Multiobjective Evolutionary Search of Difference Equations-based Models for Understanding Chaotic Systems

Luciano Sánchez and José R. Villar

Abstract In control engineering, it is well known that many physical processes exhibit a chaotic component. In point of fact, it is also assumed that conventional modeling procedures disregard it, as stochastic noise, beside nonlinear universal approximators (like neural networks, fuzzy rule-based or genetic programming-based models,) can capture the chaotic nature of the process.

In this chapter we will show that this is not always true. Despite the nonlinear capabilities of the universal approximators, these methods optimize the one step prediction of the model. This is not the most adequate objective function for a chaotic model, because there may exist many different nonchaotic processes that have near zero prediction error for such an horizon. The learning process will surely converge to one of them. Unless we include in the objective function some terms that depend on the properties on the reconstructed attractor, we may end up with a non chaotic model. Therefore, we propose to follow a multiobjective approach to model chaotic processes, and we also detail how to apply either genetic algorithms or simulated annealing to obtain a difference equations-based model.

Keywords: Nonlinear approximation; Chaotic signals; Genetic algorithms; Simulated annealing

1 Introduction

When modeling complex processes, there is always a balance between the transparency of the model and its accuracy. Chaotic signals are not an exception to this rule: we expect a technique that produces a *black box* from data [12,20,25,32,34,41]

Luciano Sánchez and José R. Villar
Computer Science Department, Universidad de Oviedo, Edificio Departamental 1, Campus de Viesques, 33213 Gijon (Spain), Tel.: +34 985182597; fax: +34 985 181 986., e-mail: villarjose@uniovi.es

R. Lowen and A. Verschoren (eds.), *Foundations of Generic Optimization, Volume 2: Applications of Fuzzy Control, Genetic Algorithms and Neural Networks*, 181–201.

to produce more accurate results than other procedures that also gains insight into the block structure of the system. The best representative of this last kind of model (that we will name *white boxes*, understandable, or transparent models) is arguably a set of difference equations. A difference equations-based model allows the user not only to predict the output of the process, but to know the dynamics of the model and ultimately to design a control system for it. Nevertheless, obtaining an appropriate set of equations from data is a problem that cannot be regarded as solved.

Many of the most recent approaches to obtain understandable descriptions of chaotic systems are based on evolutionary techniques. In particular, the use of tree-based codifications allows us to define a simultaneous search in both the different families of models, and the parameters that define a model within one of these families. Since we want to discover the structure of the set of equations (i.e., a consistent subset of state variables and the dependences between them) and also the numerical values of the coefficients in these equations, it is convenient for us to combine an evolutionary search with a tree-based representation of the model, as it was done, among others, in [3, 4, 11, 16, 40, 49].

Some of the latest algorithms are able to obtain difference equations, but there is work yet to be done. Many evolutionary modeling methods minimize the discrepancies between the data and the one-step prediction of the model, and do not take into account the dynamic behavior of the model [41, 47]. As we will show later in this paper, should we search for a model on the basis on the lowest one-step prediction error, we have high chances of finding a non-chaotic model. In that case, the obtained equations would be meaningless. The use of greater prediction horizons is not always feasible, though. Being chaotic systems, we can find large deviations between the recursive evaluation of the model and the training data.

In the following sections we will solve this problem by enforcing an additional constraint: the value of the largest Lyapunov exponent of our model has to match that value estimated from our train data. The largest Lyapunov exponent is a measure of the amount of chaos in the signal [24, 25, 48], and the difference between the maximum Lyapunov exponents of two models also gives us a measure of similarity between the complexities of their dynamics [17, 47]. Accordingly, we propose to extend the aforesaid balance between transparency and accuracy to a new triplet transparency/accuracy/dynamic. We define a multiobjective problem, designed to minimize the square error and the complexity of the model, while restricting the search to those models whose largest Lyapunov exponents are similar to the estimated value from the time series we want to analyze. Since the evaluation of the Lyapunov exponents is very time costly, we also propose to use our own custom evolutionary algorithm, that combine a tree-based codification with a population-based, multiobjective extension of the Simulated Annealing. The algorithm that we propose in this paper is able to find a set of difference equations that reproduces the dynamics of a given chaotic time series, and improves the results of modern multiobjective evolutionary algorithms like NSGA-2 [9, 10] when the number of evaluations of the objective function is limited.

The organization of this paper is as follows: in Section 2, we make a brief bibliographic analysis of transparent models of chaotic systems, detailing the unsolved

problems. In Section 3, we describe our own proposal. Experiments and results are shown in Section 4, and the paper finishes with the concluding remarks and the future work.

2 Evolutionary Transparent Modeling of Chaotic Systems

Genetic algorithms, genetic programming and evolutionary programming techniques have been applied to identify and control nonlinear and chaotic systems. The reader can refer to [30, 43], where genetic algorithms are compared against different identification techniques, or review the results in [5, 23, 38, 44]. The control problem is less studied, but there also exist works like [38], where a genetic algorithm was used to find the optimal control signals sequence in a chaotic cutting process.

