Improving Performance of Multi-objective Genetic for Function Approximation through island specialisation

A. Guillén[1], I. Rojas[1], J. González[1], H. Pomares[1], L.J. Herrera[1], and B. Paechter[2]

1) Department of Computer Architecture and Computer Technology
Universidad de Granada. Spain
2) School of Computing
Napier University. Scotland

Abstract. Nature shows many examples where the specialisation of elements aimed to solve different problems is successful. There are explorer ants, worker bees, etc., where a group of individuals is assigned a specific task. This paper will extrapolate this philosophy, applying it to a multiobjective genetic algorithm. The problem to be solved is the design of Radial Basis Function Neural Networks (RBFNNs) that approximate a function. A non distributed multiobjective algorithm will be compared against several parallel approaches that emerge as a straight forward specialisation of the crossover and mutation operators in different islands. The experiments will show how, as in the real world, if the different island evolve specific aspects of the RBFNNs, the results are improved.

1 Introduction

The problem to be tackled consists in designing an RBFNN that approximate a set of given values. The use of this kind of neural networks is a common solution since they are able to approximate any function [4, 11]. Formally, a function approximation problem can be formulated as, given a set of observations $\{(\boldsymbol{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\boldsymbol{x}_k) \in \mathbb{R}$ and $\boldsymbol{x}_k \in \mathbb{R}^d$, it is desired to obtain a function $\mathcal{F}$ so $\sum_{k=1}^{n} ||y_k - \mathcal{F}(\boldsymbol{x}_k)||^2$ is minimum. The purpose of the design is to be able to obtain outputs from input vectors that were not specified in the original training data set.

An RBFNN $\mathcal{F}$ with d entries and one output has a set of parameters that have to be optimised:

$$\mathcal{F}(\boldsymbol{x}_k; C, R, \Omega) = \sum_{j=1}^{m} \phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j) \cdot \Omega_j \tag{1}$$

where m is the number of RBFs, $C = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_m\}$ is the set of RBF centers, $R = \{r_1, ..., r_m\}$ is the set of values for each RBF radius, $\Omega = \{\Omega_1, ..., \Omega_m\}$ is

the set of weights and $\phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j)$ represents an RBF. The activation function most commonly used for classification and regression problems is the Gaussian function because it is continuous, differentiable, it provides a softer output and it improves the interpolation capabilities [2, 12]. The procedure to design an RBFNN starts by setting the number of RBFs in the hidden layer, then the RBF centers $\boldsymbol{c}_j$ must be placed and a radius r_j has to be set for each of them. Finally, the weights Ω_j can be calculated optimally by solving a linear equation system [5].

We want to find both a network with the smallest number of neurons and one with the smallest error. This is a multiobjective optimisation problem. For some pairs of networks it is impossible to say which is better (one is better on one objective, one on the other) making the set of possible solutions partially sorted [10].

2 Multiobjective Algorithm for Function Approximation: MOFA

Within the many evolutionary algorithms that have been designed to solve multiobjective optimization problems, the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [3] has been shown to have an exceptional performance. This section will describe the adaptation of this genetic algorithm to solve the problem of designing RBFNN for function approximation.

2.1 Encoding RBFNN in the Individuals

As it was shown in Section 1, to design an RBFNN it is needed to specify: the number of RBFs, the position of the centers of the RBFs, the length of the radii, and the weights for the output layer.

The individuals in the population of the algorithm will contain those elements in a vector of real numbers, but not the weights. Instead of the weights, the approximation error is stored in the chromosome, the reason of this is to save computational effort when the individuals have to be compared.

2.2 Initial Population

The initial population is generated using clustering algorithms in order to supply good individuals that will make easier and faster to find good solutions. These clustering algorithms were designed specifically to provide a good start point when designing RBFNNs for functional problems. It also includes individuals generated randomly in order keep diversity in the population. The clustering algorithms employed for this task are:

- Fuzzy C-means (FCM): This clustering algorithm [1] performs a fuzzy partition of the input data where the same input vector can belong to several clusters at the same time with a membership degree.

– Improved Clustering for Function Approximation (ICFA): this algorithm [6] uses supervised clustering in order to identify the areas where the function is more variable. To do this, it defines the concept of estimated output of a center to assign a value for the center in the output axis.
– Possibilistic Centers Initialiser (PCI) and Fuzzy-Possibilistic Clustering for Function approximation (FPCFA): these algorithms [7] modify the way the input vectors are shared between the centers of the clusters. In the ICFA algorithm, a fuzzy partition was defined. In these two algorithms the fuzzy partition is replaced by the ones used in [13] and in [9] respectively.

