Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 706247, 11 pages
http://dx.doi.org/10.1155/2014/706247

Research Article

Hierarchical Genetic Algorithm for B-Spline Surface Approximation of Smooth Explicit Data

C. H. Garcia-Capulin,[1] F. J. Cuevas,[1] G. Trejo-Caballero,[2,3] and H. Rostro-Gonzalez[3]

[1] *Centro de Investigaciones en Óptica, A.C., Loma del Bosque 115, Col. Lomas del Campestre, 37150 León, GT, Mexico*
[2] *Instituto Tecnológico Superior de Irapuato, Carretera Irapuato Silao Km 12.5, 36821 Irapuato, GT, Mexico*
[3] *División de Ingenierías Campus Irapuato-Salamanca, Universidad de Guanajuato, Carretera Salamanca-Valle de Santiago Km 3.5+1.8 Km, Com. Palo Blanco s/n, 36885 Salamanca, GT, Mexico*

Correspondence should be addressed to C. H. Garcia-Capulin; carlosg@cio.mx

Received 8 January 2014; Revised 12 May 2014; Accepted 14 May 2014; Published 5 June 2014

Academic Editor: K. M. Liew

B-spline surface approximation has been widely used in many applications such as CAD, medical imaging, reverse engineering, and geometric modeling. Given a data set of measures, the surface approximation aims to find a surface that optimally fits the data set. One of the main problems associated with surface approximation by B-splines is the adequate selection of the number and location of the knots, as well as the solution of the system of equations generated by tensor product spline surfaces. In this work, we use a hierarchical genetic algorithm (HGA) to tackle the B-spline surface approximation of smooth explicit data. The proposed approach is based on a novel hierarchical gene structure for the chromosomal representation, which allows us to determine the number and location of the knots for each surface dimension and the B-spline coefficients simultaneously. The method is fully based on genetic algorithms and does not require subjective parameters like smooth factor or knot locations to perform the solution. In order to validate the efficacy of the proposed approach, simulation results from several tests on smooth surfaces and comparison with a successful method have been included.

1. Introduction

Surface approximation is a recurrent problem in geometric modeling, data analysis, image processing, and many other engineering applications [1]. Typically in engineering applications the approximating techniques are applied both to the reverse engineering problem and to design a surface that matches a set of desired characteristics. In reverse engineering, one is concerned with the automated generation of a CAD-model from a set of points digitized from an existing 3D object.

In this regard, the surface approximation problem may be formulated in different ways, yielding results depending on various choices. B-spline surfaces can be used to approximate an arbitrary set of data analytically; in this case a system of simultaneous equations is obtained, which is linear with respect to the weights but is nonlinear with respect to the knots. Therefore, the choice of the number and positions of the knots of a spline is fundamental, as well as the method used to solve the system of equations. Both tasks are critical but also troublesome, leading to a hard continuous multimodal and multivariate nonlinear optimization problem with many local optimal solutions.

To tackle this problem, several methods have been proposed in the literature. For instance, Shepard's method [2], the finite element method [3, 4], and the tensor product spline method [5–8] are the most widely used and successful methods. Shepard's method, also known as the original inverse distance weighted interpolation method, deals with this issue through a continuous interpolation function from the weighted average of the data. The finite element method is a numerical approach for solving differential equations. This method consists of assuming the piecewise continuous function for the solution and obtaining the parameters of

the functions in a manner that reduces the error in the solution. The tensor product spline is another method commonly used to approximate surfaces. It is a generalization of the spline approximations, which aims to get smooth functions from scattered points [9].

Other techniques based on intelligent search schemes have been proposed. In [10], the authors reconstruct a surface from a given set of data points, based on a two-step algorithm using particle swarm optimization. In contrast, in [11] a genetic algorithm is applied iteratively to fit a given set of data points in two steps. Additionally, in both previous approaches, a least squares solution must be performed to compute the B-spline coefficients [12, 13]. In order to get this solution, it is often convenient that knot distribution follows the rule given by the Schoenberg-Whitney theorem [14].

On the other hand, in [8] the authors use a genetic algorithm to optimize the number and locations of the knots in order to construct a B-spline curve that fits a set of data points. A modification of [8] is presented in [15]. Here the authors use a real-coded genetic algorithm that can treat data having discontinuous points. Another example is presented in [16]. Here the authors use a firefly algorithm to compute the approximating explicit B-spline curve to a given set of noisy data points. Even although these works address the case of curves, they can be applied to approximate a surface by repeated applications of the method for one-dimensional data for x and y directions separately but with considerable time consumption.

Since we consider a spline based approach, we remark the fact that the main issue associated with the surface approximation through splines is to find the best set of knots, where the term "best" implies an adequate choice in the number and location of the knots. To perform this task, in [5], the author provides a survey on the main algorithms used to carry out this task, which are based on regression spline methods and their respective optimizations.

Unlike the authors mentioned above, we tackle the B-spline surface approximation problem by using the hierarchical genetic algorithm. To be more specific, we consider a hierarchical structure to represent both the model structure (number and location of knots) as a binary encoding and the model parameters (spline coefficients) as a real encoding. Thus, we search for the best B-spline based surface model using a novel fitness function. As a result, our method can simultaneously determine the number and position of the knots as well as B-spline coefficients. In addition, our approach is able to find solutions with fewest parameters within the B-spline basis functions.

In this work, we focus on the approximation to a given set of noise data points, which are arranged in a rectangular domain, under the assumption that these data represent a smooth surface. These are the conditions under which the method can be applied. However, our method can be extended to approximate unsmooth functions over scattered data points.

This paper is organised as follows: the fundamentals of HGA and the description of B-spline surfaces are presented in Section 2, followed by the description of our approach in Section 3. In Section 4 we present some numerical results and a comparison with popular method. Finally, in Section 5 the paper closes with the main conclusions.

2. Background

In this section, the fundamentals of the hierarchical genetic algorithms and surface approximation by splines are explained in detail.

2.1. Hierarchical Genetic Algorithms. Genetic algorithms (GA) [17], originally developed by Holland, have been used to optimize a fitness function by mimicking the natural evolution of living organisms. Individuals of this evolution are computational representations of potential solutions for the problem to be solved. Each individual is represented as a computational encoding and is commonly called chromosome. The entire set of individuals examined at a given time is called the population; this population is evaluated with a fitness function to determine which individuals represent better solutions to the problem. Fitness values are used by the selection operator to scale the likelihood that a chromosome will be used to produce a new chromosome. The production of new chromosomes from old ones is achieved through the application of the genetic operators, crossover, and mutation. An iterated application of selection and genetic operators is repeated until a desired termination criterion is reached.