There are different views of the concept of "transparent model" for chaos. For instance, linguistic fuzzy rules were combined with genetic algorithms in [5] and in [23], and applied to analyze chaotic time series. Wavelet coefficients are also considered to provide a certain degree of interpretability, as they were used in [44], where genetic algorithms were applied to select wavelet threshold parameters in an exchange-rate forecasting problem. Another approaches for nonlinear modeling use polynomial models, as can be seen in [12, 39]. Many other different, problem specific, analytical modeling approaches had been developed. For example, in [1], evolutionary algorithms were used to propose nonlinear models for a satellite based ocean forecasting system. In [11], evolutionary computing was used for extracting mathematical models, and this proposal was analyzed with three different applications. Lastly, in [16] genetic programming was used to find difference equations models of nonlinear processes, as we propose in this paper.

In all of the preceding methods, the fitness of a individual is based only in instantaneous error measures, thus not all the available information about the dynamic of the process is used. As we mentioned in the introduction, this means that the learning algorithm will surely converge to a nonchaotic model. In this case, the usefulness of a transparent model is limited. A different approach was presented in [14], where evolutionary computing was used for obtaining models for chaotic time series, using the error of the recurrent outcome of the model, which is a measure of its dynamical behavior. The recurrent outcomes of a chaotic model are very much different under small differences of the initial state, and then this measure of error has to be taken with care, but our own approach shares properties with this method. In the following section, we will propose to evaluate the dynamical properties of a candidate model by mean of its recursive evaluation, however not through the error in the trajectory, but estimating the higher Lyapunov exponent of the time series formed by this recursive prediction and including it in a multiobjective fitness function.

Multiobjective techniques have been previously used to develop models for nonlinear and chaotic systems. In some of our own previous works [13], we have pro-

posed to use a linear combination of the quadratic error and the largest Lyapunov exponent for the fitness function, and have optimized it by means of a genetic algorithm. In [12, 39] a Pareto-based approach is used instead of scalar functions, in combination with the MOGA algorithm described in [15]. There are some different Pareto-based multiobjective strategies that could also be applied for the same problem, as can be seen in [6]. Later in this paper, we will evaluate a more recent approach, the NSGA-II algorithm [9, 10].

Given the computational cost of evaluating the Lyapunov exponents of a model, and the potentially large size of some individuals, we are mostly interested in algorithms that need a low number of iterations and small population sizes. It is widely admitted that genetic algorithms are the best choice for this matter. As a matter of fact, these algorithms have become an standard in all kind of multiobjective problems [50]. However, in our opinion, the experimentation that support this assert was intended to solve problems based in a linear genotype, and it is not immediate to extrapolate all of their conclusions to tree-based representations. In previous works [42], we have combined a simulated annealing (SA) global search with a grammar-tree-based codification, in the context of the learning of fuzzy rules. An strategy so simple as keeping only one individual, and repeatedly mutating it, admitting or discarding the result according to a probability decreasing with time and distance, was able to improve the results of the GA. With this result in mind, in this paper we will extend our own algorithm to multiobjective problems, and propose a new population-based, multiobjective SA search (MOSA) able to elicit a set of nondominated solutions. In the following sections we will show that the genetic search (the NSGA-II algorithm,) while equally efficient in the long term, can be improved in this specific problem by a Simulated Annealing-based search in both accuracy and memory usage.

Interesting enough to mention, a pure Pareto-based MOSA has not been previously defined, to our knowledge. The most recent approaches weight the different criteria into a scalar function [19, 31, 45]. Otherwise, in [8] it was proposed to use the dominance to decide the evolution of the simulated annealing. That approach was also used in [18], where fuzzy numbers and uncertainty in dominance is managed to decide if an individual is better than other or not. Similarly, in [35, 36], Pareto dominance is studied to decide how the multiobjective simulated annealing evolves. But, in all of these cases, an aggregated function of objectives still is used to evaluate each individual. A different approach to Pareto-based MOSA, nearer to ours, is presented in [2]. In that work, a comparison of a Pareto-based evolutionary algorithm and a population-based simulated annealing with dominance control approach is presented. In each simulated annealing iteration, a new individual is obtained by means of an heuristic, and it is included in the population if there is nondominance relation with the current individual. If the new one dominates the current, then it becomes the current one. In the opposite case, then it is accepted with temperature dependent probability. Observe that, even in this last case, it is required that either an individual dominates or is dominated by another. This is done, again, weighting the different objectives into a scalar function and therefore the algorithm does not homogeneously sample the Pareto front.

In the next sections we will propose a different algorithm that does not pose this problem.

3 Operators Used in the Evolutionary Searches

The experimental analysis that we will show later compares the NSGA-II and the MOSA algorithms, both sharing the same representation and operators. Our SA search will be based in the mutation operator, in turn based in the genetic crossover [42].

In this section we will state, for both search schemes, the representation of an individual, its validation procedure, how to generate an individual at random, how to evaluate it, the crossover and the mutation operators. In the next section we will describe the pseudocode of the algorithms.

3.1 Representation of an Individual

We will build the input data from a time series, given an embedding dimension n, thus the training set contains the sampled values of n system state variables $x_k^1, \ldots, x_k^n$, at times $k = 1, 2, \ldots$. We wish to obtain a set of $m \leq n$ difference equation-based models, with the structure that follows:

$$x_{k+1}^i = f_i(x_k^1, \ldots, x_k^n) \quad i \in \{1, \ldots, n\}. \tag{1}$$

One of these state variables will be identified as the output of the system. It is assumed that $x_{k+1} = x^k$ for all those variables without an equation assigned.

