Improving the Performance of Multi-objective Genetic Algorithm for Function Approximation Through Parallel Islands 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 a parallel approach that emerges 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 islands evolve specific aspects of the RBFNNs, the results are improved.

1 Introduction

The problem to be tackled consists in designing an Radial Basis Function Neural Network (RBFNN) that approximates 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 [8]. 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 $\mathcal{G}$ so $\sum_{k=1}^{n} ||y_k - \mathcal{G}(\boldsymbol{x}_k)||^2$ 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 optimized:

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

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 [1]. 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 [3].

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 [7].

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) [2] 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 the weights. The storage of the weights in the chromosome is useless since they can be obtained optimally by solving a linear equation system, and for each change in a radius or a center they have to be updated.

2.2 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 of a neuron is defined as the sum of the errors between the real output and the output generated by the RBFNN for the input vectors that activate that neuron.

To make sure that the offsprings are different of their ancestors, before adding the neuron with the smallest local error, the crossover will make sure that the neuron to be added does not exist in the individual, otherwise the second neuron with the smallest local error will be considered and so on. This restriction combined with the way the NSGA-II proceeds helps to avoid the "competing convention"problem [4].

Once the offspring are obtained, they go through a refinement process that consists in the prune of the RBFs which doesn't 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 don't have a significant weight will be removed.

2.3 Mutation Operators

The mutation operators add randomness into the process of finding good solutions. 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 normalized. 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 [5]. 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 behavior 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, a parallel implementation was studied. This implementation is 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.

From the point of view of evolving the topology and structure of the RBFNN, we propose the following specialization: combine crossover 1 and the movement mutation operators in one island and the crossover 2 and the movement and increasing mutation operators. 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.

4 Experiments

The experiments were performed using a two dimensional function that was generated using a Gaussian RBFNN with randomly chosen parameters over a grid of 25x25 points.

Although some metrics have been applied to compare nondominated sets [6], in [9] it is suggested that: *"as long as a derived/known Pareto front can be visualized, pMOEA*[1] *effectiveness may still best be initially observed and estimated by the human eye"* . Therefore the experimental analysis of the algorithms will be based on the plots of the resulting Pareto fronts.

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's 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 Figure 1 where the Pareto fronts obtained from each algorithm are represented according to the objectives that have to be minimized, this is, the approximation error and the number of neurons in the network.

The parallel implementation will be compared with non distributed algorithms that are considered as those that have no communication. Within the non distributed algorithms, three different possibilities are determined by the crossover operators and all of these algorithms will use the four mutation operators at the same time. The first 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.

[1] *pMOEA* stands for parallel MultiObjective Evolutionary Algorithms

For the non distributed algorithms, as it is shown in Figure 1, 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 1 shows that the parallel model outperforms the three previous possibilities for all the migration rates. This means that the parallelization is positive independently of the migration rate. If the results are analyzed in detail, it is possible to see how, for low migration rates, the exploitation capabilities of the algorithm increases obtaining smaller networks with better approximation errors meanwhile for high migration rates, the exploration capabilities are boosted, obtaining a more complete Pareto.

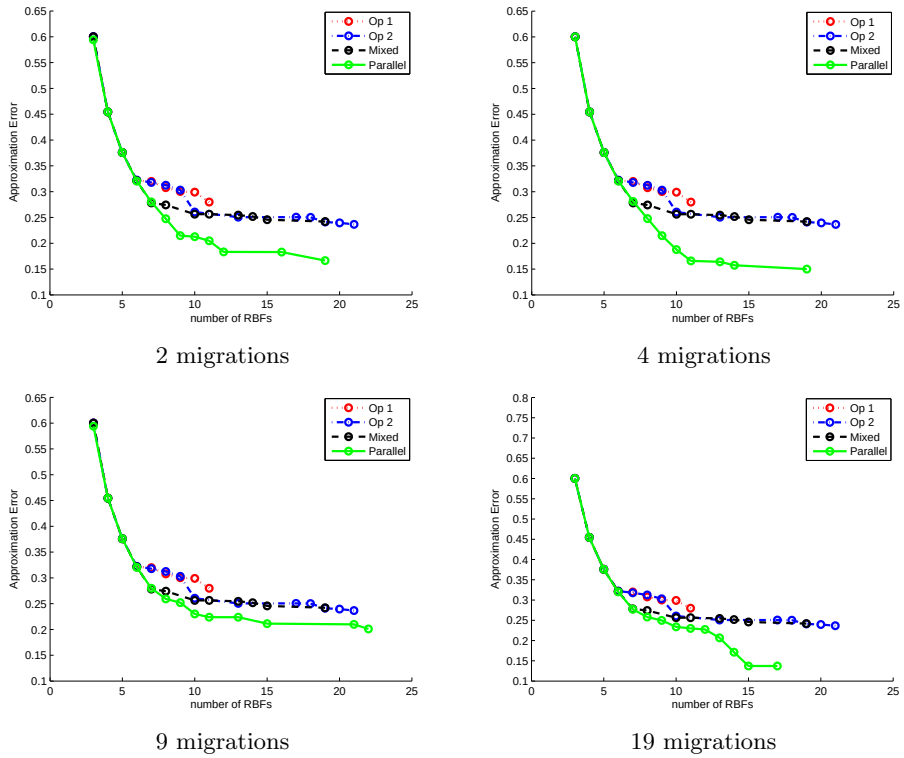

Fig. 1. Pareto front obtained performing several migration steps for the parallel algorithm and the non distributed approaches

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 multiobjective GA that designs RBFNNs to approximate functions. This algorithm has been analyzed considering a non distributed approach that has been specialised by defining separate islands that applied 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 approach could lead to obtain 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 Commission's Research Infrastructures activity of the Structuring the European Research Area programme, contract number RII3-CT-2003-506079.

References

1. A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. *OnLine Symposium for Electronics Engineers*, 1:1–7, February 2001.
2. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.
3. 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.
4. P. J. B. Hancock. Genetic Algorithms and Permutation Problems: a Comparison of Recombination Operators for Neural Net Structure Specification. In D. Whitley, editor, *Proceedings of COGANN workshop, IJCNN, Baltimore*. IEEE, 1992.
5. 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.
6. J. Knowles and D. Corne. On metrics for comparing non-dominated sets, 2002. In Congress on Evolutionary Computation (CEC 2002).
7. V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.
8. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
9. D. A. van Veldhuizen, J. B. Zydallis, and G. B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 7(2):144–173, 2003.