After the initialization of half of the individuals of the population with the clustering techniques, a few iterations of a local search algorithm are applied to each one and the results are appended to the population. This procedure increments the diversity and, as experimentally has been proven, improves the quality of the results.

The size of the RBFNNs coded by each individual should be small in the initialization step for two reasons: 1) to make the initialization as fast as possible and 2) to allow the genetic algorithm to determine the sizes of the RBFNNs from an incremental point of view, saving the computational effort that would be required to deal with big networks from the first generations.

2.3 Crossover operators

Standard crossover operators cannot be applied to our representation. Two specific crossover operators have been designed considering complete RBFs as genes to be exchanged by the chromosomes representing RBFNNs.

Crossover operator 1: Neurons exchange This crossover operator, conceptually, would be the most similar one to a standard crossover because the individuals represent an RBFNN with several neurons and a crossover between two individuals would result in a neuron exchange. The operator will exchange only one neuron, selected randomly, between the two individuals. This crossover operator exploits the genetic material of each individual in a simple and efficient way without modifying the structure of the network.

Crossover operator 2: Addition of the neuron with the smallest local error This operator consists in the addition into one parent of the neuron with the smallest local error belonging to the other parent. The local error is defined as the sum of the errors between the real output and the output generated by the RBFNN. Not all the input vectors are considered, only the ones that activate each RBF. To know if an input vector activates a neuron, its activation function is calculated for each input vector and if it is higher than a determined threshold, the input vector activates the neuron.

This operator gives the opportunity to increase the number of RBFs in one individual although once the cross has been performed, a refinement step is done.

This refinement consists of prunning the RBFs which do not influence the output of the RBFNN. To do this, all the weights that connect the processing units to the output layer are calculated and the neurons that do not have a significant weight are removed. The calculation of the weights is performed optimally by solving a linear equation system [5].

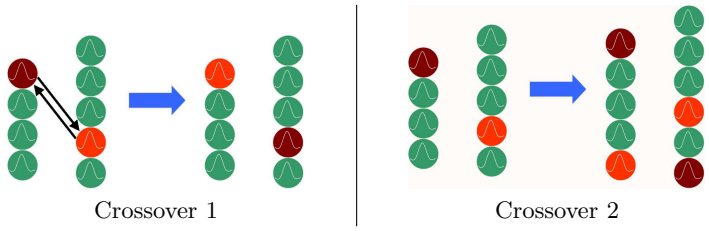


Fig. 1. Crossover operators 1 and 2

2.4 Mutation Operators

The mutation operators add randomness into the process of finding good solutions. These mutations allow the algorithm to explore the solution space avoiding being trapped in local minima. The modifications can be purely random or can be performed using problem specific knowledge. The mutation operators are desired to be as much simple as they can so it can be shown clearly, without the interference of introducing expert knowledge, the effects of the parallelization at a very basic level. For an RBFNN, two kind of modifications can be performed:

- Changes in the structure of the RBFNN (Increase and decrease operators): Addition and Deletion of an RBF. The first one adds an RBF in one random position over the input vectors space, setting its radius with a random value. All the random values are taken from an uniform distribution in the interval [0,1] since the input vectors and their output are normalised. The second operator is the opposite to the previous one, deleting a random existing RBF from the network. This mutation is constrained so that is not applied when the individual has less than two neurons.
- Changes in the elements of the structure (Movement operators): The third and the fourth operators refer to real coded genetic algorithms as presented in [8]. The third operator adds to all the coordinates of a center a random distance chosen in the interval [-0.5,0.5] from a uniform distribution. The fourth one has exactly the same behaviour but changing the value of the radius of the selected RBF.

3 Island Specialisation

In order to obtain results of maximum quality and take advantage of each type of operator, several parallel implementations were studied. All these implemen-

tations are based on the island paradigm where there are several islands (in our parallel implementation these map to processors) that evolve independent populations and, exchange information or individuals.

The design of an RBFNN, involves two well defined tasks: the design of the structure and the determination of the value of the parameters that build that structure. From this point of view many island specialisations were analysed although in this paper only the more representative ones are commented:

1) *Division of the crossover operators (P1).* This approach allows specialisation of the first stage of the evolution process, that is, the reproduction process. The two crossover operators are separated and executed in different islands. The mutation process remains the same for both types of islands. The following algorithms study the possible ways of specialising the mutation stage but always using the specialised crossover topology. As it will be shown in the experiments, allowing specialisation of the crossover operators gives significant improvements.

