Voronoi Diagrams Based Function Identification

Carlos Kavka[1] and Marc Schoenauer[2]

[1] LIDIC, Departamento de Informática, Universidad Nacional de San Luis
D5700HHW, San Luis, Argentina
ckavka@unsl.edu.ar

[2] Projet Fractales, INRIA Rocquencourt
BP 105, 78153 Le Chesnay Cedex, France
Marc.Schoenauer@inria.fr

Abstract. Evolutionary algorithms have been applied to function identification problems with great success. This paper presents an approach in which the individuals represent a partition of the input space in Voronoi regions together with a set of local functions associated to each one of these regions. In this way, the solution corresponds to a combination of local functions over a spatial structure topologically represented by a Voronoi diagram. Experiments show that the evolutionary algorithm can successfully evolve both the partition of the input space and the parameters of the local functions in simple problems.

1 Introduction

The objective when dealing with a function identification problem is to find an approximation that matches as closely as possible an unknown function defined in a certain domain. There are methods that can be used to optimize the parameters of the unknown function given a model, and also methods than can get both the model and the parameters at the same time. A well known example is the least squares method (LSM), than can compute the coefficients of a linear combination of base functions. Also non linear regression methods do exist, but they tend to be very time consuming when compared with the linear approaches. Neural networks have also been used for function approximation [7]. A neural network can be considered as a model of connected units, where usually the unknown parameters are the so called weights. The usual training algorithms can be used to obtain the weights, but there are also methods that can obtain both the connection pattern and the weights.

Evolutionary algorithms have been shown very effective in function identification problems in a wide range of domains [3] [1]. The Genetic Programming approach [8] uses a tree structure to represent an executable object (model) that can be a function – and has been successful addressing regression problems. Classifier systems have also been used for function approximation. In the XCFS classifier system [11], the value of the dependent variable is considered as a payoff to be learned, given the values of the independent variables.

But the choice of a method to solve a given function identification problem also depends on the available data: when *examples* of input/output patterns of the unknown function are given, function identification, also then called *data fitting*, can be addressed by standard deterministic numerical methods. This is not the case, however, for the so-called *inverse problems*: In control for instance, candidate controllers cannot be specified just by a set of samples of values of the independent and dependent variables; but a good simulation of the plant is usually available, and allows one to evaluate candidate controllers. Even worse, it is usual that some regions of the space are oversampled, while others are undersampled.

This work is part of a series of investigations regarding the usefulness of a particular representation and operators defined to solve the inverse approximation problem with evolutionary algorithms. The objective is to find both a partition of the domain into a set of regions and a set of functions to be applied in each region. In this way, instead of the solution being defined by a single *global* function, it is defined by a set of *local* functions. The partition of the domain is defined in terms of Voronoi regions, a geometric structure that proved to be useful in interpolation [4] [2] and evolution applied to structural mechanics problems [10]. No restrictions are defined for the local functions, but they are expected to be simple functions. Two possible rules for their combination are proposed in order to get a solution in the whole domain.

The paper is organized as follows: section 2 presents the details on the domain partition based on Voronoi regions, section 3 presents details on local approximators and the way in which they are combined, section 4 introduces the representation and the evolutionary operators, section 5 some numerical results on simple problems and section 6 presents the conclusions and the current lines of research.

2 Domain Partition

The domain partition strategy is based on Voronoi diagrams. A Voronoi diagram induces a subdivision of the space based on a set of points called *sites*. An important property is that a number of operations can be executed on its topological structure just by operating with the sites.

Formally [5], a Voronoi diagram of a set of n points P is the subdivision of the plane into n cells, one for each site in P, with the property that a point q lies in the cell corresponding to a site p_i if and only if the distance between q and p_i is smaller than the distance between q and p_j for each $p_j \in P$ with $j \neq i$. In other words, the cell of the site p_i contains all the points in the plane for which p_i is the closest site. The Voronoi diagram of P will be denoted by $\mathrm{Vor}(P)$ and the cell (or region) that corresponds to p_i by $\mathrm{Vor}(p_i)$. Figure 1 illustrates an example of a Voronoi diagram in $\mathbb{R}^2$. The definition can be straightforward extended to $\mathbb{R}^n$, with $n \geq 2$.

