1. Best Configurations

Table 2: Best unique execution configuration for GA and modGA

| **Dataset** | **Technique** | **Best Configuration** | **CCP** | $\kappa$ |
|---|---|---|---|---|
| cancer | GA | $[14; 150; 0,50; 0,10]$ | 100% | $0,95$ |
| | modGA | $[2; 500; 0,70; 0,05]$ | 100% | $0,95$ |
| cmc | GA | $[14; 150; 0,70; 0,05]$ | 50% | $0,96$ |
| | modGA | $[20; 1000; 0,50; 0,10]$ | 49% | $0,96$ |
| careval | GA | $[12; 1000; 0,70; 0,05]$ | 76% | $0,95$ |
| | modGA | $[8; 1000; 0,70; 0,10]$ | 79% | $0,95$ |
| hepatitis | GA | $[2; 150; 0,50; 0,05]$ | 93% | $0,89$ |
| | modGA | $[4; 150; 0,60; 0,10]$ | 97% | $0,89$ |
| iris | GA | $[2; 1000; 0,50; 0,05]$ | 100% | $0,19$ |
| | modGA | $[2; 150; 0,60; 0,05]$ | 100% | $0,19$ |
| pageblocks | GA | $[18; 1000; 0,60; 0,10]$ | 92% | $0,98$ |
| | modGA | $[10; 150; 0,70; 0,10]$ | 93% | $0,98$ |

Table 2 shows the best configuration founded for both traditional GA and our approach, named modGA, during this experiment. For each approach it was tested 450 different configurations, varying the amount of neurons in the hidden layer, initial population size, crossover and mutation probabilities. Take $[2; 150; 0,65; 0,01]$ for instance, it means two hidden layer neurons, initial population size of 150, crossover probability of $0,65$ (or 65%) and mutation probability of $0,01$ (or 1%).

We notice in Table 2 that there is not any trend between the datasets we've tested, not GA and modGA, regarding the parameters values. That is very common to happen when it comes to classification techniques, mathematical or statistical. And that is because these methods are essentially adaptable and sensitive to each problem/dataset characteristic.

For the cancer dataset, both GA and modGA achieved high accuracy in classification (CPP), both 100%, and a *kappa* index [10] also considered good according to literature.

For the cmc dataset, both GA and modGA had an accuracy unaceptable, according to traditional literature, 50% and 49% respectively, so in a real life application, both models could not be used.

For the careval, hepatitis and pageblocks all CCPs was considered good according to literature, varying in the interval 73–100%, and *kappa* in the interval 0,88–0,98.

A curious fact did happened on the dataset iris. Although both GA and modGA CCP are 100%, both *kappa* indexes are 0,19. That means the concordance between the almost infinite arrangement of the training, validation and test datasets are low. In other words, there is a high influence of this arrangement in the CCP.

## 4.2. Average Comparison

Table 3: Best configurations for mean CCPs of *GA* and *modGA*

| **Dataset** | **Technique** | **Best Configuration** |
|---|---|---|
| cancer | GA | $[14; 500; 0,70; 0,05]$ |
| | modGA | $[4; 150; 0,60; 0,10]$ |
| cmc | GA | $[14; 150; 0,60; 0,05]$ |
| | modGA | $[8; 500; 0,60; 0,10]$ |
| careval | GA | $[6; 500; 0,60; 0,10]$ |
| | modGA | $[16; 500; 0,70; 0,10]$ |
| hepatitis | GA | $[6; 500; 0,70; 0,05]$ |
| | modGA | $[16; 150; 0,60; 0,10]$ |
| iris | GA | $[6; 500; 0,60; 0,05]$ |
| | modGA | $[4; 500; 0,70; 0,10]$ |
| pageblocks | GA | $[4; 150; 0,70; 0,05]$ |
| | modGA | $[6; 500; 0,60; 0,10]$ |

Considering the goal of making a statistical comparison, from the experiment we've obtained the average value and the standard-deviation for the interest variables CCP, RT and NI.