The phenotype of an individual is, therefore, a list of m valid equations. We will define the concept "valid equation" by means of the the grammar shown in Figure 1.

S $\mapsto$ Structure Parameters
Structure $\mapsto$ ArithOp $\vee$ NonLinearOp $\vee$ DelayOp
Parameters $\mapsto$ Variable $\vee$ Constant
Variable $\mapsto$ System signal
Constant $\mapsto \Re$
ArithOp $\mapsto$ (+ Exp, Exp) $\vee$ (− Exp, Exp) $\vee$ (* Exp, Exp)
NonLinearOp $\mapsto$ (G [LC, UC] $\rightarrow$ OC, Exp) $\vee$ (Dz [LC, UC] $\rightarrow$ OC, Exp)
DelayOp $\mapsto$ (Ret delay Variable)
LC $\mapsto$ Constant
UC $\mapsto$ Constant
OC $\mapsto$ Constant

Fig. 1 Grammar defining a valid equation. "G" means "gain", "Dz" means "dead zone". There are some restrictions in the value of the constants that are also enforced: $LC < UC$, and all constants are bounded

createModel
`needs:` *list of system signals, id. of the output-signal, experiment parameters*
`produce:` *a random set of signals, including the one designed as system output, and a randomly generated equation for every one of them*

```
for each signal s in the list of system signals
  if (s = output-signal) or (random() < threshold) then
    signals.push(s)
for each s in signals
  equations.push(
    createRandomEquation(signals, experiment parameters)
  )
return { signals, equations }
```

Fig. 2 Simplified pseudocode of the random generation of a model using the PTC2 algorithm. The function `createRandomEquation` takes into account constrains like the maximum height of a tree, the probabilities of each type of node and the grammar shown in Figure 1

The genotype will be the syntactic tree of a valid chain in this grammar. Each node of this tree will encode the name of the production rule that originated each subtree. This information will be used later to define a typed crossover.

It can be observed that each equation comprises two parts, associated to the productions "Structure" and "Parameters". The first production defines which operations are valid to define the functions f_i, and the second one is a list of numerical parameters, on which these last functions depend. Following [26], the nonlinear elements in the definition of f_i are selected from the usual catalog of building blocks in control engineering. We have restricted ourselves to the blocks "gain with saturation" and "dead zone".

3.2 Random Generation of Genotypes

The PTC2 algorithm (see Figure 2) is used to generate random trees [27, 28]. This algorithm allows to specify the maximum number of nodes, the maximum height, the types of nodes, and the probability distribution for each tree height and the probability distribution of each type of node, conditioned to our grammar.

3.3 Genetic Crossover and Mutation

Our crossover operator has two different expressions, to which we will refer as *parametric* and *structural*. The parametric crossover takes place between the parts of the individuals that derive from the production rule "Parameters", and the structural crossover involves the parts originated in the production "Structure". Leaving apart the differences in the grammar, the same operators proposed in [42] were used:

- To perform the parametric crossover we select one of the nodes derived from the production *Constant* in each one of the trees, and modify both values with an extended intermediate crossover [33].
- To carry out the structural crossover of two individuals, a random node of the first parent is selected. The subtree rooted in this node is to be interchanged with another one in the second parent. A list of valid nodes of this last parent is produced. That list of valid nodes not only has to take into account the syntactic restrictions of the grammar, but there are also semantic constrains: the height of the offspring must not be higher than the limit, and the individuals must not have more than one equation for each one of the state variables. If the list is empty, the procedure is repeated with a different node in the first parent. Once we have a nonempty list, one of its elements is randomly chosen and interchanged with the former one.

In previous works [42], we have proposed to implement the macromutation in the SA algorithm by means of a subtree crossover with a randomly generated individual [21, 37]. In our MOSA implementation we will use this technique: crossover with a random individual followed by a selection at random from the offspring. The same mutation operator will also be used in our implementation of the NSGA-II algorithm.

3.4 Fitness Function

The fitness function comprises a pair of numbers: the mean error of the one-step prediction of the model, and the absolute difference between the largest Lyapunov exponents of the model and the training data. Different procedures have been proposed to compare this kind of compound values [7]. We will use a Pareto multiobjective evaluation, and guide the search towards obtaining a set of nondominated individuals. In the most general case, it is said that an individual x dominates to another individual y ($x \prec y$,) if all the F_j components of the fitness vector F verify $F_j(x) \leq F_j(y)$, and $\exists t \mid F_t(x) < F_t(y)$. However, we are not interested in the whole Pareto front, because models with a high prediction error are not of practical interest. We will discard all models whose one-step prediction error is higher than the variance of the time series, no matter their Lyapunov value.

The estimation of the one-step prediction error is immediate. Unfortunately the same cannot be said about estimating the largest Lyapunov exponent of a model. It will be computed, as mentioned, from the time series produced by the recursive evaluation of the model since a given initial state, discarding the first samples of the recursive evaluation, so we are certain that the trajectory is in the attractor.