On the other hand, HGA has a hierarchical structure of chromosomes. From biological and medical viewpoints, the genetic structure of a chromosome is formed by a number of gene variations arranged in a hierarchical manner. In the light of this issue, Man et al. [18] proposed a hierarchical structure of chromosome to emulate the formulation of a biological DNA structure, where genes dominate other genes and there are active and inactive genes. Such a phenomenon is a direct analogy to the topology of many engineering systems [19]. Thus, allow us to emulate the formulation of a biological DNA structure so that a precise hierarchical genetic structure can be formed for engineering purpose.

The computational chromosome in an HGA consists of two types of genes called control and parametric genes. Typically, control genes are represented as a binary encoding, and the parametric genes are coded as real numbers, although these last ones can be coded as any type of data structure. The purpose of control genes is to represent the parametric selection; this is particularly important for determination of structure or topology on the individuals. On the other hand, the parametric genes are usually used to represent the parameters of the designed variables. The main advantage of HGA is that both the system structure and the parametric variables can be optimized in a simultaneous way and that the standard genetic operators continue being used. A more detailed discussion about HGA can be found in [18].

2.2. Surface Approximation by Tensor Product Splines. We can describe the problem of surface approximation as follows: given a set of noisy measurements in a rectangular domain

described as $z_{i,j}$, $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$, and expressed in the following form:

$$z_{i,j} = f\left(x_i, y_j\right) + \epsilon_{i,j}, \tag{1}$$

where f is an unknown functional relationship that we wish to estimate, the term $\epsilon_{i,j}$ represents the zero-mean random errors, and $z_{i,j}$ is a sample at (x_i, y_i). Therefore, the goal is to find the best estimation of the function f.

In this study, we assume that f is a smooth surface that can be well approximated in the interval $[a, b] \times [c, d]$ by a B-spline surface. The B-spline surfaces are constructed as a tensor product of univariate B-spline basis functions. The B-spline surface is modeled using the following considerations: let us define $\{u_1, \ldots, u_m\}$ as a set of m points placed along the domain of the variable x and let $\{v_1, \ldots, v_n\}$ be a set of n points placed along the domain of the variable y, which are called interior knots. Thus, the knot vectors are defined as follows:

$$\begin{aligned} \mathbf{u} : u_{1-k} &= \cdots = u_0 \\ &= a < u_1 < \cdots < u_m < b = u_{m+1} = \cdots = u_{m+k}, \\ \mathbf{v} : v_{1-l} &= \cdots = v_0 \\ &= c < v_1 < \cdots < v_n < d = v_{n+1} = \cdots = v_{n+l}. \end{aligned} \tag{2}$$

With these assumptions, the function f can be now written as a tensor product:

$$f(x, y) = \sum_{i=1}^{m+k} \sum_{j=1}^{n+l} P_{i,j} B_{i,k}(x) B_{j,l}(y), \tag{3}$$

where $P_{i,j}$ are the B-spline coefficients and $B_{i,k}(x), B_{j,l}(y)$ are the B-spline basis functions of order k and l, respectively, defined over the knot vectors $\mathbf{u}$ and $\mathbf{v}$. The B-spline basis functions are denoted by the following recurrence relations:

$$\begin{aligned} B_{i,1}(x) &= \begin{cases} 1, & \text{if } t_i \le x < t_{i+1} \\ 0, & \text{otherwise,} \end{cases} \\ B_{i,k}(x) &= \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x). \end{aligned} \tag{4}$$

If k and l are specified beforehand, f can be completely specified by $\theta = \{\mathbf{u}, \mathbf{v}, P\}$, where $\mathbf{u}$ and $\mathbf{v}$ are the knot vectors and P is the coefficient matrix. Now, the problem is to find the number and location of the interior knots $\{u_1, \ldots, u_m, v_1, \ldots, v_n\}$ and then estimate the coefficients $P_{i,j}$. This problem cannot be solved with simple standard methods due to fact that the parameter estimation in B-spline based methods is a high-dimensional nonlinear optimization problem. A more detailed discussion about B-splines can be found in [20, 21].

3. B-Spline Surface Approximation Using HGA

In this paper, we use an HGA to determine simultaneously the number and positions of the knots (model structure) and the B-spline coefficients (model parameters) by minimizing a fitness function. In this approach, the main characteristics to consider are (1) the chromosome encoding of potential solutions, (2) the fitness function, to evaluate the fitness of the chromosomes, and (3) the operators to evolve the individuals.

3.1. Chromosome Encoding. We use a fixed length binary string to represent the number and the locations of the interior knots, $\{u_1, \ldots, u_m, v_1, \ldots, v_n\}$, and real numbers to represent the P B-spline coefficients. Furthermore, we assume that u_i is a subset of design points x_i and v_i is a subset of design points y_i. Thus, the maximum number m of interior knots is equal to the number of points x_i on interval (a, b) and the number of coefficients is $m + k$, as well as the maximum number n of interior knots is equal to the number of points y_i on interval (c, d) and the number of coefficients is $n + l$.

Once the previous assumptions are stated, we can represent the chromosome of an individual as

$$\begin{aligned} \theta = \{&b_1, \ldots, b_m, b_{m+1}, \ldots, b_{m+n}, r_{1-k/2,1-l/2}, \ldots, r_{1-k/2,n+l/2}, \\ &\ldots, r_{m+k/2,1-l/2}, \ldots, r_{m+k/2,n+l/2}\}, \end{aligned} \tag{5}$$

where each b_i is a control bit and $r_{i,j}$ is a real value (coefficient). Here, each control bit enables or disables one of the interior knots and one of the coefficients simultaneously. We establish one-to-one correspondences between the interior knots and the coefficients to be activated at the same time. The real values represent the coefficients of the B-spline. The general structure of a chromosome is graphically shown in Figure 1. This representation scheme does not allow us to duplicate knots, because our interest is on smooth functions. However, it can be extended to handle discontinuous functions if we introduce an additional type of gene.

Finally, the number of interior knots m and n is limited by the number of points in each surface dimension. However, in [22, 23] the authors recommend placing a knot after every three x_i design points; that is, the number of knots interior can be divided by three. This is enough to ensure that a knot is at, or near, the positions required to capture the topology of the surface.

