Brief Papers

Interval-Valued GA–P Algorithms

Luciano Sánchez

***Abstract*—When genetic programming (GP) methods are applied to solve symbolic regression problems, we obtain a point estimate of a variable, but it is not easy to calculate an associated confidence interval. We have designed an interval arithmetic-based model that solves this problem. Our model extends a hybrid technique, the GA–P method, that combines genetic algorithms and genetic programming.**

Models based on interval GA–P can devise an interval model from examples and provide the algebraic expression that best approximates the data. The method is useful for generating a confidence interval for the output of a model, and also for obtaining a robust point estimate from data which we know to contain outliers.

The algorithm has been applied to a real problem related to electrical energy distribution. We were asked to search for a model capable of producing a confidence interval for the total length of electrical line in some Spanish rural villages, given their area and the number of inhabitants. Classical methods were applied first, and interval GA–P later. The results of both studies are used to compare interval GA–P with GP, GA–P, classical regression methods, neural networks, and fuzzy models.

***Index Terms*—Genetic algorithms, genetic programming, regression problems.**

I. Introduction

NTERVAL arithmetic-based models can associate the input of a system with a confidence interval for its output. For example, we can relate the height of a man to his weight, and say that a man whose height is 1.80 m is expected to weigh 80 kg (exact model) or we could say that, with 90% confidence, a man whose height is 1.80 m weighs between 75 and 85 kg (interval model).

There has been some research concerning interval neural networks [11] and fuzzy models [12], [22], but as far as we know, interval genetic programming (GP) models have not yet been studied. GP replaces neural networks and fuzzy models when we need a readable algebraic expression that describes a system's input–output behavior. Interval GP will play the same role in interval models because it will generate an expression that relates confidence intervals for the output to values of the input. We illustrate this fact with numerical and graphical examples, and also perform a comparison among fuzzy, neural, classical, GP, GA–P, and interval GA–P models.

Manuscript received July 10, 1997; revised April 24, 1998 and September 8, 1998. This work was supported by Hidroeléctrica del Cantábrico under Contract CN-96-055-B1 with the University of Oviedo.

The author was on leave at General Electric CRD, Schenectady, NY USA. He is with the Department of Computer Science, Oviedo University, Gijon, Asturias, Spain (e-mail: luciano@lsi.uniovi.es).

Publisher Item Identifier S 1089-778X(00)01383-7.

II. Interval Symbolic Regression

Regression techniques are intended to find an adequate expression for a function g so that, given a variable Y which depends on the value of a variable X, $g(X)$ is a good approximation to Y. In practice, this means that we know N pairs (X_i, Y_i), and we search for a function g such that the mean-square error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N}[Y_i - g(X_i)]^2$$

is minimum [17].

Where the expression of g is not known, the techniques used to solve the problem are known as *symbolic regression*. Genetic programming is one of the methods that has been successfully applied to solve these kinds of problems [14].

Note that symbolic regression methods find *exact* estimates for Y given X, but sometimes we require a confidence interval that covers the unknown value of Y with a given probability. We extend the applicability of GP methods to this case: a function that relates a vector input X to a set Γ that contains Y with probability β is needed. This mapping is a *set-valued* function. We make its definition depend on m interval parameters, and use interval arithmetic [13]. For example, one function of the class we are interested in is $\Gamma(x) = k \otimes x$, which depends on one interval parameter $k = [1, 2]$, and it is defined as $\Gamma(x) = [x, 2x]$ where "$\otimes$" means "product" in interval arithmetic.

There are many different mappings Γ for which the probability that $Y \in \Gamma(X)$ is greater than or equal to our confidence degree β, but in general, we will be interested in the mapping that makes the expected length of $\Gamma(x)$ as short as possible. Let us define a pair of functions g^+ and g^- such that $\Gamma(x) = [g^-(x), g^+(x)]$. The numerical problem that we need to solve is as follows. Given a value of probability ϵ near zero and *two* independent samples that contain N pairs (X_i, Y_i) and M pairs (X'_j, Y'_j), respectively, find $g^-(x)$ and $g^+(x)$ such that

$$\frac{1}{N}\sum_{i=1}^{N}(g^+(X_i) - g^-(X_i))$$

is minimum and the proportion of elements of the *first* sample for which $g^-(X_i) \le Y_i \le g^+(X_i)$ is greater than $1-\epsilon$. Observe that the value of β is not known, but it can be estimated. $1-\epsilon$ is not a good estimation of the confidence level because g^+ and g^- are correlated with the sample, but we can estimate β with the second sample, and say that $\hat{\beta}$ is the proportion of elements of this last sample for which $g^-(X'_i) \le Y'_i \le g^+(X'_i)$.

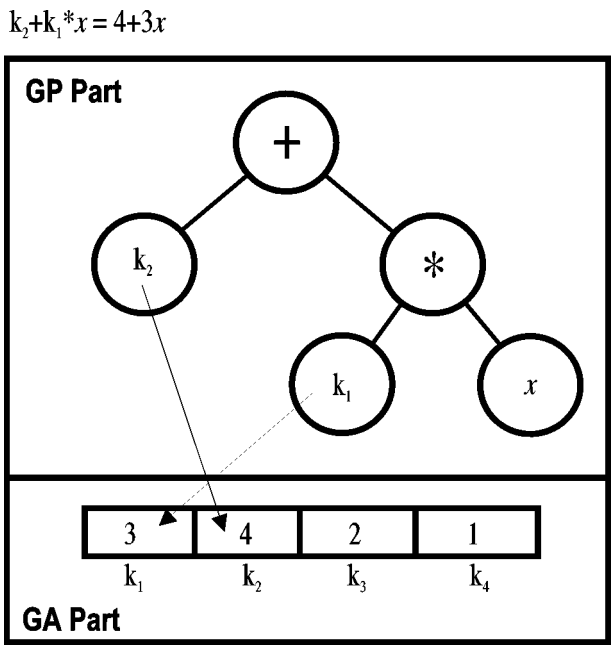

Fig. 1. Member of a population in a GA–P algorithm. An individual consists of an expression and a string of parameters. Internal nodes are operators, and leaf nodes are either coefficients (k_1, k_2, k_3, k_4) or input variables (x). The crossover can occur in the linear part or in the tree part; thus, expressions and numeric coefficients evolve concurrently.