A related concept that will be used in the paper is the so called Delaunay triangulation. A triangulation [5] of a set of points P is defined as the maximal planar subdivision whose vertex set is P. A maximal planar subdivision S is a subdivision such that no edge connecting two vertices can be added to S without destroying its planarity. In other words, any edge that is not in S intersects one of the existing edges. A triangulation $\mathcal{T}$ of a set of points P is a Delaunay triangulation if and only if the circumcicle of any triangle in $\mathcal{T}$ does not contain a point of P in its interior. A circumcicle of a triangle is defined as the circle that goes through its three summits. Figure 1 illustrates an example of a Delaunay triangulation in $\mathbb{R}^2$.

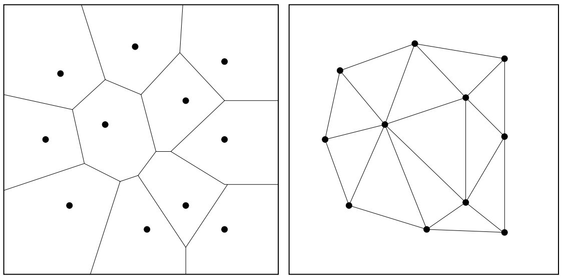

Fig. 1. An example of a Voronoi diagram (left) and a Delaunay triangulation (right) for a set of points in $\mathbb{R}^2$

3 Approximation

A solution to the function identification problem is defined by a partition of the domain in Voronoi regions together with a set of local functions. Two ways of recombining local functions are proposed: a non continuous combination, where each function defines the approximation in its own region, and a continuous combination, where the local functions are adapted in order to get a continuous approximation in the complete domain.

3.1 The Non-continuous Combination

Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$ and a set of continuous local functions $F = \{f_1, f_2, \ldots, f_n\}$, each f_i being defined on $Vor(p_i)$, the value of the approximation $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(x) = f_i(x) \quad \text{if} \quad x \in \text{Vor}(p_i) \ . \tag{1}$$

The value of the approximation $\mathcal{F}$ is computed as the value of the local function f_i associated to the Voronoi region to which the input value x belongs. The definition of $\mathcal{F}$ presents the following properties:

- It is defined on every point in the domain, provided that the local functions are defined, since the Voronoi diagram induces a partition of the complete domain.
- It is a (possibly) discontinuous function, presenting discontinuities at the Voronoi region boundaries.

3.2 The Continuous Combination

Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$, a set of continuous local functions $F = \{f_1, f_2, \ldots, f_n\}$ and a set of real values $V = \{v_1, v_2, \ldots, v_n\}$, the value of the approximation $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(x) = f_i(x) * \mathcal{D}(x) + \mathcal{L}(x) \quad \text{if} \quad x \in \text{Vor}(p_i) \; . \tag{2}$$

The value of the approximation $\mathcal{F}$ is computed as the value of the local function f_i associated to the Voronoi region to which the input value x belongs to, scaled by a distance factor $\mathcal{D}$, with the addition of the value of a global function $\mathcal{L}$ evaluated on x.

The distance factor $\mathcal{D}$ is defined as follows:

$$\mathcal{D}(x) = \begin{cases} \phi\left(\frac{d(x, boundary(x))}{d(p_i, boundary(x))}\right) & \text{if } boundary(\text{x}) \text{ is defined} \\ 1 & \text{if } boundary(\text{x}) \text{ is not defined} \end{cases} \tag{3}$$

where p_i is the center of the Voronoi region to which x belongs to, $boundary(x)$ the point that corresponds to the Voronoi regions boundary, $d(a, b)$ the Euclidean distance between the points a and b, and ϕ a continuous function defined in the range [0..1] such that $\phi(0) = 0$ and $\phi(1) = 1$. $boundary(x)$ is defined as the intersection of the line joining p_i and x and the boundaries of $Vor(p_i)$, if that intersection lies within the domain of the definition of the problem.