Some different numerical algorithms were evaluated by us. Our first choice was the well-known Wolf algorithm [48], that we had already used in previous works. Unfortunately, the number of samples that this algorithm needs is rather high; this, in combination with the large number of iterations and the population sizes needed to obtain good models with multiobjective genetic algorithms makes the whole

identification procedure impractical (more than one week in a modern scientific workstation.) There exist other algorithms, in particular those of Rosenstein and Kantz [22, 29], which need lower sample sizes than Wolf's; we have successfully used a combination of the Rosenstein algorithm and our own heuristic estimation of the point where the slope of the curves time vs divergence changes. The use of the Rosenstein algorithm, in combination with the MOSA algorithm that we will detail in the next section, reduces the computation time from days to hours. However, the best results in both accuracy and computational effort have been obtained by an estimation based on the equations of the model and the principal axes of expansion, as discussed in [46]: we follow the divergence of two close trajectories. One of them is retained for reference. The other one is repeatedly renormalized so that the distance between both is kept small. The maximum Lyapunov exponent is then estimated by the average value of the logarithm of the quotients between the starting distance between the trajectories and the distances after one step, before renormalizing.

4 Detailed Description of the MOSA Algorithm

4.1 Outline of the Algorithm

The pseudocode of the Multi-Objective Simulated Annealing-Programming (MOSA) algorithm is shown in Figure 3. This algorithm is based in a variable sized population of search points. At each iteration, all the search points are mutated and their respective fitness evaluated. The comparison between the fitness of the mutated individual and that of its corresponding search point can produce three different results:

1. The new individual dominates the current search point.
2. The new individual is dominated by the search point.
3. Neither of them dominates the other.

The strategy of MOSA for these three cases is as follows:

1. If the mutated individual dominates the current search point, it replaces its parent in an intermediate population.
2. If the mutated individual is dominated, then a random decision is made between storing the current search point or the mutated one. Observe that, being a SA search, the probability of admitting the mutated point depends on the cooling pattern and decreases with both the distance between the fitness values and the time. The distance between the fitness values is explained in the next subsection.
3. Otherwise, the size of the intermediate population is increased, and the mutated model initiates a new search path.

Once all the individuals in the population have been mutated and the preceding decisions have been taken, the intermediate population is sampled by means of the `selection` operator to form the following generation. Aside from the population, note that an elitist set of nondominated solutions is also kept; this set is the current sample of the Pareto front and eventually will be the output of the algorithm.

4.2 The Distance Operator

To implement the simulated annealing we need to generate new individuals in the neighborhood of the current one. The chances of a new individual being admitted depend on the distance between the current and the new individual. When vector based individuals are used, the euclidean distance can be used, but this is not longer true with tree-based representations. In previous works [42], we postulated the use of an *edition distance* between trees as the number of edition operations (add, remove or replace a node) needed to transform the current into the new model. Besides, in the same paper we also checked that there was possible that proximal individuals had a very different evaluation of the fitness, and the same happens here. Therefore, we have chosen to implement a distance in the fitness landscape (the supremum of the distances in all the criteria) instead of an edition distance in the genotypical space.

```
Select initial and final temperatures: T_0, T_1
Select the cooling factor : C
Select a starting model: x_0
Initialize the population of search paths: X = {x_0}
Initialize the set of elites (sample of Pareto front): P = {x_0}
T ← T_0
while T ≤ T_1
    // Initialize intermediate population X'
    X' ← X
    for path ← 1 to size(X)
        x ← mutation(X_path)
        if x ≺ X_path then
            // The search point is updated
            X'_path ← x;
        else if X_path ≺ x then
            // The search point might be updated
            if rnd() < exp(-distance(X_path, x)/T) then X'_path ← x
        else
            // A new search path is generated
            X' ← X ∪ {x}
        end if
    end for
    // The set of nondominated values up to this moment is updated
    P ← nondominated models of P ∪ X'
    // If needed, the size of the set of paths is adjusted
    X ← selection(X')
    T ← T · C
end while
```

Fig. 3 Pseudocode of the MOSA algorithm

4.3 The Selection Operator

The size of the intermediate population can be twice as high as the the current population size, in the worst case. To control the maximum population size, all the dominated values and duplicated search points are removed at each iteration.

Our selection operator is a variation of that used in the NSGA-II algorithm [9,10]. In the first place, the set of nondominated search points is computed by pairwise comparisons of all individuals in X'. Observe that we do not need to use fast sorting algorithms to compute this set, because the size of X in our experimentations ranges between 10 and 25 individuals and performing 25^2 comparisons is much faster than evaluating once the fitness value. Secondly,

- If the size of the set of nondominated search points is small enough, this set is the new population.
- If its size must be further reduced, we sort the individuals in this last set by means of the same crowding distance defined in the NSGA-II algorithm, and choose them in inverse order of distance.

4.4 Example of a MOSA Evolution

In Figure 4 a typical example of the evolution of the MOSA algorithm is shown. The problem being solved is taken from [15]. Since this problem consists in finding two real values, we have codified each individual by means of a vector instead of a tree, and used a extended intermediate crossover with a randomly generated chain to mutate them, but otherwise the search scheme of MOSA was followed. It can be observed that all the solutions are in the Pareto front after 100 iterations, and it is also shown how the population size evolves.