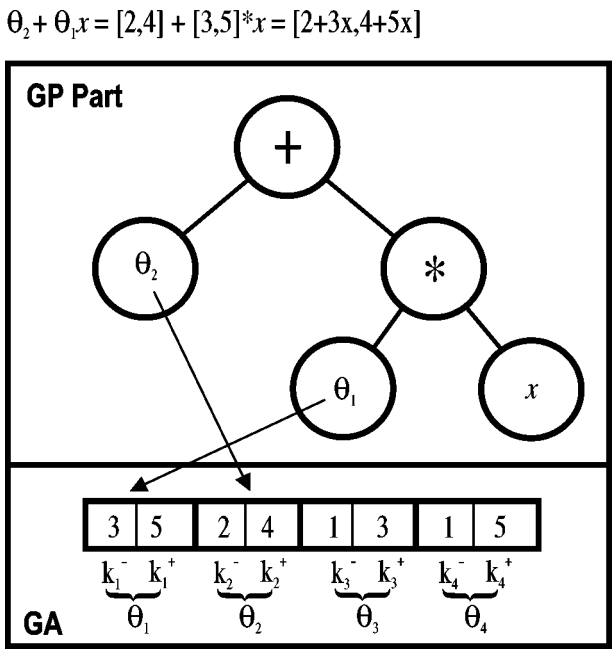

Fig. 2. Member of a population in an interval GA–P algorithm. Internal nodes are interval arithmetic operators, and leaf nodes are either intervals ($[k_1^-, k_1^+]$, etc.) or input variables (x).

III. Numerical Optimization Method: GA–P Algorithms

GA–P algorithms are an evolutionary computation method, a hybrid between genetic algorithms (GA) and genetic programming, optimized to perform symbolic regression. A complete description of the GA–P method can be found in [8]. Briefly, each element of the population comprises a chain of parameters (the GA part which governs coefficients) and a tree-based description of a function (the GP part which controls their expression) which depends on the encoded parameters. Internal nodes of the tree are mathematical operators, and terminal nodes can be instances of the independent variable or pointers to the parameters. When evaluating an individual, terminal nodes are replaced by the value of the parameter to which they point, or by the value of the input variable they represent (see Fig. 1).

The two basic operations we use to obtain new members of the population are crossover and mutation. These operations are independently performed over both parts, and we use the operations defined in [8].

A. Modifications to GA–P

The multivalued mapping Γ proposed in the last section will be defined in terms of a function h_θ and m interval parameters $[k_i^-, k_i^+]$, making

$$[g^-(x), g^+(x)] = \{t \in \mathbb{R} | t = h_\theta(x), \qquad \theta \in [k_1^-, k_1^+] \times \cdots \times [k_m^-, k_m^+]\}.$$

This definition will be clearer with the following example. Let $h_\theta(x) = \theta_1 x + \theta_2$. h_θ depends on two *numerical* parameters θ_1 and θ_2; let $\theta_1 \in [k_1^-, k_1^+]$ and $\theta_2 \in [k_2^-, k_2^+]$ with $k_1^- = 2$, $k_1^+ = 4$, $k_2^- = 1$, $k_2^+ = 5$. Then, $g^-(x) = 2x + 1$ and $g^+(x) = 4x + 5$. This can also be expressed if we write $\Gamma(x) = [2, 4] \otimes x \oplus [1, 5] = [2x + 1, 4x + 5]$. This last definition (that involves the interval operators "$\otimes$" and "$\oplus$") is very convenient for extending the GA–P method. We simply allow intervals to be terminal nodes, and extend the set of operations so that it includes the interval arithmetic counterparts of every algebraic operator used in GA–P operations.

The differences between GA–P and interval GA–P are as follows.

1) *Constant Part Codification:* We need to represent m intervals, each depending on two numbers. We have not codified k_i^- and k_i^+, but rather k_i^- and $k_i^+ - k_i^-$. We order the values so that k_i^- is next to k_i^+.
2) *Fitness Function:* There is an important difference between usual symbolic regression problems and this one: the function we wish to minimize does not depend on evaluations of the expression in the set of examples, but instead, on the separation between g^+ and g^- in the points of that set, and g^+ and g^- were defined by means of h. To find the value of $g^+(x)$, we find the maximum of h inside the allowed range for its parameters:

$$g^+(x) = \max_{\mathbb{R}}\{h_\theta(x), \qquad \theta \in [k_1^-, k_1^+] \times \cdots \times [k_m^-, k_m^+]\}$$

and the same could be said of g^-. Fortunately, this minimum and maximum can be calculated if we use interval arithmetic operations to implement the evaluation of the pair g^+, g^-, as noted before.
3) *Set of Operators:* The proposed representation is based on the use of interval arithmetic to perform all operations involved in the expression part (see Fig. 2). That is, we codify the function in a tree whose terminal nodes point to interval parameters $[k_i^-, k_i^+]$ or input variables. The internal nodes represent interval operations that can be

unary or binary. Any unary interval operation O_u depends on an operation o_u as follows:

$$O_u(A) = \{x \in \mathbb{R} | x = o_u(t) \text{ and } t \in A\}.$$

For example, $\sin(A) = \{x \in \mathbb{R} | x = \sin(t) \text{ and } t \in A\}$. The definition of the binary operations is similar:

$$O_b(A, B) = \{x \in \mathbb{R} | x = o_b(t, u) \text{ and } t \in A, u \in B\}$$

where $A, B \in I(\mathbb{R})$, o_a: $\mathbb{R} \longrightarrow \mathbb{R}$, and o_b: $\mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$. The evaluation of the expression for an input value (that can be a point or an interval) will be an interval.