2) *Crossover 1 + Movement + Decreasing and Crossover 2 + Movement + Increasing (P2).* This combination of operators aims to boost the exploration capabilities of the crossover 2 and the exploitation ones of crossover 1. Since the crossover 2 explore more topologies by increasing the size of the NNs, the specialisation can be done by letting this island to produce only bigger networks. The other island with the crossover 1, will take care of exploitation of the chromosomes and will decrease the size of the networks.

3) *Crossover 1 + Movement + Increasing and Crossover 2 + Movement + Decreasing (P3).* This algorithm is the opposite to the previous one. The aim is to not allow the crossover 2 to create too big NNs by adding the operator that removes an RBF. If the NNs in the population become too big, the computational time would be highly increased. To be able to explore more different topologies, the island with the crossover 1 will increase the number of RBFs using the increasing mutation operator.

4) *Crossover 1 + Movement and Crossover 2 + Increasing and Decreasing (P4).* This approach is the one that specialises the most each island on each task of the design of an RBFNN. The island with the crossover 1 will take care of changing the parameters of the centers and the radii and at the same time, will exploit the genetic information contained in one topology through its crossover operator. The second island will just take care of the modifications on the structure of the RBFNN by increasing or decreasing its size through both the crossover and mutation operators.

4 Experiments

This section analyses the behaviour of the algorithms described above to show that specialisation of islands to use different operators can lead to better results. The experiments were performed using a two dimensional function (Figure 2) that was generated using a gaussian RBFNN ($e^{-\frac{||\boldsymbol{x}_k - \boldsymbol{c}_i||^2}{r_i^2}}$) over a grid of 25x25 points using the randomly chosen parameters in Table 1.

RBF centers		radii	weights
0.7181	0.8162	0.2237	0.4996
0.5692	0.9771	0.0715	2.4712
0.4608	0.2219	0.0628	0.4887
0.4453	0.7037	0.2332	-0.7052
0.0877	0.5221	0.0327	-0.2006
0.4435	0.9329	0.2352	-0.8020
0.3663	0.7134	0.1755	1.2668
0.3025	0.2280	0.2119	-0.5123
0.8518	0.4496	0.0523	1.0884
0.7595	0.1722	0.1138	-0.5318
0.9498	0.9688	0.0203	2.0797
0.5579	0.3557	0.2237	0.5002
0.0142	0.0490	0.0715	2.2935
0.5962	0.7553	0.0628	-0.8135

Table 1. Parameters for the function f_2

As it can be seen in Figure 2, the target function presents a fairly irregular topology, making it quite difficult to be approximated.

The algorithms were executed using the same initial population of the same size and the same values for all the parameters. The crossover probability and mutation probability were 0.5, the size of the population 100 and the number of generations was fixed to 100. The probabilities might seem high but it is necessary to have a high mutation probability to allow the network to increase its size, and the centers and radii to move. The high crossover probability makes it possible to show the effects of the crossover operators. Several executions were made changing the migration rate, allowing the algorithms to have 2 (each 40 generations), 4 (each 20 generations), 9 (each 10 generations), and 19 (each 5 generations) migrations through the 100 generations.

The results are shown in Figures 3, and 4 where the Pareto fronts obtained from each algorithm are represented according to the objectives that have to be minimised, that is, the approximation error and the number of neurons in the network.

The first experiment shows how specialisation of the crossover operators over two machines can improve the performance. The the other experiments show that as the specialisation increases, the results continue to improve.

4.1 Parallel Approach P1 vs. sequential approaches

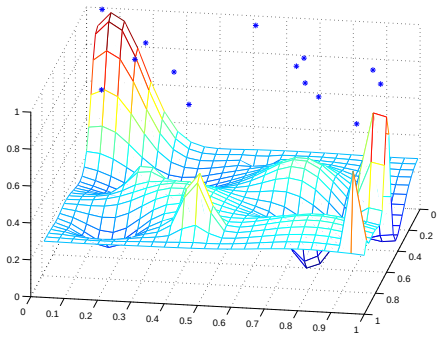
The three different sequential algorithms are determined by the crossover operators and all of these algorithms use all the four mutation operators at the same time. The first sequential algorithm uses only the crossover 1, the second algorithm uses the crossover 2 and the third non distributed algorithm chooses randomly between the two crossover operators each time it has to be applied.

For the sequential algorithms, as it is shown in Figure 3, the crossover operator 1 obtains better results for small networks but it is not able to grow them as

much as the crossover operator 2 does. When the two operators are applied together the results improve significantly, obtaining a better Pareto front although the crossover operator 2 is able to design networks with a smaller approximation error when the size of them is big.