5 Experiment and Results

5.1 Dynamic Behavior of Universal Approximators

As we have mentioned in the introduction, a good error in the one-period prediction error does not necessarily imply that the dynamic behavior of the system has been captured. Suppose we intend to model a chaotic time series with an universal model, a neural network, say. We first choose an embedding dimension d and convert the time series into a training set. Each instance of this set has d inputs (the last d values of the series) and one output (the next value in the series). We expect that, if the embedding dimension is high enough, the neural network will capture the dynamics of the model that generated the series.

The problem with this reasoning is, there are many different networks able to approximate the former training set without error. Most of them do not correspond with chaotic models. For instance, observe the one-step prediction errors of the networks in the table that follows. They all are near zero, and apparently the models are very precise, although some of them have too low an embedding dimension. However, in Figure 5 we have plotted the step responses of these models. Observe that all of them are stable systems, with a punctual attractor. None of the nets captured the chaotic nature of the signal.

Multilayer Perceptron		
Embedding dimension	Nodes in each layer	Err
1	1 - 3 - 1	0.000972
2	2 - 5 - 1	0.000034
3	3 - 10 - 1	0.000004
4	4 - 10 - 1	0.000009

If we use a transparent model instead, the same can happen. In Figure 6 a Genetic Algorithm was used, with the same representation and operators described in the text but an scalar fitness (based only on the one-step error.) We have trained it with data from the Henon map. The learned model is not chaotic, though, as pictured in the center part of the figure. Lastly, in the lower part of the same figure the step response of a model learned by the MOSA algorithm is shown. This is a chaotic model, and in the next section we will also show some examples of reconstructed attractors. It is remarked that the one-step error of either model the MOSA and the GA are close to zero.

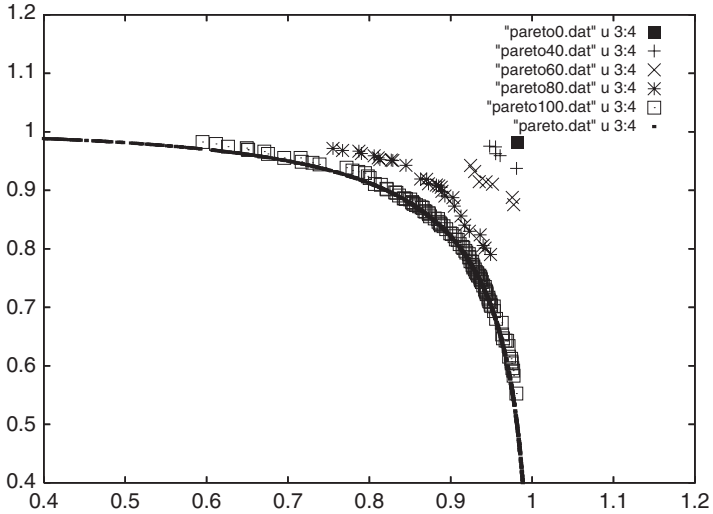

Fig. 4 Evolution of the population in the MOSA algorithm, for a two-criteria problem taken from [15]