B. Tuning the Algorithm

A number of decisions must be made when applying GP that involve the probabilities of the operators, the initial population composition, and the definition of the fitness function. For a GA–P algorithm, all of these decisions, as well as others related to the GA, must be made. We tabulate the precise selections made in our particular case in the following section. For now, we only remark on two differences between this method and the original GA–P.

1) *Local Optimization:* Local optimization is a technique used in genetic algorithms to increase the precision of the solution [19], [20]. GP-based search was also combined with a local parameter-tuning mechanism employing statistical search [10]. We use local optimization when crossover is performed in the chain of parameters, and the offspring scores in the best half of the current population. Simplex search is used instead of more advanced methods because we do not know the gradient of the fitness function. Numerical results are given in the next section.
2) *Fitness Function:* When solving a constrained optimization problem with GA's or GP, we can formulate a fitness function that involves penalties [18] or we can apply multiobjective methods [7]. We chose the first method. Suppose that we wish to find an interval model for which the probability of Y being covered by the output is higher than a value β, $P(g^-(X) < Y < g^+(X)) > \beta$. We begin by selecting two values ϵ_1 and ϵ_2 such that $\epsilon_1 > 1 - \beta > \epsilon_2$. These values serve to classify models in three classes: those that cover a fraction higher than $1 - \epsilon_2$ of the examples, those that cover less than $1 - \epsilon_1$, and those that cover a fraction between $1 - \epsilon_1$ and $1 - \epsilon_2$. We treat these three classes separately and define the following.

   a) The fitness of an individual covering a fraction of examples less than $1 - \epsilon_1$ to be proportional to the MSE of a model formed by replacing all interval parameters and interval inputs in the model by their midpoints (we choose a proportionality constant so that the first item holds).
   b) The fitness of an individual covering more than $(1-\epsilon_2)N$ examples to be its mean amplitude ($g^+ - g^-$).
   c) The fitness of an individual that covers more than $1-\epsilon_1$ and less than $1-\epsilon_2$ to be a weighted average of its MSE as defined in item b) and its mean amplitude as defined in item c).

IV. Example

The characteristics of the method will be made clear with a graphical example. We have generated 100 pairs that form a sawtooth with three line segments to which we have added 5% of uniformly distributed random noise. GA–P, neural, fuzzy, and interval GA–P model outputs are shown in Fig. 3. The GA–P and neural net are exact models, and their output is represented by the dotted line in the figures. Further, the output of a fuzzy model is a fuzzy set, which is usually defuzzified to give an exact output too. We could also use the α cuts of these fuzzy sets, but it may happen that they are not intervals nor include the dependent variable because the output of a fuzzy model does not necessarily have a convex membership function.

In contrast, we can estimate the probability for which the output of interval GA–P contains the dependent variable, and we can also build an exact model from it, if we replace the interval parameters with values. This exact model can offer benefits over the original GA–P model when there are outliers in the data (see Fig. 4).

In that figure, we altered two points in the dataset. It is easy to detect and purge these points in a one-dimensional problem like this, but it becomes more difficult to detect them when modeling high-dimensional data with small sample sizes. Methods for handling data sets with outliers are called *robust* [9].

The effect of these two points in GA–P, neural, and fuzzy models is similar. The neighborhood of the outliers (input values between 40–50) is incorrectly modeled because the weight of these two points is very high in the global squared error. Interval GA–P models are robust, and they will downweight, and in some cases, reject erroneous data. The computational effort is, however, higher. It took 1 h on a personal computer (Pentium II, 300 MHz) for interval GA–P to finish. The neural net only needed 30 s, and the fuzzy model [22] and the GA–P about 10 min each.

V. Practical Application

The problem described here is solved by different methods, and the obtained solutions are compared.

A. Preliminaries

Some hundreds of generators, including thermal, nuclear, and hydroelectric plants, are connected to the Spanish electrical network, along with many different clients, ranking from small domestic consumers to important aluminum or steel producers.

Energy is transported from suppliers to clients by means of electrical lines of different capacities. We can distinguish between high-voltage lines (100–400 kV), medium-voltage lines