The global function $\mathcal{L}$ is defined based on the Delaunay triangulation as follows:

$$\mathcal{L}(x) = \sum_{j=1}^{d+1} v_j * l_j \; . \tag{4}$$

where d is the dimension of the input space, $l_1, \ldots, l_{d+1}$ are the barycentric coordinate of x in simplex T (triangle the case if $d = 2$), and $v_1, \ldots, v_{d+1}$ are given values, each one associated with the corresponding Voronoi site (see section 4.1). The definition of $\mathcal{F}$ presents the following properties:

– It is defined on every point in the domain, provided that the local functions are defined, since the Voronoi diagram induces a partition for the complete domain, and $\mathcal{D}$ and $\mathcal{L}$ are continuous functions.
– The value of the first term is the local approximator in the center of the region and 0 in the boundaries, since the distance factor $\mathcal{D}$ produces a value in the range [0..1], with the value 0 in the boundaries of the Voronoi region and the value 1 in the center. The function ϕ controls the shape of the change of $\mathcal{D}$ between 0 and 1. It can be linear, quadratic, exponential, ... Ultimately, different functions could be attached to different Voronoi sites. In the external area of unbounded Voronoi regions the value of $\mathcal{D}$ is 1.
– The function $\mathcal{L}$ is global since it depends on the values v_i associated to the sites and their positions in the space. It is a piecewise linear continuous function. The vertices of the triangles defined by the Delaunay triangulation are the sites from P. Given a point x that belongs to the triangle T in the

Delaunay triangulation, the value of the global function $\mathcal{L}$ is computed by performing an interpolation of the values associated to the vertices by using the barycentric coordinates of the point x in the triangle. The barycentric coordinates of a point x are the local coordinates of x in T representing the ratios of the areas of the sub-triangles formed by x and the sides of the triangle. It corresponds to what is called a triangular element in the Finite Elements Method terminology [12]. As an example, there are three barycentric coordinates for a point in $\mathbb{R}^2$: l_1, l_2, l_3, with $\sum l_i = 1$. The value of one of each of the barycentric coordinates is 1 in a vertex while the value of the others barycentric coordinates is 0. Figure 2 presents the value of the three barycentric coordinates for a triangle in $\mathbb{R}^2$, together with an example of the global function $\mathcal{L}$ evaluated on the same triangle. Note that even if

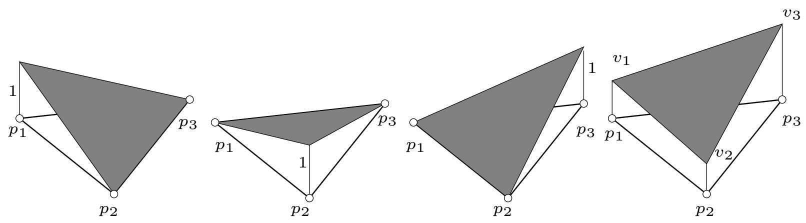

Fig. 2. Barycentric coordinates in a triangle in $\mathbb{R}^2$ (from left to right): barycentric coordinate l_1, barycentric coordinate l_2, barycentric coordinate l_3, the global function $\mathcal{L}$ in the triangle T

it is true that every point in the domain belongs to a Voronoi region, it is not the case with the Delaunay triangulation. Some points that belong to unbounded Voronoi regions do not belong to any triangle defined by the Delaunay triangulation (see figure 1). In order to define $\mathcal{L}$ on these points, a large triangle covering the whole domain and containing all points in P is defined with values 0 associated to its vertices [5].

4 The Evolutionary Algorithm

This section introduces the representation used for the individuals and the definition of the evolutionary operators.

4.1 The Representation

Each individual has to represent a complete solution, or in other words, a complete approximation $\mathcal{F}$. A convenient representation is a list of variable length of Voronoi sites, local approximator parameters and global function values (if used), represented as real values.

In order to formalize the definition, let us define a *local vector* as the vector that contains all parameters associated to a Voronoi region: the coordinates that define the site, the parameters of the local approximator and the value of the global function (if applicable). A local vector lv_i of an individual *ind* with non continuous approximation is defined as the vector:

$$lv_i = [c_1, \ldots, c_d, par_1, par_2, \ldots, par_m]$$

or as the following vector with the continuous approach:

$$lv_i = [c_1, \ldots, c_d, v, par_1, par_2, \ldots, par_m]$$

where the site coordinates are $(c_1, \ldots, c_d)$, d the dimension of the domain, v the value of the global function on the site and $par_1, \ldots, par_m$ the m parameters that defines the local approximator.