3.2. Fitness Function. To evaluate the fitness of each individual θ, the fitness function F is formulated as a sum of three terms and is given by the following equation:

$$F(\theta) = \omega_1 \mathrm{RSS} + \omega_2 \mathrm{SSD} + \omega_3 \mathrm{PKS}, \tag{6}$$

where each term of the equation is described as follows.

(a) The first term is the residual sum of squares (RSS). It is used as a measure of the deviation between the observed data set and the estimated function $\widehat{f}$. The RSS is calculated as follows:

$$\mathrm{RSS} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left\{z_{i,j} - \widehat{f}\left(x_i, y_j\right)\right\}^2. \tag{7}$$

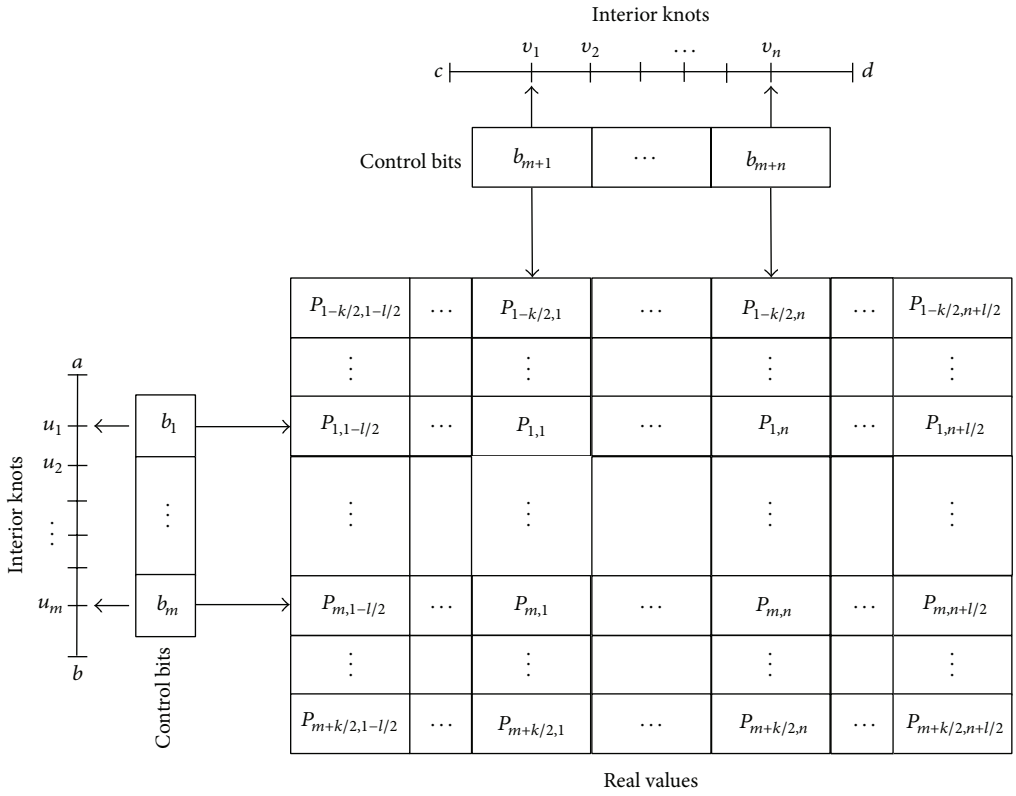


Figure 1: General structure of a chromosome.

(b) The second term is the sum of the squared differences (SSD), which is the discrete approximation of the gradient. This term is used to penalize high sums of gradients to generate smooth solutions. The SSD is given by the next equation:

$$\mathrm{SSD} = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left\{\left[\hat{f}\left(x_i, y_j\right) - \hat{f}\left(x_{i-1}, y_j\right)\right]^2 + \left[\hat{f}\left(x_i, y_j\right) - \hat{f}\left(x_i, y_{j-1}\right)\right]^2\right\}. \tag{8}$$

(c) The last term is a penalty function for knot structure (PKS). It is computed as follows:

$$\mathrm{PKS} = \sum_{i=0}^{m}\frac{b-a}{\left(u_{i+1}-u_i\right)^2} + \sum_{i=0}^{n}\frac{d-c}{\left(v_{i+1}-v_i\right)^2}. \tag{9}$$

In (9), PKS tries to favor solutions with uniform distributions of knots for each dimension. In other words, it penalizes solutions with knots very close, which generates over fitting of the function. Therefore, the individuals with fewest knots and better distribution are favoured.

3.3. Operators

3.3.1. Selection Operator. The roulette wheel method is used as a selection operator. In this method, each individual is assigned to one of the slices in the roulette wheel. This selection strategy favors best fitted individuals but also gives a chance to the less fitted individuals to survive. To prevent premature convergence, the sigma scaling method [24] is used. This method tries to keep the selection pressure relatively constant over all evolution process, and it is calculated according to

$$F_{\mathrm{new}}\begin{cases} F_{\mathrm{act}} - \left(\overline{F} - c\cdot\sigma\right) & \text{if } \left(F_{\mathrm{act}} > \overline{F} - c\cdot\sigma\right) \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where F_{new} is the new scaled fitness value, F_{act} is the current fitness value, $\overline{F}$ is the average fitness, σ is the standard deviation of the population, and c is a constant to control the selection pressure. In addition, elitism is used in order to keep elite individuals in the next population to prevent losing the best solution found.

3.3.2. Crossover Operators. The uniform crossover operator is used for the binary-valued chromosome and the simulated binary crossover operator (SBX) is used for the real-valued chromosome [25]. These crossover operators are applied with the same crossover probability. In the uniform crossover method, two parents are chosen to be recombined into a new individual. Each bit of the new individual is selected from one of the parents depending on a fixed probability. On the other hand, in SBX method, two new individuals c_1 and c_2 are generated from the parents p_1 and p_2 using a probability distribution.

The procedure used in SBX is the following: first, a random number u between 0 and 1 is generated. Then the probability distribution β is calculated as

$$\beta = \begin{cases} (2u)^{1/(\eta+1)} & \text{if } u \leq 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta+1)} & \text{otherwise,} \end{cases} \tag{11}$$

where η is a nonnegative real number that controls the distance between parents and the new individuals generated. After obtaining β, new individuals are calculated according to

$$\begin{aligned} c_1 &= 0.5\left[(1+\beta)\,p_1 + (1-\beta)\,p_2\right], \\ c_2 &= 0.5\left[(1-\beta)\,p_1 + (1+\beta)\,p_2\right]. \end{aligned} \tag{12}$$

3.3.3. Mutation Operators. For the binary-valued chromosome, the bit mutation method is used. In this method, each bit is inverted or not depending on a mutation probability. For the real-valued chromosome each numeric value γ is changed depending on the same mutation probability according to

$$\gamma_i = \gamma_i + \delta\,(\text{rand} - 0.5), \tag{13}$$

where δ is the maximum increment or decrement of the real value and rand is a function that generates a random value between 0 and 1.