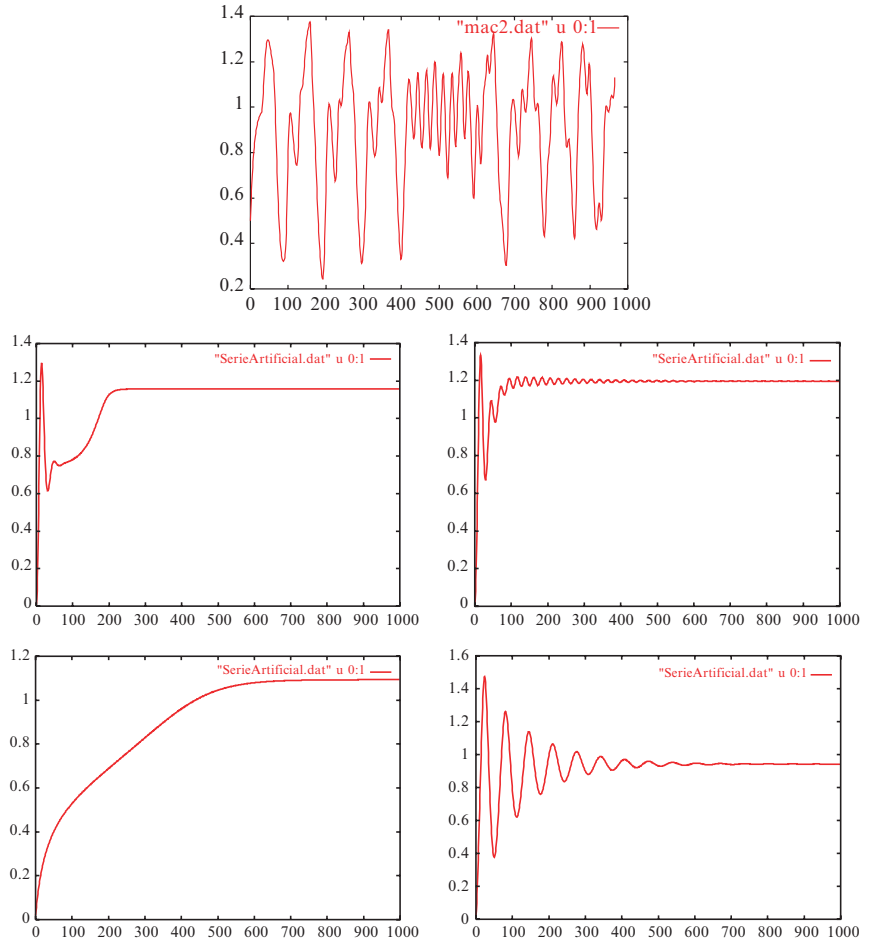

Fig. 5 Chaotic time series, and multilayer perceptrons with one hidden layer, trained to minimize the one-step error. Upper part: Chaotic signal (train data). Central part: Step responses of the neural networks 1-3-1 and 2-5-1. Lower part: Networks 3-10-1 and 4-10-1. Despite the low values in the objective funcion shown in the text, none of the neural nets is a chaotic model

5.2 Benchmark Problems

In this section we will compare the results of MOSA and NSGA-II over some benchmark problems. The NSGA-II is an implementation of the Pareto-based multiobjective genetic algorithm detailed in [9, 10], which is currently assumed to be among the best available implementations of such kind of algorithms.

The results will be shown with two different methodologies, graphical and statistical. The graphical (qualitative) approach serves to identify the differences between the combined Pareto fronts after a certain number of repetitions of each exper-

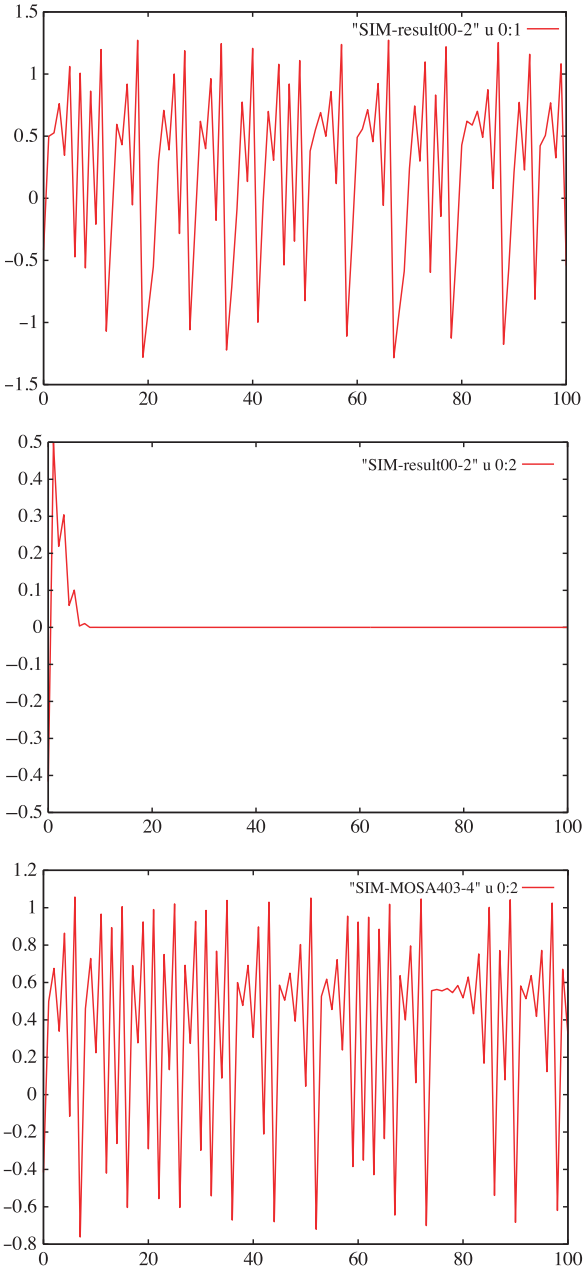

Fig. 6 Graphical analysis of experimental results, Henon map. Upper part: Train data. Center: Typical recursive evaluation of a transparent model obtained by an evolutionary algorithm when the maximum Lyapunov exponent is *not* included in the fitness function: In this case, the optimization has converged to a stable model (Lyapunov exponent < 0.) Bottom: Step response of one of the models found with the procedures mentioned in this paper

iment. The statistical (quantitative) comparison of the results of multiobjective Evolutionary Algorithms is a current research field. There exist many different measures of the degree to a Pareto front improves the results of another one, but it is acknowledged that there are problems derived from the stochastic nature of evolutionary algorithms that are still unsolved [50–52]. We propose to use an statistical test about the probability of either algorithm dominates the other, based in the binary ε-indicator described in [51]. Both the qualitative and quantitative analysis will be explained in the sections that follow.