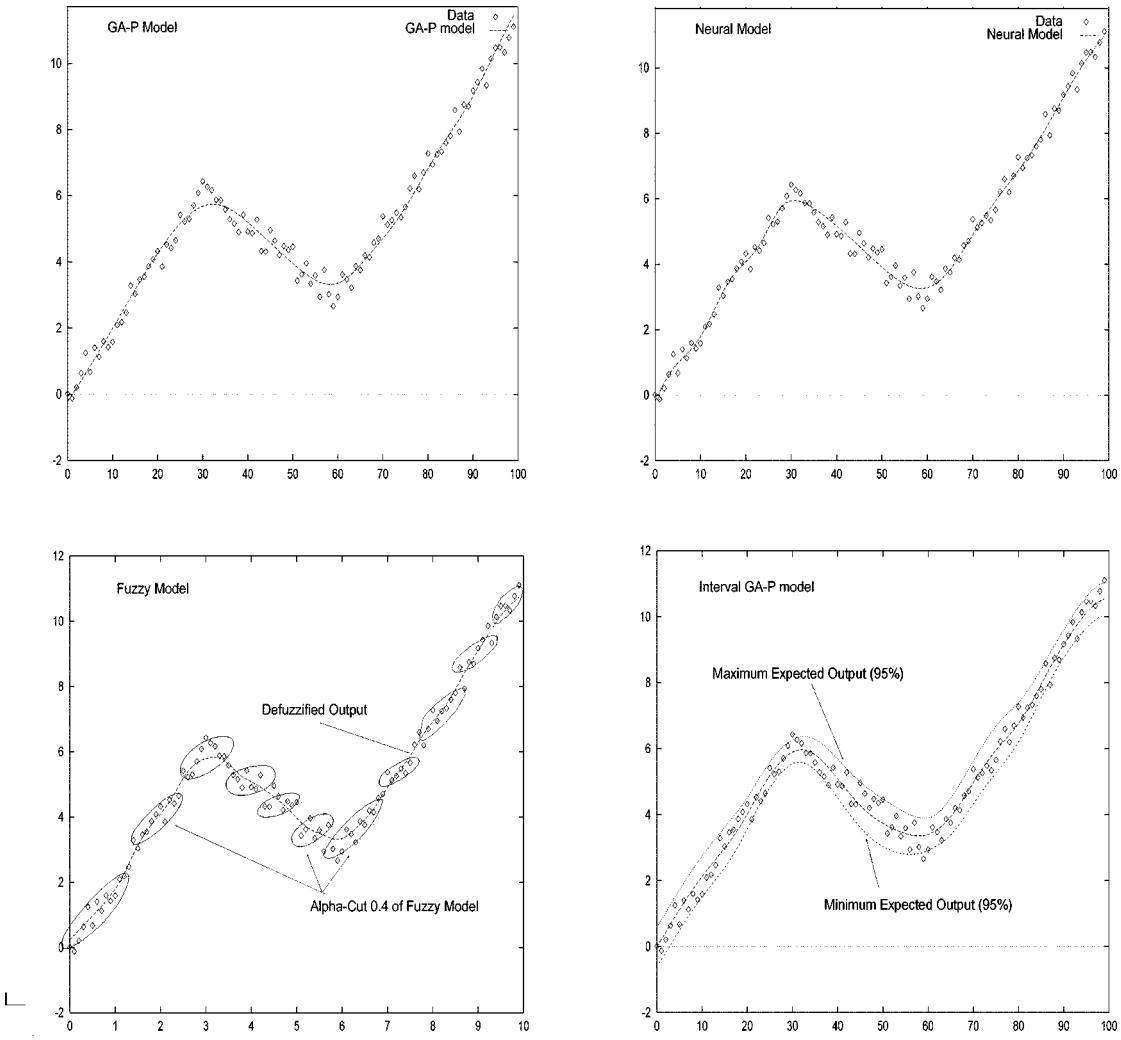

Fig. 3. GA–P, neural net, fuzzy, and interval GA–P applied to a simulated function. 5% uniform noise was added. The output of the fuzzy model cannot be interpreted as a confidence band.

(10–100 kV), and low-voltage lines (<10 kV). The government expropriated high-voltage lines some years ago because of antitrust regulations, and formed a company that manages the high-voltage network. Former owners kept medium- and low-voltage lines. This regulation forced the distribution companies (the same companies that own the power generation plants in Spain) to buy and sell the electrical energy from the market instead of using their own generation plants, thus creating separate markets for the generation and distribution of electrical energy.

On the other hand, the payment that the distribution companies receive depends not only on the transported energy, but on indicators like the actual length of medium- and low-voltage lines being maintained by the company. It is obvious that the definition of these indicators is of great economic importance. Recently, some companies argued that the increased population of some cities prevented the installation of the optimal distribution network, and that there are many lengthy obsolete networks. These lengthy networks are being favored over the optimal ones by a payment that depends on the actual length of line, so their owners tend not to modernize their installations. A new payment structure based on new models other than the actual length of line was proposed.

To discuss this new structure, it was necessary to develop models of the optimal and actual length of electrical line installed in cities and villages. In fact, companies have maps of their lines, but they do not know precisely the kilometers of line they maintain. They instead use models that relate the characteristics of cities with their expected line length. The task is to develop new models that could be contrasted with the old ones, which requires exact models and *confidence intervals* for our predictions (i.e., one numerical value plus best and worst case within a 95% confidence). GA–P methods were chosen because we can select both the maximum complexity of the final expression and the desired algebraic operators. Black-box models (i.e., neural networks, high degree polynomials) were not admitted, and neither were classical or fuzzy-rule-based models. However, we did use them to contrast the performance of GA–P methods.