This operator acts on each gene of the chromosome independently, generating a binomial distribution of mutated genes. Thus, the mutation probability is applied per locus per replication.

3.4. HGA Parameters Tuning. We carried out experiments in order to tune the HGA parameters. We have tuned the parameters of HGA by running our algorithm over 100 noisy data sets generated from a test function. The test function is generated as is stated in Section 4. To analyze the process of convergence, the fitness of the best individual is recorded over 1500 generations for each noisy data set tested.

First, we analyze the variation of the mutation probability of the HGA while all other parameters are fixed. In Figure 2, we can see the results of the variation of the mutation probability from 0.005 to 0.050. As we can see in the same figure, the HGA performs better with a mutation probability of 0.008 (line in color green). From Figure 2, we can observe that the performance of HGA-based method decrease with an increase of the mutation probability.

On the other hand, we vary the population size and we analyzed their effects in the process of convergence of HGA. Figure 3 shows the performance of HGA for several population sizes. We can see that the algorithm performance is quite similar for population sizes of 90, 120, and 200 (lines in color red, cyan, and magenta resp.). The average time of execution for each population size is shown in Table 1. Therefore, the best choice is the population size of 90 individuals.

Finally, in Figure 4, the performance of HGA for 3000 generations is shown. In this figure, the 50 best runs of HGA are plotted. As it can be seen in this figure, the convergence of the HGA is achieved at the 1500th generation.

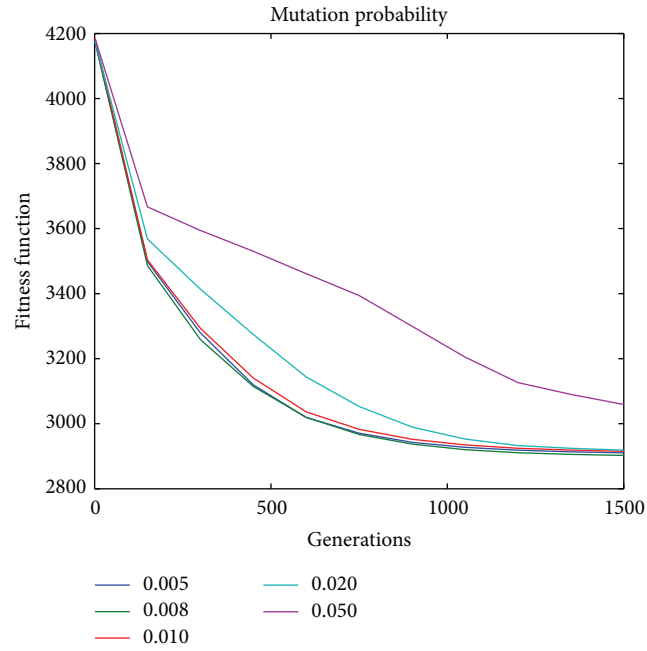


Figure 2: Variation of the mutation probability. Each line shows the average over 100 test sets.

Table 1: Average computation time for the HGA over 100 test sets.

Population size	Time (s)
30	34
60	68
90	103
120	136
200	226

4. Numerical Results

We carried out numerical simulations to evaluate the performance of our approach. Thus, in order to perform these tests, we defined an experimental set of four bivariate functions, whose equations are given in Table 2 and graphically shown in Figure 5. These test functions were taken from previous works [5, 26–28] as a reference to validate our method. Besides, a comparison with the well-known locally weighted scatter smoothing (LOWESS) method [29] is presented.

In this study, an experimental setup was configured for each test function as follows: the function is evaluated at 1024 points in a square grid of 32×32 over the interval $[0, 1] \times [0, 1]$. Then the data are translated to make the range nonnegative in order to facilitate comparisons among them. Here, a zero-mean normal noise with σ known is added to the data set. An example of the noisy data sets generated for the four test functions are shown in Figure 7. Furthermore, for each experimental setup, a collection of 100 noisy data sets is generated at three different signal noise ratios (SNR) 2, 3, and 4, respectively, where SNR is defined as $\text{SD}(f)/\sigma$. Thus, we tested our method over a total of 1200 data set for the four

Table 2: Experimental set of four bivariate functions.

Number	Test function
01	$f(x,y) = 42.659\{0.1 + \hat{x}(0.05 + \hat{x}^4 - 10\hat{x}^2\hat{y}^2 + 5\hat{y}^4)\}, \hat{x} = x - 0.5, \hat{y} = y - 0.5$
02	$f(x,y) = 1.3356[1.5(1-x) + e^{(2x-1)}\sin\{3\pi(x-0.6)^2\} + e^{(3(y-0.5))}\sin\{4\pi(y-0.9)^2\}]$
03	$f(x,y) = 1.9[1.35 + e^x \sin\{13(x-0.6)^2\}e^{-y}\sin(7y)])$
04	$f(x,y) = 3 + 2\sin(2\pi x)\sin(2\pi y) + e^{\{-15(x-1)^2-15(y-1)^2\}} + e^{\{-10x^2-15(y-1)^2\}}$

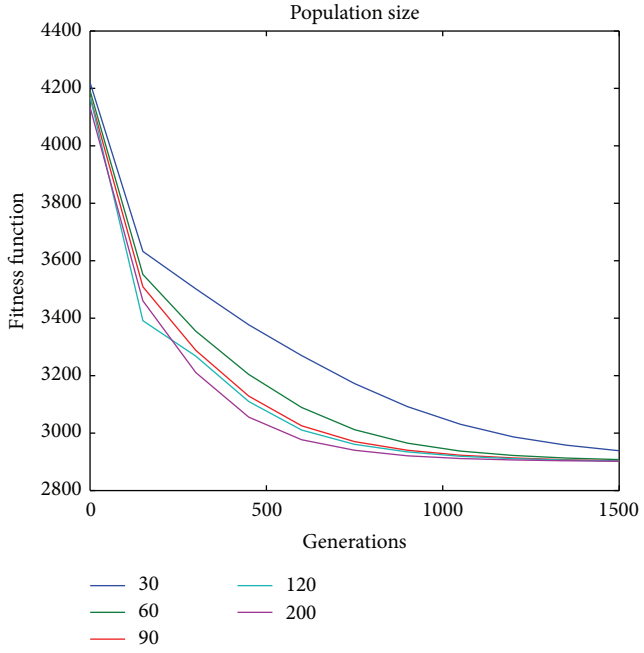


Figure 3: Variation of the population size. Each line shows the average over 100 test sets.