An individual *ind* can be represented as a vector of variable length that is built from the concatenation of the local vectors that represent the local approximators in the solution:

$$\text{ind} = lv_1 + lv_2 + \ldots + lv_n$$

As an example, an individual in $\mathbb{R}$ with 3 local approximators defined with two parameters (a and b) each one is:

with non continuous approximation: $[\underbrace{p_1, a_1, b_1}_{lv_1}, \underbrace{p_2, a_2, b_2}_{lv_2}, \underbrace{p_3, a_3, b_3}_{lv_3}]$

with continuous approximation: $[\underbrace{p_1, v_1, a_1, b_1}_{lv_1}, \underbrace{p_2, v_2, a_2, b_2}_{lv_2}, \underbrace{p_3, v_3, a_3, b_3}_{lv_3}]$

The representation proposed does not impose restrictions on the shape of the local approximators. In this work, we have used linear, quadratic and RBF approximators. They were defined in $\mathbb{R}^2$ respectively as $f(x) = ax+b$, $f(x) = ax^2+bx+c$ and $f(x) = \exp\left(-(x-p)^2/a^2\right)b + c$, where a, b and c are the parameters, and p the center of the corresponding Voronoi region.

4.2 The Operators

Three mutation operators and one crossover operator have been specifically defined for individuals that represent a Voronoi diagram. They are described below:

Voronoi crossover: This crossover operator is based on the crossover defined in [10] and exchanges the local vectors of the individuals by using geometric properties of the Voronoi diagrams. A random hyperplane h of dimension $d-1$ is randomly defined. The first child receives the local vectors from the first parent that lie on the left of h, and the local vectors from the second parent that lie on the right of h. The second child receives the remaining local vectors. Figure 3 presents an example of the application of this operator

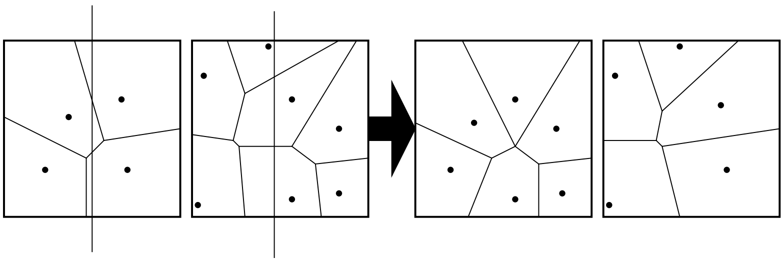

Fig. 3. An example of the application of the Voronoi crossover in $\mathbb{R}^2$

in $\mathbb{R}^2$. Formally, given two individuals ind^1 and ind^2 with n and m local vectors respectively:

$$ind^1 = [lv_1^1, ..., lv_n^1] \qquad ind^2 = [lv_1^2, ..., lv_m^2]$$

The two children $child^1$ and $child^2$ are:

$$child^1 = \{lv_i^1, lv_j^2 / lv_i^1 \in Left(h), lv_j^2 \in Right(h)\}$$

$$child^2 = \{lv_i^1, lv_j^2 / lv_i^1 \in Right(h), lv_j^2 \in Left(h)\}$$

where h is a random hyperplane of dimension $d-1$, a local vector belongs to $Left(h)$ if its site is on the left of h and a local vector belongs to $Right(h)$ if its site is on the right of h. The concept of being on the right or on the left of an hyperplane can be found in computational geometry textbooks, like for example [5].

Mutation: This operator modifies the coordinates of the Voronoi site, the parameters of the approximator or the global function value (if used) by updating the values as follows:

$$x_i(t) = \begin{cases} x_i(t) + \Delta(t, ub_j - x_i(t)) & u < 0.5 \\ x_i(t) - \Delta(t, x_i(t) - lb_i) & u \geq 0.5 \end{cases} \tag{5}$$

where $x_i(t)$ is the i-th parameter at time t, u is a uniformly generated random number in $[0:1]$, lb_i and ub_i are respectively the lower bound and the upper bound for the parameter x_i, and Δ is a function defined as follows:

$$\Delta(t, y) = yv(1 - t/T)^b \; . \tag{6}$$

where $v \in [0:1]$ is a random number obtained from a gaussian distribution with mean 0 and standard deviation 0.3, T the maximum number of generations and b a parameter that controls the degree of nonuniformity. This operator is used to search the space uniformly in the first generations and very locally at the end, in order to fine tune the values in the individual. It is based on the nonuniform mutation operator defined by Michalewicz [9].

Add mutation: This mutation operator is used to add a new random local vector to the individual.

Del mutation: This mutation operator is used to remove a local vector from an individual.

5 Numerical Results

The objective of this section is to evaluate the representation and the operators described in the previous sections, when applied to simple function identification problems.