B. Characteristics of the Model

For the sake of clarity, we restrict attention to one simple model that relates the low-voltage line length in Spanish villages

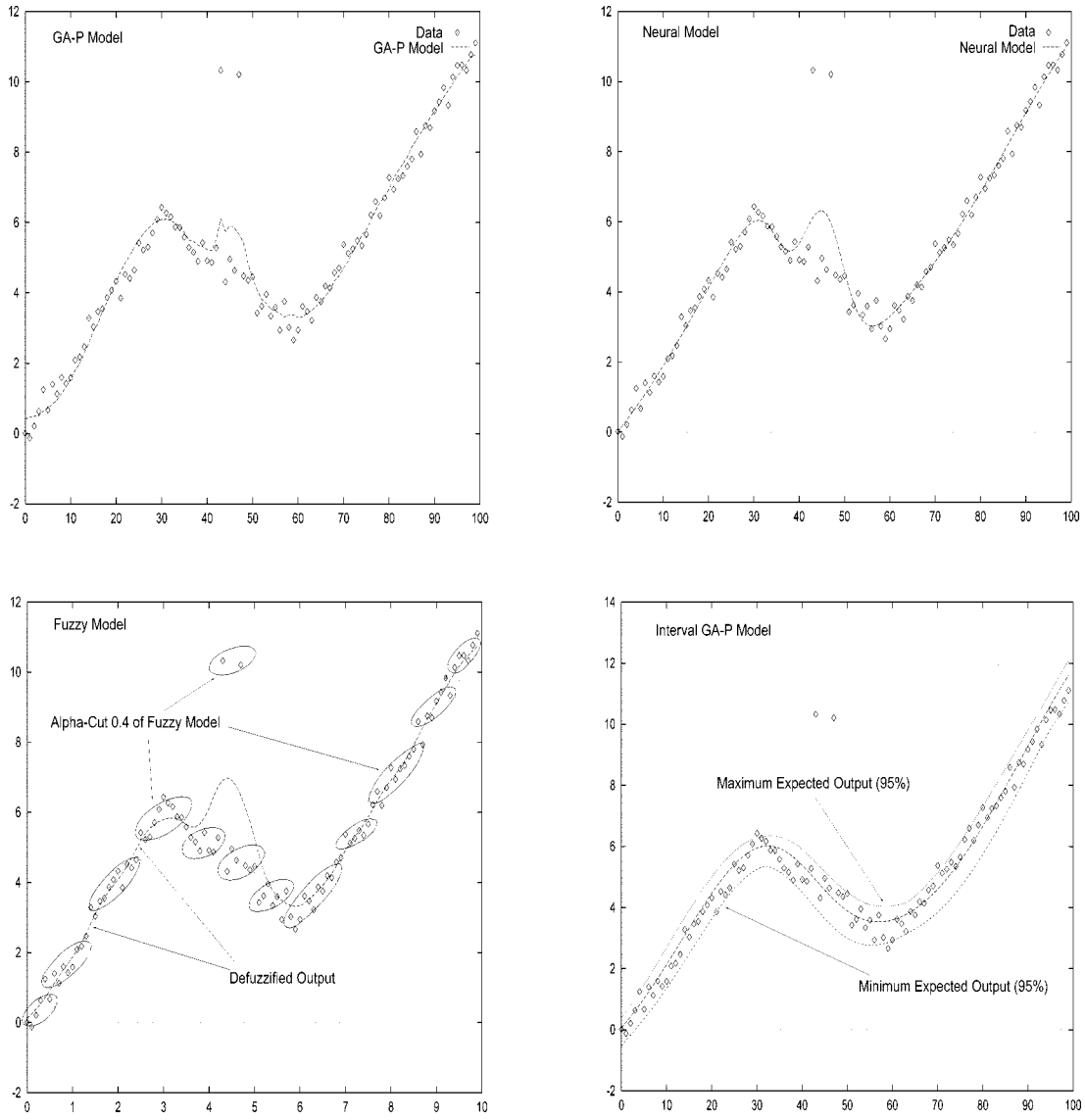

Fig. 4. Simulated dataset with outliers. Two points were altered to show that nonrobust methods (GA–P, neural net, fuzzy) can produce wrong estimates in the neighborhood of the outliers.

TABLE I

Symbol	Meaning
A_i	Number of clients in village i
R_i	Radius of village i
n	Number of villages in the sample
l_i	Line length, village i
$\tilde{l}_i$	Estimation of l_i
s_i	Number of sectors in village i

to their number of inhabitants and their area. The data provided included the measured line length, the number of inhabitants, and the mean of the distances from the transformation center to the three furthest clients in a sample of 491 villages. The objective was to relate the line length to the other variables, first by classical methods, and later by applying GA–P methods. We will also determine a range of values of length for every pair of client radius so that approximately 95% of the sample data were within the interval. The nomenclature in Table I was adopted.

C. Application of Classical Methods

Low-voltage electrical networks are arranged in sectors in the villages that are being studied. A main line passes near all clients inside the village, and clients are connected to these main lines by small segments (see Fig. 5).

To construct a simple model, we have assumed the following.

- Village i comprises s_i sectors. All sectors in the same village cover the same angle $2\theta_i$. Main lines depart from the center of the village.
- The density of clients is constant inside every sector.
- All sectors in a village have the same radius R_i, and contain a main line of length R_i, and as many branches as consumers.

Assuming that customers are uniformly distributed, we can approximate the total length of line by multiplying the mean distance between a customer and the nearest main line by the

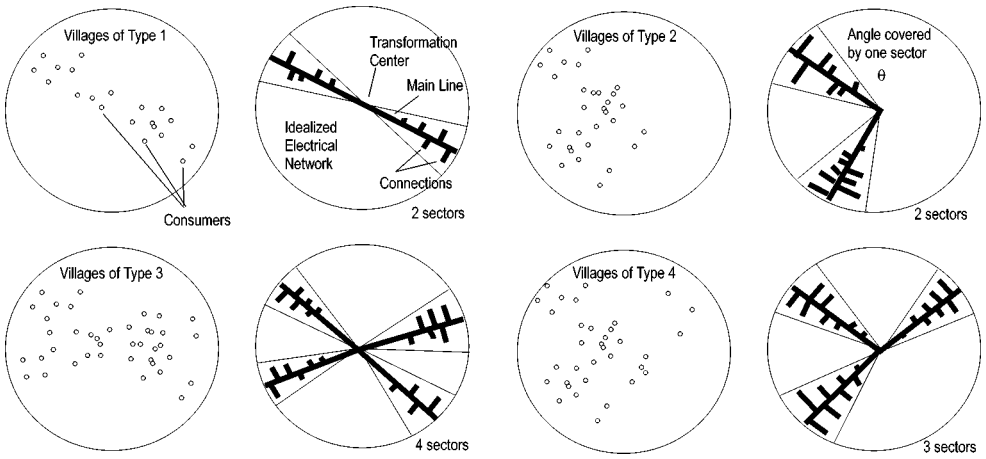

Fig. 5. Idealized models of electrical networks in small rural villages. Main lines depart from the center of the village. Consumers are connected by small segments to the nearest main line. Villages have two, three, or four main lines.