Figure 3 shows that the parallel model outperforms the three previous possibilities, although its important to notice that it is dependent on the migration rate. If the migration rate is high (every 5 generations) the behaviour is almost the same as with the crossover operators combined in one algorithm. When the migration rate becomes too low (every 40 generations) the performance is decreased in comparison with the mixed approach although it always obtains better results than any of the other possibilities. So the migration rate should be high enough to share the individuals but allowing each operator to use its properties to evolve its individuals in the best way.

Fig. 2. Target function f_2 and original position of the centers of the RBFs



4.2 Comparison Between Parallel Approaches (P1, P2, P3, and P4)

Having shown that the parallel approach could lead to better solutions, the different parallel algorithms were compared. Figure 4 shows the results of the four parallel algorithms studied using different migration rates.

The results show that parallel implementation P3 has poor performance in comparison with the other approaches. This implementation is the less specialised from the other ones because the mutation operators perform the opposite task of the crossover operators. The second type of island (crossover 2 + decreasing + movement) is able to reduce the number of RBFs, annulling the exploration effect of the crossover 2 that increases the size of the RBFNNs. The first type of island (crossover 1 + increasing + movement) increases the size of the network not being able to exploit the genetic recombination by its crossover operator. The objective of studying this implementation was to show if a smoothed specialization could improved the results.

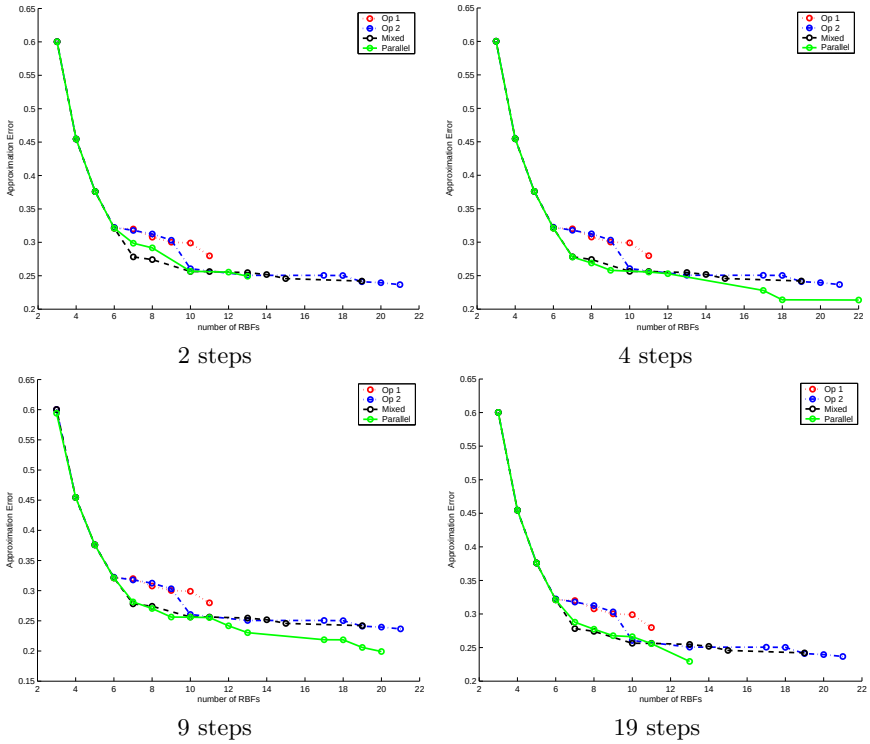


Fig. 3. Pareto front obtained performing several migration steps for P1 and the non distributed approaches

Algorithms P2 and P4 could be considered the most specialised because their mutation operators are oriented to only one task (P4) or to complement the features of the crossover operators on each island, unlike P3 where the mutation operators counteract the effects of the crossovers. For P2, the best results are obtained for few migrations while P4 achieves the best results with the maximum number of migrations. If the best results for both algorithms are compared, P2 obtains slightly better results for smaller networks and P4 obtains better results for larger networks and it finds more elements in the Pareto. The behaviour of these two algorithms is logical, P2 is able to exploit more the solutions since it reduces the size of the networks and recombines the genes in a more exhaustive way, so it achieves the best results when there are not many migration steps. P4 explores more topologies than P2, that is why it obtains a more complete Pareto front. The reason why it gets the best results with the highest migration rate is because the exploration island must have fast feedback about the exploitation of the individuals in its Pareto set, although there is a limit; when if the migration rate becomes too frequent, the islands do not have the chance to exploit and explore, decreasing the quality of the results.