The first experiment consists in the approximation of the following function in $\mathbb{R}$, with linear, quadratic and RBF approximators, using both the non continuous and the continuous approach:

$$f(x) = \begin{cases} \frac{\sin(3\pi x)^2}{x} & -1 \leq x < 0 \\ \\ \frac{1}{x+\exp(-x)} - 1 & 0 \leq x \leq 1 \end{cases} \tag{7}$$

The algorithm implemented is a classical generational genetic algorithm with population size 100 and tournament selection (100 parents give birth to 100 offspring that replace the 100 parents). For the experimental test, the crossover rate set to 0.7, mutation rate to 0.1, mutation b parameter to 0.2, adding mutation rate equal to 0.1, delete mutation rate equal to 0.1, the maximum number of function evaluations to 50 000. Individuals are evaluated by computing the error on the approximation of a set of points, selected randomly for each individual on each evaluation. It means that the individuals are not evaluated with the same dataset – as would happen in inverse problems. The function ϕ (see equation (3)) is the identity. The results are summarized in table 1, and corresponds to 10 independent runs for each set of parameters. The errors are specified as a percentage of the output range, divided by 100. The error is computed by evaluating the approximation obtained in 1000 points evenly selected from the whole domain, and comparing its value with the real value obtained from the target function. As a consequence of this fact, the fitness values are not the same as the errors presented in this table. The figure 4 presents examples of the kind of solutions obtained with the different kind of approximators. Each plot shows the function to be approximated and the approximation obtained. The boxes in the diagram correspond to the Voronoi regions in which the domain is partitioned.

The solutions obtained by all methods are comparable in quality. Better solutions can be found by increasing the limit on the number of function evaluations. The main difference is on the size of the approximators obtained. A continuous combination can produce solutions with less local approximators, maintaining the same quality in the approximation. The second experiment concerns the crossover operator. Since the representation allows the use of standard one point crossover, the objective of the second experiment is to compare the Voronoi crossover with the one point crossover. The one point crossover tends to generate very long individuals, or in other words, individuals with a big number of Voronoi regions. It was not possible to use just the one point crossover, without adding a regularization term [7] in order to penalize long individuals. The results are summarized in table 2, and corresponds to 10 independent runs for each set of parameters. The value of the regularization parameter corresponds to the factor used to weight the size of the individual against the error in order

Table 1. Solutions found by the evolutionary algorithm for the first experiment. The smallest (resp. largest) size values correspond to the size of the smallest (resp. largest) best individuals found in the set of runs

approximator	continuity	best error	std. dev.	smallest	largest
linear	no	0.017882	0.00562	12	23
quadratic	no	0.017015	0.00693	12	29
RBF	no	0.013958	0.00561	11	53
linear	yes	0.018051	0.00577	6	11
quadratic	yes	0.011043	0.01041	5	13
RBF	yes	0.011468	0.00353	4	16