5.2.1 Experimental Setup

The parameters of the operators used in the experimentation are shown in the following tableaux:

NSGA2			
Parameter	Value	Parameter	Value
Structural crossover	0.5	Parametric crossover	0.5
Mutation	0.01	Embedding dimension	2
Population size	100	Evaluations of fitness	5000
Constants minimum value	-5	Constants maximum value	5

MOSA			
Parameter	Value	Parameter	Value
Initial temperature	1.00	Cooling Factor	0.999
Structural mutation	0.5	Parametric mutation	0.5
-		Embedding dimension	2
Maximum population size	10	Evaluations of fitness	5000
Constants minimum value	-5	Constants maximum value	5

The learning time is roughly proportional to the number of times that we estimate the greater Lyapunov exponent of a model, and both algorithms are allowed to evaluate 5,000 times this function. Since this estimation is not performed when the one-step error is higher than the variance of the time series, this is equivalent to $50 \approx 100$ generations of the NSGA-II algorithm. The parameters defining the random initialization of the individuals are as follows:

Parameter	Value
Maximum number of nodes in equations	10
Prob. of number of nodes/equation, 1 - 10	.05 .12 .11 .15 .15 .15 .11 .08 .05 .03
Maximum height	7
Height probability distribution, 1 - 7	.05 .4 .3 .15 .05 .025 .025
node types	$+$; $-$; *; G; Dz;
Node type probability distribution	.21 .21 .21 .21 .09 .07

Each experiment was repeated 10 times. The time series used for training and validation have size 1,000. The chaotic systems that have have been used are the

Logistic and the Henon maps, with the set of parameters shown in the equations that follow:

$$\text{Logistic map:} \quad x_{k+1} = 4.0 * x_k * (1 - x_k) \tag{2}$$

$$\text{Henon map:} \quad \begin{cases} x_{k+1} = 0.3y_k + 1 - 1.4x_k^2 \\ y_{k+1} = x_n \end{cases} \tag{3}$$

5.2.2 Commented Graphical Results

The graphical results are displayed in Figures 7 and 8. In both cases, we have obtained the combined Pareto front (upper left part) after 10 repetitions of either algorithm. This combined Pareto front is formed by selecting all the nondominated individuals of the 10 runs. In the upper right part, all the elements of the 10 Pareto fronts of each algorithm are displayed together, in the same graph. By last, in the right lower part of the figures we have displayed a couple of reconstructed attractors

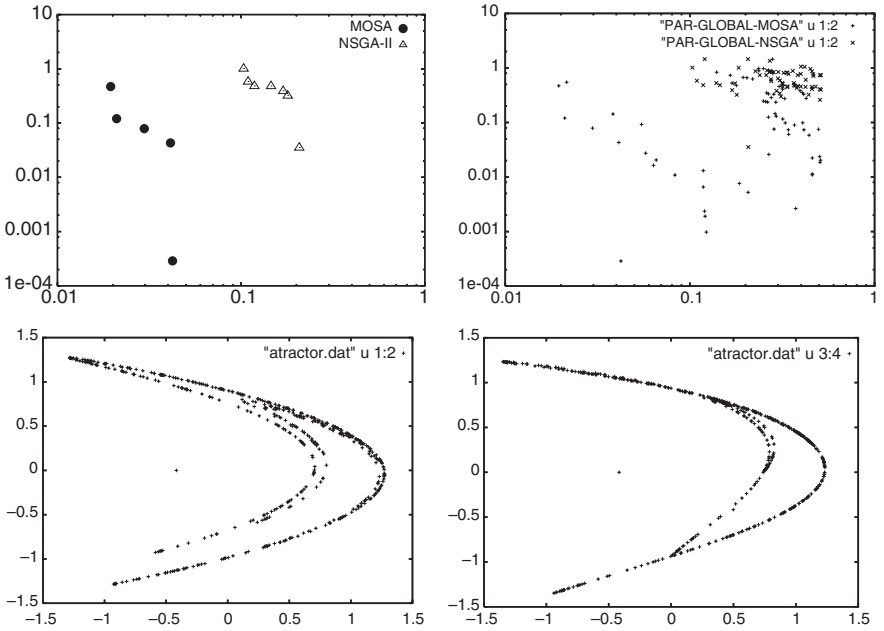

Fig. 7 Graphical analysis of experimental results, Henon map. Upper part, left: Combined Pareto front of ten repetitions of the algorithms NSGA-II (triangles) and MOSA (circles). All the models in the Pareto front of the NSGA-II algorithm are dominated by at least one element in the Pareto front of the MOSA. A logarithmic scale is used, to enhance the differences. The vertical axe represents the error in the Lyapunov exponent, the horizontal one is the one-step error. Upper part, right: combined cloud of the 10 Pareto fronts of both experiments, from which the Pareto fronts were calculated. Lower part, left: Attractor of the Henon map. Lower part, right: Attractor of one of the models induced by the MOSA method