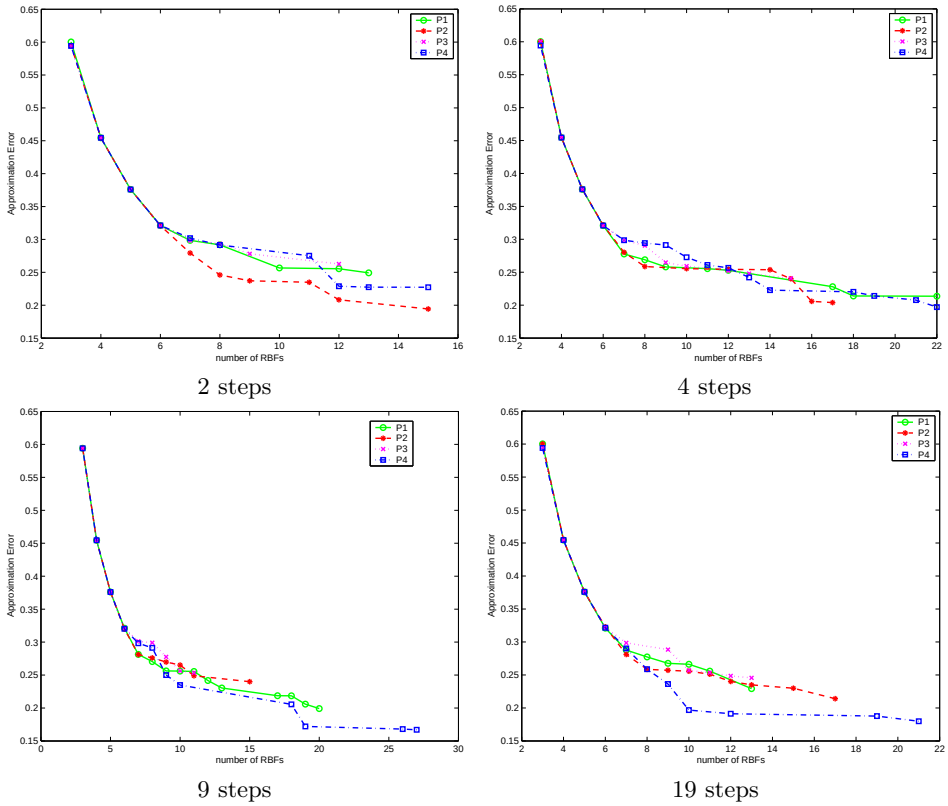


Fig. 4. Pareto front obtained performing several migration steps for P1, P2, P3 and P4

5 Conclusions

The design of an RBFNN is a complex task that can be solved using evolutionary approaches that provide satisfactory results. This paper has presented a multi-objective GA that designs RBFNNs to approximate functions. This algorithm has been analysed considering sequential approaches and approaches that have been specialised by defining separate islands that apply different combinations of mutation and crossover operators. The results confirm that the specialisation on the different aspects of the design of RBFNNs through the parallel approaches leads to better results.

Acknowledgements

This work has been partially supported by the Spanish CICYT Project TIN2004-01419 and the HPC-Europa programme, funded under the European Commis-

sion's Research Infrastructures activity of the Structuring the European Research Area programme, contract number RII3-CT-2003-506079.

References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.
2. A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. *OnLine Symposium for Electronics Engineers*, 1:1–7, February 2001.
3. Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.
4. A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transanctions on Information Theory*, 25(4):373–380, July 1979.
5. J. González, I. Rojas, J. Ortega, H. Pomares, F.J. Fernández, and A. Díaz. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478–1495, November 2003.
6. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. *Lecture Notes in Computer Science*, 3512:272–280, June 2005.
7. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. A possibilistic approach to rbfn centers initialization. *Lecture Notes in Computer Science*, 3642:174–183, 2005.
8. F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behavioural analysis . *Artificial Intelligence Reviews*, 12(4):265–319, 1998.
9. N. R. Pal, K. Pal, and J. C. Bezdek. A Mixed C–Means Clustering Model. In *Proceedings of the 6th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, volume 1, pages 11–21, Barcelona, July 1997.
10. V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.
11. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
12. I. Rojas, M. Anguita, A. Prieto, and O. Valenzuela. Analysis of the operators involved in the definition of the implication functions and in the fuzzy inference proccess. *International Journal of Approximate Reasoning*, 19:367–389, 1998.
13. J. Zhang and Y. Leung. Improved possibilistic C–means clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 12:209–217, 2004.