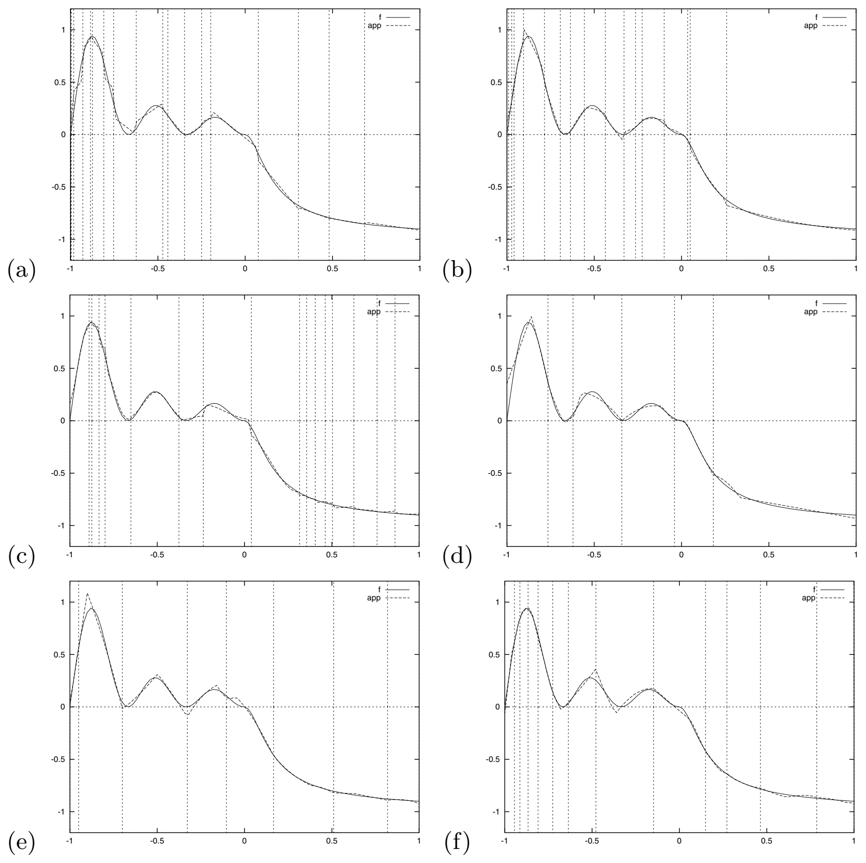

Fig. 4. Example of approximations obtained for the first function by using (a) linear, (b) quadratic and (c) RBF local approximators with no continuity, and using (d) linear, (e) quadratic and (f) RBF local approximators with continuity

Table 2. Solutions found for the second experiment

approximator	continuity	crossover	regularization	best error	smallest	largest
linear	yes	voronoi	no	0.018051	6	11
linear	yes	one point	no	-	-	-
linear	yes	one point	0.995	0.024664	23	41
linear	yes	one point	0.990	0.030236	11	29

to compute the fitness. The third experiment is intended to analyze the shape of the Voronoi partitions obtained in the approximation of the following functions in $\mathbb{R}^2$, by using linear approximators and continuous combination:

$$f(x,y) = x \exp(-x^2 - y^2) \qquad f(x,y) = \sin(\sqrt{x^2 + y^2}) \; . \tag{8}$$

The same parameters of the first experiment are used. The results are summarized in table 3, and correspond to 5 independent runs. The figure 5 presents the plot of the target functions, examples of the approximations obtained by the algorithm and their corresponding Voronoi partitions.

Table 3. Solutions found by the evolutionary for the third experiment

approximator	continuity	crossover	regularization	best error	std.dev.
linear	yes	voronoi	no	0.021192	0.01092
linear	yes	voronoi	no	0.028055	0.02411

6 Conclusions

This work presents preliminary results on the use of evolutionary computation and Voronoi diagrams in function identification. The proposed representation and the operators allow the algorithm to evolve both the partition of the input space and the local functions to be applied in each region. The partition of the input space is performed through Voronoi diagrams. Two ways to combine the local approximators are proposed. The representation of the partition does not depend on the type of local approximators used. In this work, linear, quadratic and RBF local approximators were used, but other kind of approximators can also be used.

Experiments on simple problems show that good approximations can be obtained with linear, quadratic and RBF local approximators. They also show that continuity in the combination of local functions can help to obtain approximations with smaller number of local approximators. The Voronoi crossover showed