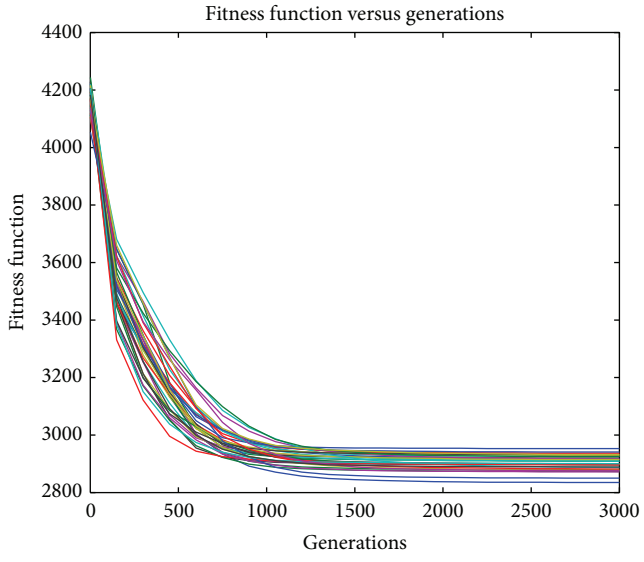


Figure 4: Evolution of best individuals for the 50 best tests.

Table 3: Parameters used for the HGA.

Parameter	Value
Population size	90
Crossover probability	0.85
Mutation probability	0.008
Number of elite individuals	9

functions. Finally, the proposed approach and the LOWESS method were applied to estimate the test functions.

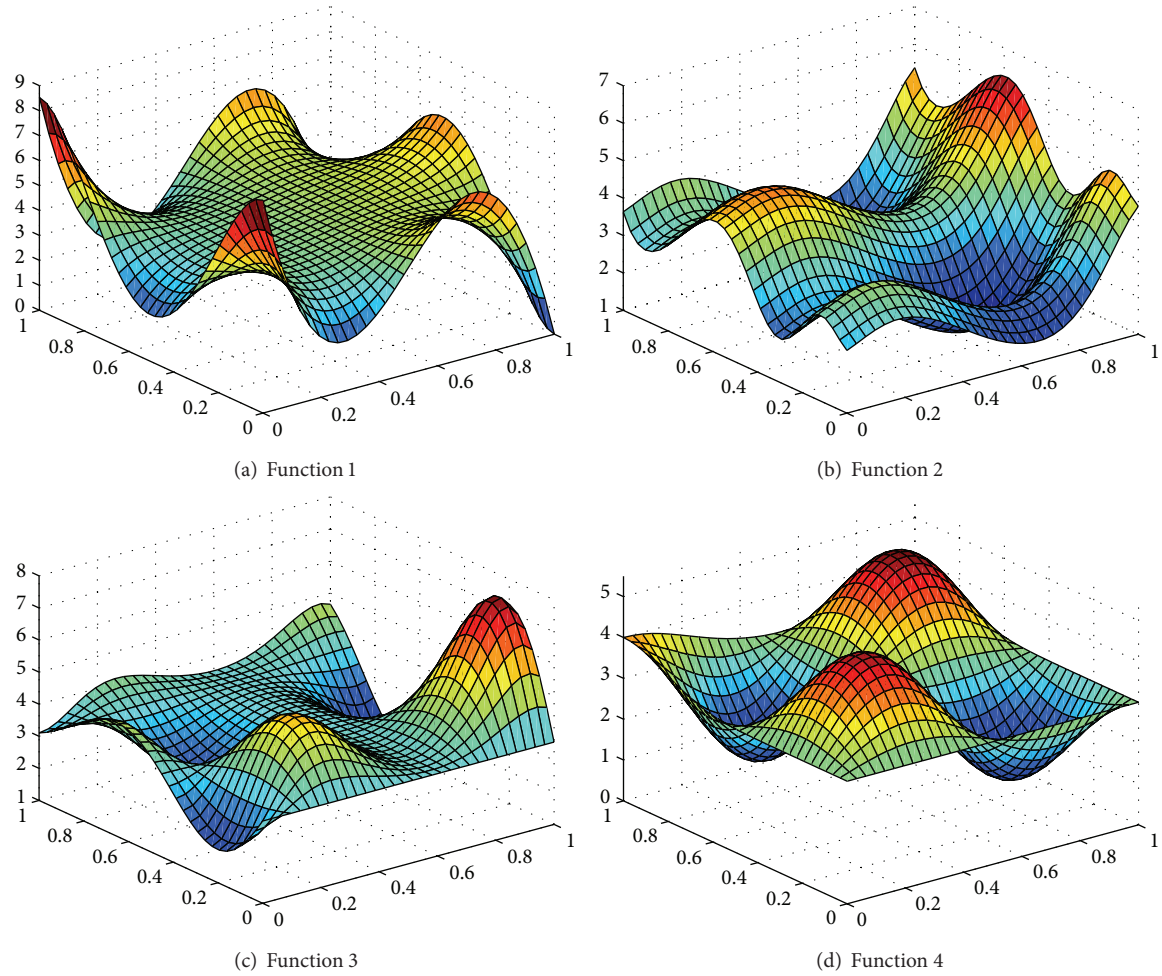
In the numerical tests, our approach was configured as follows: we used cubic B-spline functions, that is, $k = 4$ and $l = 4$, and the interior knots are defined as a subset of design points ($m_{\max} = 32$, $n_{\max} = 32$). The population was randomly initialized at the beginning. Each control gene b_i is randomly selected from $[0, 1]$ and each parametric gene r_i is calculated as a random real number defined over the range $[\min(z_{i,j}), \max(z_{i,j})]$ of the measurements $z_{i,j}$. The HGA parameters were tuned experimentally and they are presented in Table 3. The population was evolved during 3000 generations in all cases.