number of inhabitants, and adding to the result the length of the main lines. Let d_i be the mean length of a branch in village i. It follows that

$$d_i = \frac{2(1-\cos\theta_i)}{3\theta_i} R_i$$

so the cable length is

$$\tilde{l}_i = s_i \left(R_i + \frac{A_i}{s_i} d_i \right) = s_i R_i + A_i \frac{2(1-\cos\theta_i)}{3\theta_i} R_i$$

and simplifying,

$$\tilde{l}_i / R_i = s_i + k(\theta_i) A_i.$$

If the angles θ_i and the numbers s_i were sufficiently similar, we could regard them as constants, and estimate them by the parameters $\overline{\theta}_i = \theta$ and $\overline{s}_i = s$ of a least squares linear regression

$$\tilde{l}_i / R_i = s + k(\theta) A_i$$

to a set of pairs $(x, y) = (A_i, l_i/R_i)$.

We can get a better fit by allowing a certain dependence among the number of sectors, their amplitudes, and the number of inhabitants. This can be done by dividing the sample into classes, and then fitting a linear model to each one (multilinear model), or we can perform a change of variables followed by a linear regression on the transformed data. Both cases were studied, and the best fit was obtained with the exponential model

$$\frac{\tilde{l}_i}{R_i} = k_1 A_i^{k_2}.$$

We also studied linear regression in two variables ($\tilde{l}_i = k_1 A_i + k_2 R_i + k_3$), quadratic, and cubic models. Numerical results are given in Table III.

D. Neural and Fuzzy Models

It is interesting to compare black-box methods to GA–P. We used a multilayer perceptron (with one hidden layer fully connected to input and output layers) and two fuzzy models, one of them with Mamdani-type rules with approximative modeling [4], and the other with Takagi–Sugeno–Kang (TSK) rules and approximative modeling [24]. The conjugate gradient method was used for the multilayer perceptron [21], the Wang–Mendel method followed by a genetic tuning [5] that reduces the number of rules was applied for the Mamdani-type fuzzy model, and pure genetic learning was used for the TSK model [5]. The number of neurons in the hidden layer of the neural network was determined by trial and error to minimize the error on the test data [15].

Descriptive fuzzy models are designed to linguistically describe the behavior of the model, but the high number of rules

TABLE II
TABLEAU FOR THE LENGTH-OF-LINE PROBLEM

Parameter	Decision
Population size	100
Iterations	1000 to 5000 (steady state)
Parent selection	Tournament
GA Part encoding	Floating point
GA Crossover operator	Two points
GP Crossover operation	Standard
GA Cross. probability	0.9
GP Cross. prob. internal nodes	0.9
GP Cross. prob. leaves	0.1
GA Mutation (individuals)	0.01
GP Mutation (individuals)	0.01
Expressional part	20 to 500 nodes
Maximum number of parameters	10
Enrichment initial population	best 100 of 1000 evaluations
Edition probability	0
Encapsulation probability	0
Permutation probability	0
Decimation	No
ADFs maximum	0
Local GA optimization	Nelder and Mead's simplex

TABLE III
COMPARISON OF FITNESS; VARIOUS METHODS

Method	MSE Training	MSE Test	Complexity
Linear,2 variables	369	443	9 nodes, 3 par.
Linear,1 variable	365	443	7 nodes, 2 par.
Multilinear, 1 variable, 2 classes	338	458	17 nodes, 6 par.
Exponential	342	426	7 nodes, 2 par.
2th order polynomial	332	393	22 nodes, 6 par.
3rd order polynomial	318	941	53 nodes, 10 par.
3 layer perceptron 2-25-1	312	391	102 par.
W. M. fuzzy model	262	610	22 rules
TSK fuzzy model	272	462	34 rules
GP	336	420	17 nodes, 1 ERC
GA-P	325	399	18 nodes, 1 par.
Interval GA-P 98%(outliers not purged)	341	424	8 nodes, 2 par.
Interval GA-P 98%(outliers purged)	306	385	8 nodes, 2 par.
Interval GA-P 95%(outliers purged)	258	250	11 nodes, 1 par.

(22) obtained in this case after the genetic tuning makes the descriptive fuzzy model much more difficult to interpret than the expression obtained with GA–P.

E. GP, GA–P, and Interval GA–P

We applied GA–P algorithms to search for a formula that is comparable in complexity with the exponential model (see Section V-C), while adjusting better to the real data. We restricted the search to expressions that can be codified in a tree with no more than 20 nodes, and depending on no more than 10 parameters. Binary operations were sum, difference, product, ratio, and power. The unary operation was the square root.