Considering the average value of 40 executions, Table 3 shows the best configurations for GA and modGA. Some cases, the configurations are different

Table 4: Mean CCP, Mean RT and NI for each dataset

| **Dataset** | **Technique** | **CCP (%)** | | **RT (s)** | | **NI** | |
|---|---|---|---|---|---|---|---|
| | | $\bar{x}$ | $s$ | $\bar{x}$ | $s$ | $\bar{x}$ | $s$ |
| cancer | GA | $95,7$ | $1,4$ | $11,7$ | $2,4$ | $63,7$ | $12,1$ |
| | modGA | $95,5$ | $1,5$ | $3,2$ | $0,5$ | $52,1$ | $8,2$ |
| | ***p*-values** | $0,65$ | | $10^{-7}$ | | $0,00$ | |
| cmc | GA | $37,1$ | $7,5$ | $12,5$ | $1,6$ | $60$ | $4,9$ |
| | modGA | $39,6$ | $7,4$ | $15,1$ | $2,3$ | $51,5$ | $7,7$ |
| | ***p*-values** | $0,19$ | | $10^{-3}$ | | $10^{-5}$ | |
| careval | GA | $70,9$ | $1,9$ | $13,6$ | $3$ | $63,3$ | $11,8$ |
| | modGA | $72,9$ | $3,2$ | $30,6$ | $8,1$ | $76,1$ | $12,2$ |
| | ***p*-values** | $0,04$ | | $10^{-7}$ | | $0,02$ | |
| hepatitis | GA | $78,9$ | $7,7$ | $4,1$ | $0,3$ | $32$ | $0$ |
| | modGA | $79,7$ | $4,9$ | $1,5$ | $0,2$ | $33$ | $2$ |
| | ***p*-values** | $0,78$ | | $10^{-7}$ | | $0,00$ | |
| iris | GA | $70,8$ | $25,3$ | $10,6$ | $2,7$ | $75,7$ | $19,3$ |
| | modGA | $79,7$ | $22,9$ | $15,7$ | $3,9$ | $69,3$ | $17,3$ |
| | ***p*-values** | $0,29$ | | $10^{-3}$ | | $0,27$ | |
| pageblocks | GA | $89,9$ | $1,1$ | $14,8$ | $1,7$ | $81,9$ | $6,7$ |
| | modGA | $90,3$ | $0,8$ | $76,4$ | $10,5$ | $89,2$ | $11,6$ |
| | ***p*-values** | $0,45$ | | $10^{-7}$ | | $0,03$ | |

from Table 2, because there we've taken the best configuration of an unique execution, not the average value of 40 execution.

Table 4 shows the average value and the standard-deviation for CCP, RT and NI. Also we've provided the Man-Whitney $p$-value test for our interest variables. The significance of the test is 5%.

For all the datasets except careval there was no difference between the GA and modGA concerning the average values of CCP as we can see in the last column of Table 4. For careval, with a $p$-value of $0,04$ the modGA achieved a higer mean CCP, however, both RT and NI also was higher than GA.

For the cancer dataset, both RT and NI was lower in modGA than GA. That means to obtain an equivalent CCP in our approach less time and number of iterations is needed for this dataset. Thus, for this dataset, we can say that modGA could be the right choice over traditional GA.

For the cmc and iris datasets, any of approaches can be used, depending of requirements. Traditional GA had a lower runtime but a greater number of iterations over our approach modGA, meaning that it spends less time in each iteration.

For the hepatitis dataset, a curious fact happened, the number of iterations of the GA had no variation among the 40 executions of the algorithm. That leaded to a $p$-value of $0,00$ in the Man-Whitney test comparing NI with the modGA. However, modGA had a lower RT according to the same test, meaning that it stays less in each iteration.

Finally, for the pageblocks dataset, the traditional GA achieved the same mean CCP with both less time and number of iterations. So, for this dataset, traditional GA is the best choice for a real world implementation.