For comparison purposes, we performed the same numerical experiments with the LOWESS method. For this, we made use of the Curve Fitting Toolbox provided by MATLAB. In this simulation, the default parameters were considered. To compare the results, we used the mean square error (MSE) given by

$$\text{MSE} = \frac{1}{N_x N_y}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left\{\hat{f}\left(x_i, y_j\right) - f\left(x_i, y_j\right)\right\}^2, \tag{14}$$

where $\hat{f}$ is the estimated function given by each method applied and f is the real function. The MSE value is computed and recorded for each of the 100 test sets for the four test function. The box plots of MSE($\hat{f}$) values are shown in Figure 6. The box plot shows a visual representation of the distribution of MSE values and facilitates the comparison between them.

The box plot partitions a data distribution into quartiles. The quartiles divide the data into four equal parts; 25% of the data are in each part. The central box is used to indicate the positions of the upper (third) and lower (first) quartiles; the interior of this box indicates the interquartile range (IQR), which is the distance between the upper and the lower quartiles and consists of 50% of the data. The horizontal line inside the box is the median value of the distribution. The vertical lines, sometimes referred to as whiskers, extended from the box out to the smallest and the largest observations, show the full spread of the distribution;



(a) Function 1 (b) Function 2

(c) Function 3 (d) Function 4

Figure 5: Perspective plots of test functions.

if there are observations that fall more than 1.5 × IQR above the upper quartile or below the lower quartile, these are considered as suspected outliers and they are plotted as individual points marked with a cross.

In Figure 6, for each combination of test function and SNR there is a panel, in which the box plots of MSE values from HGA and LOWESS are arranged in pairs. The noise level added to the signal is decreased from left to right. As we can see in the same figure, in most cases the MSE values of HGA are considerably lower than those from the LOWESS algorithm. In some cases the upper quartile of the MSE values for the HGA is under the first quartile of those from LOWESS, which implies that 75% of MSE values are lower than 75% of MSE values from LOWESS; in others words, less than 25% of MSE values for the LOWESS algorithm are comparable with the 75% of those from HGA. The highest difference can be seen on Figure 6 for the test function 4 with SNR = 4, where the upper quartile of HGA is far below the lower quartile of LOWESS. Moreover the median of MSE values for HGA is the lowest virtually for all cases.

On the other hand, the spread of the middle half (the box) of the MSE values it seems less variable and symmetric around the median in some cases for LOWESS method, contrary to those from HGA. In addition, although both algorithms present unusual MSE values known as outliers, the LOWESS method has fewer outliers than the HGA. We consider that this behavior represents cases where the HGA converges towards local optima or even arbitrary points rather than the global optimum; nevertheless, this behavior is expected in any heuristic technique.

Additionally, we computed the mean and standard error of mean for MSE values from both methods. They are summarized in Table 4. We see from this table that HGA achieved the lowest MSE values on all experiments. Particularly, we can see the most significant statistical difference appearing in test function 4 and SNR = 4, where the mean of the MSE values from HGA is far away from the mean of MSE values from the LOWESS method. The most statistically significant difference is marked in bold. As for the standard errors, although the LOWESS method has lower values than those from HGA, the differences between means are significant, so despite this; the results of HGA are better.

To visually evaluate the performance of our approach, an example from the obtained results for each function is graphically shown in Figure 7. For practical reasons, we only present results from the HGA, together with an example of

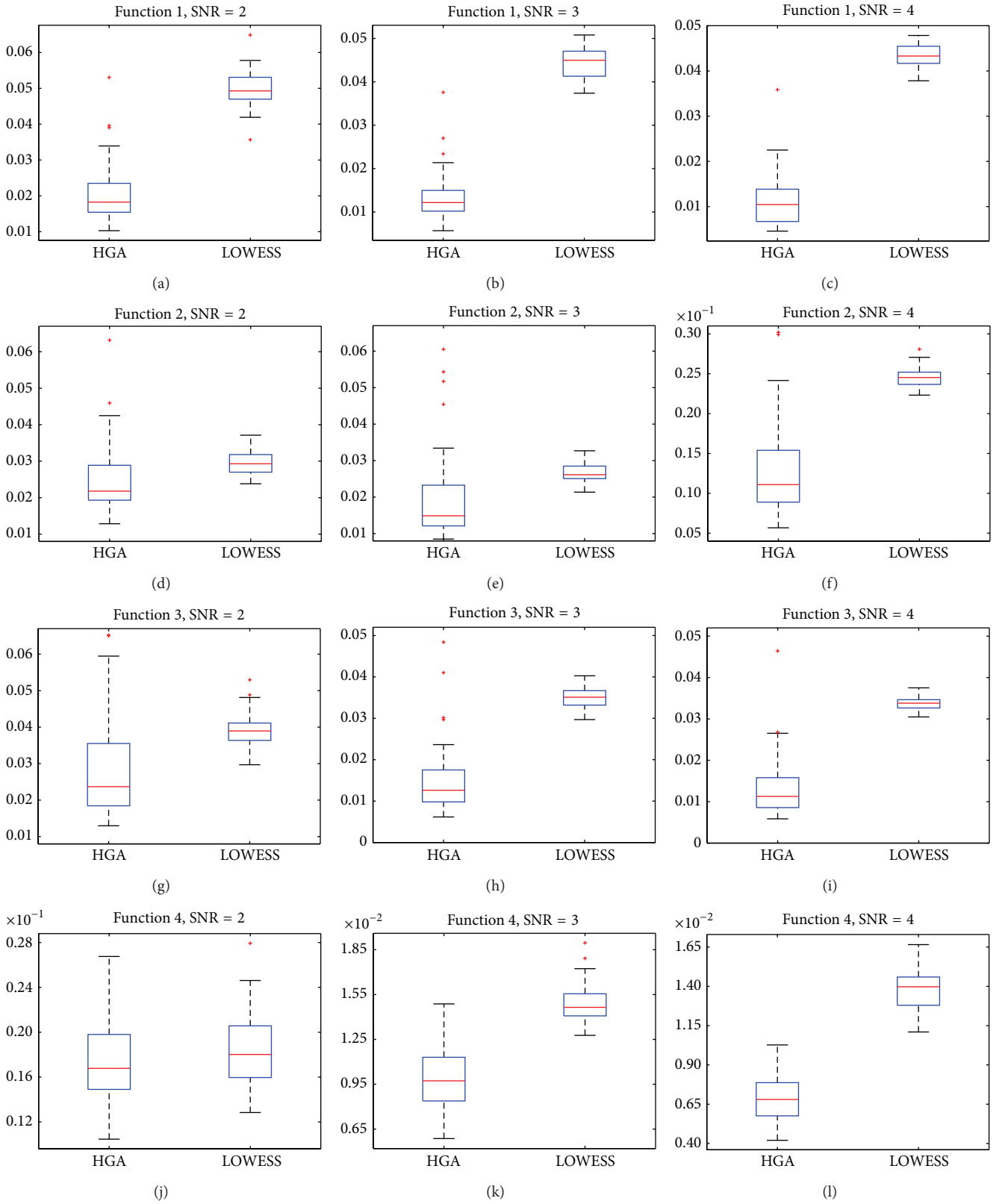


Figure 6: Boxplots of the MSE values for both methods. In each panel, the boxplots correspond to the algorithms tested. The columns of panels correspond from left to right to 2, 3, and 4 SNR ratios. The rows of panels correspond from top to bottom to test functions 1 to 4.