We conducted two numerical test benches. First, we compared the best model obtained with GP, GA–P, and interval GA–P to fuzzy, neural, and classical models; then we made 25 series of 25 runs each, starting GP, GA–P, and interval GA–P from different, random populations. We ran the algorithm with and without local optimization, and limiting the maximum number of nodes to values ranging from 20 to 500. The parameters of the learning processes are shown in Table II. The best expressions obtained were compared to the best models obtained with fuzzy rule learning, neural networks, and classical methods in Table III.

Mean-square error values are labeled "MSE training" and "MSE test." We define "mean-square error" as

$$\frac{1}{N}\sum_{i=1}^{N}(\tilde{l}_i - l_i)^2.$$

The column "Complexity" contains the number of parameters and nodes in the tree that codifies the model (for example, the model $\tilde{l} = k_1 A_i + k_2 R_i + k_3$ depends on three parameters k_1, k_2, and k_3, and can be codified in a tree comprising nine nodes). For neural models, the number of weights, and for fuzzy models, the number of rules are shown. The models with the best numerical adjustment were obtained with the locally optimized GA–P and interval GA–P models.

TABLE IV
AVERAGE FITNESS. STEADY-STATE GP

	MSE Train	MSE Test	Length	Max. Nodes
50%	337645	425471	19.83	20
90%	352990	433724	19.72	20
50%	325714	418703	50	50
90%	331828	420690	49.95	50
50%	322483	419228	96.2	100
90%	329953	421326	97.9	100
50%	315140	418712	180.2	200
90%	323582	422878	186.6	200
50%	310145	418448	439.9	500
90%	320592	420503	425	500

TABLE V
AVERAGE FITNESS. GA–P WITHOUT LOCAL OPTIMIZATION

	MSE Train	MSE Test	Length	Max. Nodes
50%	334405	420034	19.8	20
90%	341077	427570	19.5	20
50%	323066	418091	49	50
90%	329900	422577	49.5	50
50%	318585	425032	98	100
90%	326592	424276	96	100
50%	316199	423244	164	200
90%	324052	424986	167	200
50%	304587	424314	365.3	500
90%	315218	425058	370.1	500

TABLE VI
AVERAGE RESULTS. GA–P WITH LOCAL OPTIMIZATION

	MSE Train	MSE Test	Length	Max. Nodes
50%	323832	413292	19.5	20
90%	327486	411652	19.6	20
50%	305305	412509	49.6	50
90%	313307	407980	49.2	50
50%	292196	420247	93.4	100
90%	299424	433541	95	100
50%	299766	418911	183.8	200
90%	307737	418483	181	200
50%	289325	411468	413.3	500
90%	299603	411117	388.5	500

TABLE VII
AVERAGE RESULTS. INTERVAL GA–P WITHOUT LOCAL OPTIMIZATION

	MSE Train	MSE Test	Length	Covering	Max. Nodes
50%	310194	298069	19.7	96	20
90%	417114	364497	17.8	96	20
50%	322455	296747	36.2	96	50
90%	361595	323370	36	95.7	50
50%	288588	288060	67.7	95.8	100
90%	340739	306115	68.7	95.4	100
50%	324389	284100	135.6	95.9	200
90%	372395	320332	119.5	95.7	200
50%	293898	268389	256.2	95.4	500
90%	322930	293589	204.8	95.6	500

TABLE VIII
AVERAGE RESULTS. INTERVAL GA–P WITH LOCAL OPTIMIZATION

	MSE Train	MSE Test	Length	Covering	Max. Nodes
50%	283780	273850	18.6	94.3	20
90%	319651	288816	17.1	94.7	20
50%	269643	249326	42.2	93.1	50
90%	291323	273217	41.5	93.7	50
50%	261992	222634	78.2	92.3	100
90%	282614	243503	74.9	92.9	100
50%	283195	274803	71.7	93.9	200
90%	311411	288721	81.5	94.1	200
50%	274910	266219	86.9	93.5	500
90%	296791	273058	95.9	93.8	500

This kind of comparison poses a problem with the interval GA–P method, which should not be evaluated on the basis of MSE since it is not an exact model. We could speak of maximum and minimum MSE, but this is not a good indicator of the model's performance because it is easy to obtain a minimum MSE of 0, and the maximum MSE can be much higher than the average MSE for exact models in the rank of intervalar parameters. We decided to build exact models from interval models by replacing every interval parameter by its midpoint value, and computing these punctual models only at the points that were not discarded as outliers. With this information, we can determine if the deviation produced by the outliers is relevant to the MSE obtained, as shown graphically in Figs. 3 and 4. The results indicated that if 2% of points were discardable, a model comparable to a neural network in performance, but not more complex than a linear one was obtained. If 5% of points can be discarded, the result widely outperformed the neural network. Of course, a new neural network could be trained over these 95% points, and a new comparison could be made, but interval GA–P automatically selects the 5% points that most influence the error. Doing this selection by hand is a difficult task.

The best model obtained by GP was

$$R_i + A_i + \sqrt{1.06\left(\sqrt{R_i} + R_i + R_i\right)\sqrt{R_i}A_i} + R_i.$$

The best GA–P model was

$$R_i f\left(f\left(f\left(\sqrt{R_i} + k_0\right)\right)\right)$$

where $f(x) = (A_i/x)^{1/2}$, and the best model obtained with interval GA–P (2% of outliers allowed) was

$$\tilde{l} \in \left\{ l \,\middle|\, l = k_1 R_i k_2^{(A_i^{0.0625})}, \quad k_1 \in [13.56,\ 53.47], \quad k_2 \in [0.058,\ 0.064] \right\};$$

hence, the exact model we constructed was

$$\tilde{l} = 38.5 R_i \cdot 0.061^{(A_i^{0.0625})}.$$

This model depends on two parameters, and can be codified in a tree of eight nodes. We also have included the results of evaluating this exact model over the complete training and testing sets [row "Interval GA–P 98% (outliers not purged)"].