## 5. Conclusion

In this research, we've tested two GA-based algorithms to find one parameter in the RBFNN - Radial Basis Function Neural Network, training in a classification problem. To evaluate both approaches we used six datasets from the UCI Machine Learning Dataset Repository, from Lichman [7].

Training a RBFNN basically consists of defining the center positions, the spread vector associated to each center and the weight matrix from the hidden layer to the output layer. The amount of neurons is most times defined empirically.

Concerning the best configuration found in each technique, there is no significative difference, the CCPs - Correct Classified Percentuals was very similar in all datasets. Even for the Contraceptive Method Choice (cmc) dataset there was no difference, both techniques was not capable of offering an acceptable CCP.

In the average comparison, we notice:

- for the cancer dataset there was no significative difference on mean CCP, but both mean RT and mean NI were lower in modGA approach;
- for the cmc dataset there was no significative difference on mean CCP, but the GA approach had a lower mean RT and the modGA achieved a lower mean NI;
- for the careval dataset there was a significative difference in CCP, modGA approach mean was higher than GA approach, that means a significative gain on classification capability;
- in hepatitis there was no significative difference on mean CCP, but the mean RT of modGA was lower than GA;

- for the iris dataset there was no significative difference on mean CCP, but the mean RT of GA was the lowest and mean NI were lower in modGA approach;
- for the pageblocks dataset there was no significative difference on mean CCP, but both mean RT and mean NI were lower in GA approach.

Based on that evidence, we conclude that the proposed adaptation can obtain equivalent or higher mean CCPs to traditional GA approach in all tested datasets. The number of iterations of modGA was lower in 50% of datasets, so we can say this hypothesis - adaptation can reduce the number of iterations - works on some cases and when don't work, at least the CCP is statistically equivalent.

## 6. Acknowledgements

The authors would like to thank the Araucária Foundation for Development and Research, Paraná, Brazil, for financing this research.

## References

[1] A. P. Engelbrecht, *Computational intelligence: an introduction*, John Wiley & Sons Ltd, Chichester (2007).

[2] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley Longman, New Jersey (1989).

[3] S. Haykin, *Redes neurais - princípios e práticas*, Bookman, São Paulo (2001).

[4] J. H. Holland, *Adaption in Natural and Artificial Systems*, MIT Press, Cambridge (1975).

[5] T. Kurban, E. Besdok, A comparison of rbf neural network training algorithms for inertial sensor based terrain classification *Sensors*, **9** (8) (2009), 6312-6329.

[6] N. Li, H. Ling, Study of an algorithm of ga-rbf neural network generalized predictive control for generating unit, In: *International Conference on Electric Information and Control Engineering (ICEICE)*, Wuhan, China (2011), 1723-1726.

[7] M. Lichman, UCI machine learning repository, 2013.

[8] Z. Michalewicz, T. D. Logan, S. Swaminathan, Evolutionary operators for continuous convex parameter spaces, In: *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, River Edge, New Jersey (1994), 84-97.

[9] J. F. Mota, P. H. Siqueira, L. V. Souza, A. Vitor, Training radial basis function netowrks by genetic algorithms, In: *4th International Conference on Angents and Artificial Intelligence*, Vilamoura, Portugal (2012).

[10] J. Sim, C. C. Wright, The kappa statistic in reliability studies: Use, interpretation, and sample size requirements, *Physical Therapy*, **85** (3) (2005), 257-268.

[11] Y. Zhangang, C. Yanbo, K. W. E. Cheng, Genetic algorithm-based rbf neural network load forecasting model, *Power Engineering Society General Meeting*, (2007), 1-6.

[12] S. Yu, K. Wang, Y. M. Wei, A hybrid self-adaptive particle swarm optimization-genetic algorithm-radial basis function model for annual electricity demand prediction, *Energy Conversion and Management*, **91** (2015), 176-185.

[13] Z. Zhao, X. Wu, C. Lu, H. Glotin, J. Gao, Optimizing widths with pso for center selection of gaussian radial basis function networks, *Science China Information Sciences*, **57** (5) (2014), 1-17.