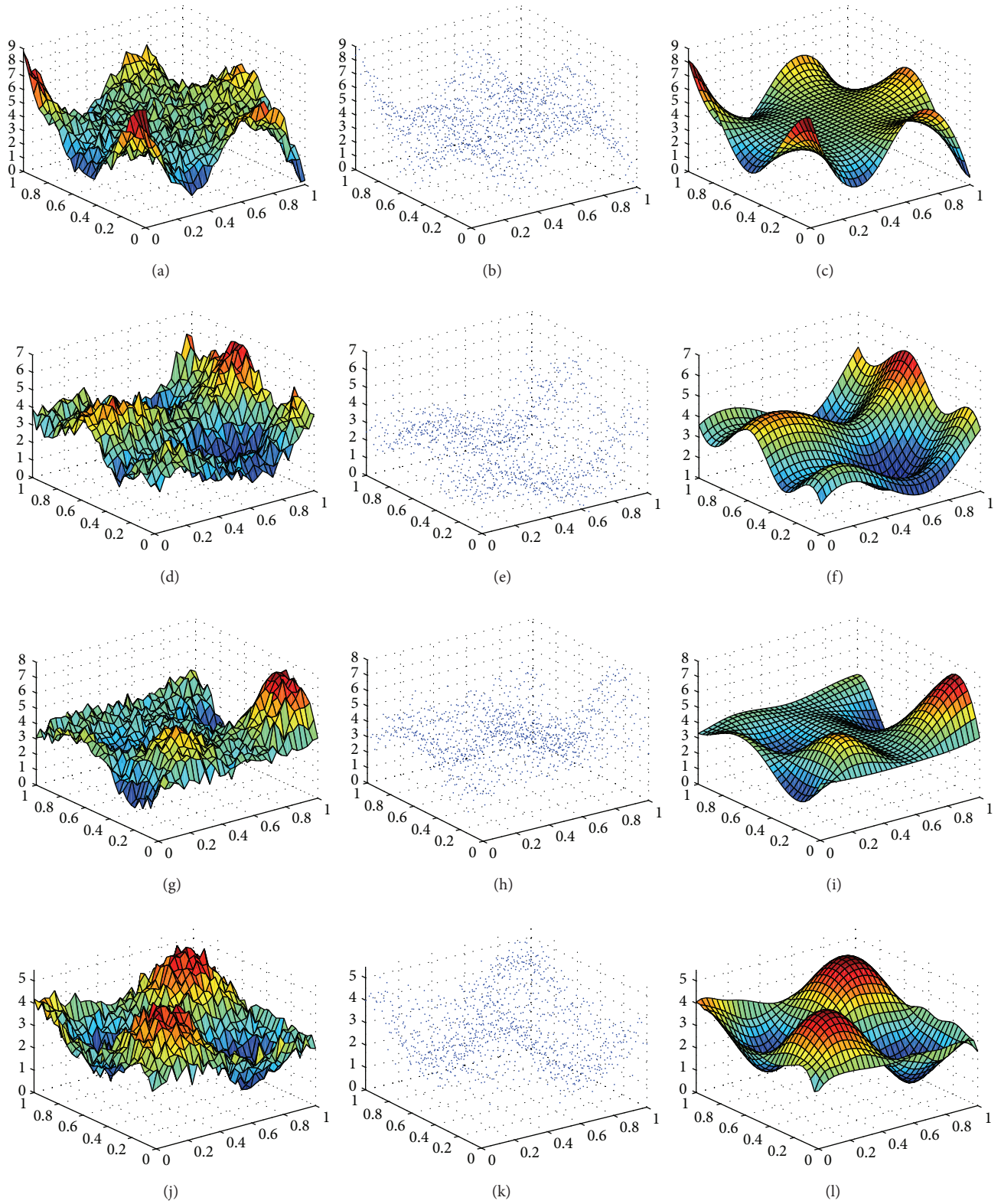


Figure 7: Results from HGA. The columns correspond from left to right: noise surface inputs and cloud data inputs, approximated surfaces by HGA. The rows correspond from top to bottom to the test functions 1 to 4.

Table 4: Simulation results. Mean of the computed MSEs with estimated standard errors based on 100 test sets.

Test	Algorithm	SNR = 2	SNR = 3	SNR = 4
1	LOWESS	0.049889 ± 0.000722	0.044644 ± 0.000461	0.043435 ± 0.000349
	HGA	0.020792 ± 0.001195	0.013381 ± 0.000757	0.011380 ± 0.000824
2	LOWESS	0.029555 ± 0.000464	0.026570 ± 0.000325	0.024656 ± 0.000180
	HGA	0.028086 ± 0.002732	0.025059 ± 0.004591	0.017700 ± 0.002964
3	LOWESS	0.039298 ± 0.000621	0.035072 ± 0.000337	0.033791 ± 0.000218
	HGA	0.030230 ± 0.002457	0.014920 ± 0.001158	0.013911 ± 0.001307
4	LOWESS	0.018480 ± 0.000434	0.014860 ± 0.000172	**0.013765** ± 0.000163
	HGA	0.017167 ± 0.000538	0.009926 ± 0.000295	**0.006886** ± 0.000221

the simulated noisy data. From this figure, we can qualitatively evaluate the performance of our algorithm. Figure 7 suggests that the HGA has the ability to adapt the surfaces optimally.