We intended to obtain a model with $\beta = 0.98$, and we chose the values $\epsilon_1 = 0.05$ and $\epsilon_2 = 0.01$. The estimation of β is $\hat{\beta} = 0.975$; that is, we estimate that the probability for the range of the interval GA–P model to include the true value is greater than 0.975.

With the next set of experiments, we tried to determine how many times the algorithm fails to find a good solution, and we also study the effect of the local optimization on a GA–P algorithm. If we compare the results of Tables IV and V, we observe that standard GA–P is slightly better in this problem, but the result is not statistically significant ($\alpha = 0.05$). The figures indicate the average MSE over the best 50% and the best 90% experiments in every series, and the mean number of nodes of the best individuals obtained. The locally optimized version of GA–P performs somewhat better (see Table VI).

The canonical GP did not produce good results when we implemented interval models, and we do not include them here. GA–P results are in Tables VII and VIII. Again, the influence of the local optimization is noticeable. All of the experiments were driven with $\epsilon_1 = 0.1$ and $\epsilon_2 = 0.05$, and the estimated coverings are also shown in these tables. Observe that interval GA–P has a different behavior from those of GP and the original

GA–P when we increase the number of nodes. Interval GA–P (both optimized and without optimization) produces smaller expressions, and the trees in the expressional parts of the best individuals do not tend to grow up to the maximum allowed limit.

VI. Concluding Remarks

In this work, we used genetic programming for finding expressions of variability bands for functions. The bands are based on sets of examples, and they cover the exact model of the data with a given probability.

Canonical GP was not used to find these bands, but a hybrid method between GA and GP, called GA–P, was adapted to use interval arithmetic. The method provides the mathematical expression of an interval model, and this expression can be converted into an exact model. Interval GA–P is also a robust method that is less influenced by outliers than canonical GP, fuzzy models, and neural nets.

Acknowledgment

The authors would like to thank I. Couso, from Oviedo University, for numerous comments in connection with the theoretical development of the method. O. Cordón and F. Herrera, from Granada University, applied Wang–Mendel and TSK fuzzy modeling methods to the dataset, and made helpful comments on the paper as well.

References

[1] D. Adler, "Genetic algorithms and simulated annealing: A marriage proposal," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, 1993, pp. 1104–1109.

[2] P. Billingsley, *Probability and Measure*. New York: Wiley Ser. in Prob. and Math. Statist., 1995.

[3] I. Couso, "La envolvente probabilística. Definición y propiedades," trabajo de investigación, Departamento de Estadística e I.O y D.M., Univ. Oviedo, Spain, 1997.

[4] O. Cordón, "A general study on genetic fuzzy systems," in *Genetic Algorithms in Engineering and Computer Science*, J. Periaux, G. Winter, M. Galn, and P. Cuesta, Eds. New York: Wiley, 1995, pp. 33–57.

[5] ——, "Una metodologiá para el diseño automático de sistemas basados en reglas difusas mediante algoritmos evolutivos," doctoral dissertation, Departamento de Ciencias de la Computación e Inteligencia Artificial, Univ. Granada, Spain, 1997.

[6] A. P. Dempster, "Upper and lower probabilities generated by a random closed interval," *Ann. Math. Statist.*, vol. 38, pp. 325–339, 1967.

[7] C. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, pp. 1–16, 1995.

[8] L. Howard and D. D'Angelo, "The GA–P: A genetic algorithm and genetic programming hybrid," *IEEE Expert*, pp. 11–15, June 1995.

[9] W. Härdle, *Applied Nonparametric Regression*. Cambridge, U.K.: Cambridge Univ. Press, 1989.

[10] H. Iba, T. Sato, and H. De Garis, "System identification approach to genetic programming," in *Proc. 1st IEEE Conf. Evol. Comput.*, vol. 1, 1994, pp. 401–406.

[11] H. Ishibuchi, H. Tanaka, and H. Okada, "An architecture of neural networks with interval weights and its application to fuzzy regression analysis," *Fuzzy Sets Syst.*, vol. 57, pp. 27–39, 1993.

[12] L. Kacprzyk and M. Fedrizzi, Eds., *Fuzzy Regression Analysis*. Warsaw, Poland: Omnitech, 1984.

[13] A. Kaufmann and M. Gupta, Eds., *Introduction to Fuzzy Arithmetic*. New York: Van Nostrand Reinhold, 1984.

[14] J. Koza, *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: M.I.T. Press, 1992.

[15] C. G. Looney, *Pattern Recognition Using Neural Networks. Theory and Algorithms for Engineers and Scientists*. Oxford, U.K.: Oxford Univ. Press, 1997.

[16] G. Matheron, *Random Sets and Integral Geometry*. New York: Wiley Ser. Prob. and Math. Statist., 1975.

[17] B. Lindgren, *Statistical Theory*. London, U.K.: Chapman & Hall, 1993.

[18] Z. Michaliewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1992.

[19] J. M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways," in *Proc. 1st IEEE Conf. Evol. Comput.*, vol. 1, 1994, pp. 312–317.

[20] J. M. Renders and S. P. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Trans. Syst., Man, Cybern.—Part B: Cybern.*, vol. 26, Apr. 1996.

[21] R. Rojas, *Neural Networks—A Systematic Introduction*. Berlin, Germany: Springer-Verlag, 1996.

[22] L. Sánchez, "A random sets based method for identifying fuzzy models," *Fuzzy Sets Syst.*, vol. 3, pp. 343–354, 1998.

[23] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.

[24] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, 1985.