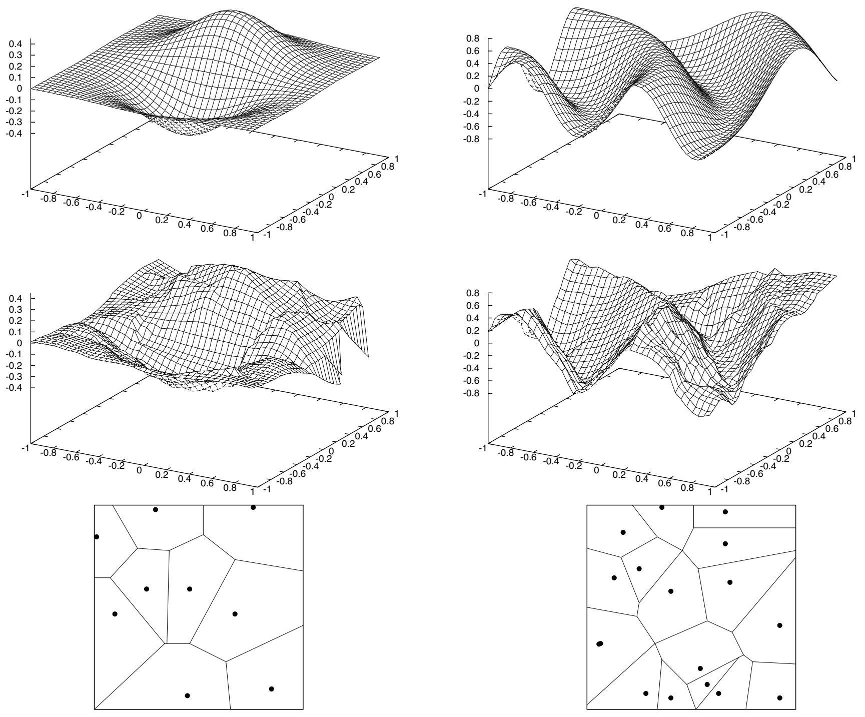

Fig. 5. Target functions, examples of approximations and Voronoi diagram partitions

superior performance when compared to standard one point crossover. In particular, it was not necessary to add a regularization term in order to avoid the solutions to be composed of a very large number of local approximators.

The evolution of individuals with RBF approximators produces an approximation $\mathcal{F}$ that is equivalent to an RBF neural network, since the computation performed by the local approximators correspond to what the nodes of the first layer do, and the combination strategy to the computation done at the output node. The usual training strategy of an RBF neural network consists in two steps [7]: a selection for the centers of the nodes in the first layer, and training in order to compute the weights that connect the two layers. Usually the first step is done through a non supervised technique (or by randomly selecting the centers), and the second step with supervised training. With the evolutionary algorithm, both steps are performed at the same time. An advantage of this approach is that it is not necessary to define in advance the number of centers (or regions) since the algorithm does the partition by itself.

It can be noted from the experiments that the evolutionary algorithm assigns more local approximators in areas that are more difficult to approximate. The concept of being difficult depends on the kind of local approximator. For example, a linear segment is easy to approximate for a linear local approximator but more difficult for an RBF approximator. A representation that allows different kinds of local approximators to be evolved at the same time in different regions of the space is under analysis now.

Future experiments will include problems defined in terms of ordinary differential equations, like for example, problems from control, or models of some physical or chemical processes [6]. The main characteristic is the fact that these problems cannot be defined just in terms of a set of input-output patterns. In the experiments performed till now, even if it could have been possible to define the problems in terms of a fixed data set, it was preferred to follow this approach as much as possible, generating new random patterns for each evaluation of the individuals. It is expected that the local approximation properties of the method proposed can demonstrate itself to be useful in the identification of complex functions that do not pertain to data fitting.

References

1. Ahmed, M. and De Jong, K.: Function Approximator Design using Genetic Algorithms. Proceedings of the 1997 IEEE Int. Conference on Evolutionary Computation, Indianapolis, IN, pp. 519–523 (1997)
2. Anton, F., Mioc D. and Gold C.: Line Voronoi Diagram based Interpolation and Application to Digital Terrain Modeling. Proceedings of the 13th Canadian Conference on Computational Geometry, University of Waterloo, (2001)
3. Back, T. Fogel, D. and Michalewicz, Z.(Eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd and Oxford University Press. (1997)
4. Boissonnat, J. and Cazals, F.: Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions. Rapport de recherche de l'INRIA – Sophia Antipolis (2000)
5. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry, Algorithms and Applications. Second Edition. Springer-Verlag. (1998)
6. Fadda, A. and Schoenauer, M., Evolutionary Chromatographic Law Identification by Recurrent Neural Nets. In J. R. McDonnell and R. G. Reynolds and D. B. Fogel, Eds, Proc. 4^{th} Annual Conference on Evolutionary Programming, pp 219–235. MIT Press, 1995.
7. Fiesler, E. and Beale R. (Eds.): Handbook of Neural Computation. Institute of Physics Publishing. (1997)
8. Koza, J.: Genetic Programming: On the Programming of Computers by means of Natural Evolution. MIT Press. (1992)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended Edition. Springer-Verlag Berlin Heidelberg. (1999)
10. Schoenauer, M., Jouve, F. and Kallel, L. Identification of Mechanical Inclusions. Dasgupta and Z. Michalewicz Ed. Identification of Mechanical Inclusions. Evolutionary Algorithms in Engineering Applications. (1997)
11. Wilson, S.: Classifiers that approximate functions. Natural Computing, 1(2–3), 211–234 (2002)
12. Zienkiewicz, O. C. and Taylor R. L.: Finite Element Method: Volume 1, The Basis. Fifth edition. Butterworth-Heinemann.(2000)