Finally, it could be assumed that the success of our approach is due to the simultaneous global search performed over all parameters of the B-spline model. This shows the potential of the HGA to manage the loss of information and a high noise level. On the other hand, the main drawback is the computational time required for the HGA compared with the LOWESS method. The average computation time for the LOWESS method was 4 seconds for all functions tested, compared to 201 seconds for the HGA for all functions tested. However, this drawback can be solved if we consider a parallel implementation of the HGA. Our algorithm was implemented in C++ language. Both algorithms were executed on a PC using an AMD Phenom II processor running at 1.9 GHz with 4 MB of RAM.

5. Conclusions

In this paper, we proposed an efficient hierarchical genetic algorithm to tackle the B-spline surface approximation problem. The method introduced a novel hierarchical gene structure for the chromosomal representation, thus allowing us to find simultaneously the best model with the fewest knots, optimal knot locations, and coefficients of the B-spline surface. It is important to highlight the fact that the method does not require subjective parameters like smooth factor or knot locations to perform the solution.

To test our method, we performed several tests on benchmark functions as well as a comparison with the LOWESS method, which is provided with the Matlab Curve Fitting Toolbox. Numerical results show that our method responds successfully to the problem of surface approximation. In terms of visualization (qualitatively), the obtained results are comparable to the original surfaces. Comparative tests demonstrated a better performance of our method than the LOWESS method over all the proposed tests. Given the performance characteristics of the proposed approach, our future work will be to apply this method over an experimental data set. We are interested in extending our approach to experiment with variable length chromosome and different basis functions.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

We acknowledge the support of the Consejo Nacional de Ciencia y Tecnología de México, Centro de Investigaciones en Óptica, A.C., and Instituto Tecnológico Superior de Irapuato. This work has been partially funded by the CONACYT Project no. 229839 (Apoyo al Fortalecimiento y Desarrollo de la Infraestructura Científica y Tecnológica 2014) and the SEP through the PIFI-UGTO 2013.

References

[1] A. Safari, H. G. Lemu, S. Jafari, and M. Assadi, "A comparative analysis of nature-inspired optimization approaches to 2D geometric modelling for turbomachinery applications," *Mathematical Problems in Engineering*, vol. 2013, Article ID 716237, 15 pages, 2013.

[2] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 23rd ACM National Conference*, pp. 517–524, 1968.

[3] M. A. Olshanskii, A. Reusken, and J. Grande, "A finite element method for elliptic equations on surfaces," *SIAM Journal on Numerical Analysis*, vol. 47, no. 5, pp. 3339–3358, 2009.

[4] L. D. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1131–1147, 1993.

[5] T. C. M. Lee, "On algorithms for ordinary least squares regression spline fitting: a comparative study," *Journal of Statistical Computation and Simulation*, vol. 72, no. 8, pp. 647–663, 2002.

[6] A. Mazroui, H. Mraoui, D. Sbibih, and A. Tijini, "A new method for smoothing surfaces and computing hermite interpolants," *BIT Numerical Mathematics*, vol. 47, no. 3, pp. 613–635, 2007.

[7] T. Schütze and H. Schwetlick, "Bivariate free knot splines," *BIT Numerical Mathematics*, vol. 43, no. 1, pp. 153–178, 2003.

[8] F. Yoshimoto, M. Moriyama, and T. Harada, "Automatic knot placement by a genetic algorithm for data fitting with a spline," in *Proceedings of the International Conference on Shape Modeling and Applications*, pp. 162–169, IEEE Computer Society Press, 1999.

[9] H. Pottmann and S. Leopoldseder, "A concept for parametric surface fitting which avoids the parametrization problem," *Computer Aided Geometric Design*, vol. 20, no. 6, pp. 343–362, 2003.

[10] A. Gálvez and A. Iglesias, "Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points," *Information Sciences*, vol. 192, pp. 174–192, 2012.

[11] A. Gálvez, A. Iglesias, and J. Puig-Pey, "Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction," *Information Sciences*, vol. 182, no. 1, pp. 56–76, 2012.

[12] C.-G. Zhu and R.-H. Wang, "Least squares fitting of piecewise algebraic curves," *Mathematical Problems in Engineering*, vol. 2007, Article ID 78702, 11 pages, 2007.

[13] H. Haron, A. Rehman, D. I. S. Adi, S. P. Lim, and T. Saba, "Parameterization method on B-spline curve," *Mathematical Problems in Engineering*, vol. 2012, Article ID 640472, 22 pages, 2012.

[14] I. J. Schoenberg and A. Whitney, "On Pólya frequence functions—III. The positivity of translation determinants with an application to the interpolation problem by spline curves," *Transactions of the American Mathematical Society*, vol. 74, pp. 246–259, 1953.

[15] F. Yoshimoto, T. Harada, and Y. Yoshimoto, "Data fitting with a spline using a real-coded genetic algorithm," *Computer Aided Design*, vol. 35, no. 8, pp. 751–760, 2003.

[16] A. Gálvez and A. Iglesias, "Firefly algorithm for explicit B-spline curve fitting to data points," *Mathematical Problems in Engineering*, vol. 2013, Article ID 528215, 12 pages, 2013.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, 1992.

[18] K.-F. Man, K.-S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Designs with Disk*, Springer, New York, NY, USA, 1999.

[19] K. F. Man and K. S. Tang, "Genetic algorithms for control and signal processing," in *Proceedings of the 23rd Annual International Conference on Industrial Electronics, Control, and Instrumentation (IECON '97)*, pp. 1541–1555, November 1997.

[20] C. de Boor, "On calculating with B-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[21] C. de Boor, *A Practical Guide to Splines*, Springer, New York, NY, USA, 1978.

[22] M. Smith and R. Kohn, "Nonparametric regression using Bayesian variable selection," *Journal of Econometrics*, vol. 75, no. 2, pp. 317–343, 1996.

[23] J. H. Friedman and B. W. Silverman, "Flexible parsimonious smoothing and additive modeling," *Technometrics*, vol. 31, pp. 3–39, 1989.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1989.

[25] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[26] J.-N. Hwang, S.-R. Lay, M. Maechler, R. D. Martin, and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 342–353, 1994.

[27] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith, "Bayesian MARS," *Statistics and Computing*, vol. 8, no. 4, pp. 337–346, 1998.

[28] T. X. Yue and S. H. Wang, "Adjustment computation of HASM: a high-accuracy and high-speed method," *International Journal of Geographical Information Science*, vol. 24, no. 11, pp. 1725–1743, 2010.

[29] W. S. Cleveland, "Robust locally weigthed regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 829, 836 pages, 1979.