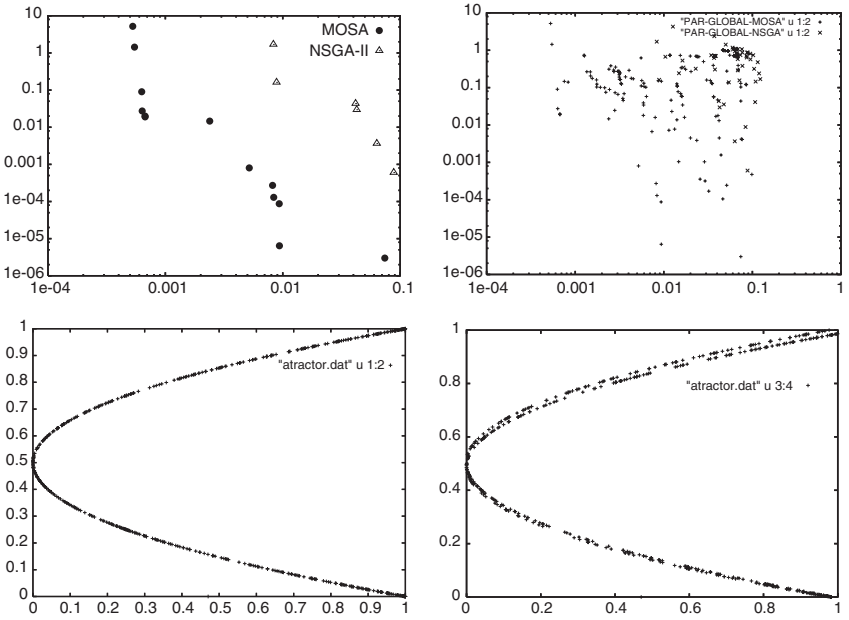

Fig. 8 Graphical analysis of experimental results, Logistic map. Upper part, left: Combined Pareto front of ten repetitions of the algorithms NSGA-II (triangles) and MOSA (circles). All but one of the models in the Pareto front of the NSGA-II algorithm are dominated by at least one element in the Pareto front of the MOSA. A logarithmic scale is used, to enhance the differences. The vertical axe represents the error in the Lyapunov exponent, the horizontal one is the one-step error. Upper part, right: combined cloud of the 10 Pareto fronts of both experiments, from which the Pareto fronts were calculated. Lower part, left: Attractor of the Henon map. Lower part, right: Attractor of one of the models induced by the MOSA method

that show the similarities between the dynamic behavior of the models and that of the original system (left part.)

As there is a clear difference between the combined fronts (all of the points in the NSGA-II front are dominated by those of the MOSA) this is not so in this second graph, since some of the executions of MOSA were dominated by NSGA-II and vice versa. The extent to which, in average, one algorithm is better than the other, will be studied in the next section.

5.2.3 Numerical Comparison

There are functions (unary indicators) that can convert a Pareto front into a representative value. It is possible to compare sets of these representative values with the same methodology used in scalar evolutionary algorithms, i.e., a statistical test able to discard that the expected errors are the same. However, some studies have shown that these unary indicators are not able to show all the dominance relations

that can happen between Pareto fronts [52]. Therefore, to assess the average improvement between one algorithm and the other, we will used a method based on a binary indicator, namely, the binary ε-indicator defined in [51].

Two different definitions of this last indicator are possible: the standard (multiplicative) I_ε and the additive indicator $I_{\varepsilon+}$. Given two fronts A and B, if $I_\varepsilon(A,B) < 1$ and $I_\varepsilon(A,B) > 1$, or if $I_{\varepsilon+}(A,B) < 0$ and $I_{\varepsilon+}(A,B) > 0$, we can state that A dominates B. The values of these indicators for our combined Pareto fronts follow:

	I_ε(MOSA,NSGA)	I_ε(NSGA,MOSA)
Henon	0.25	122.98
Logistic	0.13	201.483

	$I_{\varepsilon+}$(MOSA,NSGA)	$I_{\varepsilon+}$(NSGA,MOSA)
Henon	-0.04	0.19
Logistic	$-6 \cdot 10^{-4}$	0.04

In both cases, we can conclude that combined MOSA results dominate that of NSGA-II. These results are not conclusive, though, since one exceptionally good result of either algorithm could be responsible of the dominance of the combined Pareto front. Therefore, we propose to apply the ε-indicator to perform a full set of comparisons between all pairs of fronts, and to calculate the fraction of times each instance of the algorithm A dominates one of the instances of the algorithm B, and vice versa.

Our methodology is as follows: Let $p_A(B)$ be 1 if A dominates B (i.e. when $I_\varepsilon(A,B) > 1$ and $I_\varepsilon(B,A) < 1$), 0 otherwise. Given 10 repetitions $B_1, \ldots, B_{10}$ of an algorithm B, let

$$P_A(B) = \frac{1}{10} \sum_{i=1}^{10} p_A(B_i). \tag{4}$$

and, given another 10 repetitions $A_1, \ldots, A_{10}$ of an algorithm A, let

$$\mathbf{P}_A(B) = (P_{A_1}(B), P_{A_2}(B), \ldots, P_{A_{10}}(B)). \tag{5}$$

The vector $\mathbf{P}_A(B)$ can be seen as a sample of a random variable: the fraction of times that the output of the algorithm A dominates the algorithm B. If the expectation of $\mathbf{P}_A(B)$ is greater that the expectation of $\mathbf{P}_B(A)$, then we can state that the algorithm A is better than the algorithm B, since it is easier that results of the former improve that of the latter than the opposite.

Therefore, to know whether there is a significant difference between the two algorithms we can use a statistical test to discard that the expectations of $\mathbf{P}_A(B)$ and $\mathbf{P}_B(A)$ are the same. Since the distributions of none of them were compatible with the Gaussian distribution, we have used a Wilcoxon test (null hypothesis $E(\mathbf{P}_A(B)) = E(\mathbf{P}_B(A))$, alternate hypothesis $E(\mathbf{P}_A(B)) > E(\mathbf{P}_B(A))$.) The resulting p-values are shown in the following table:

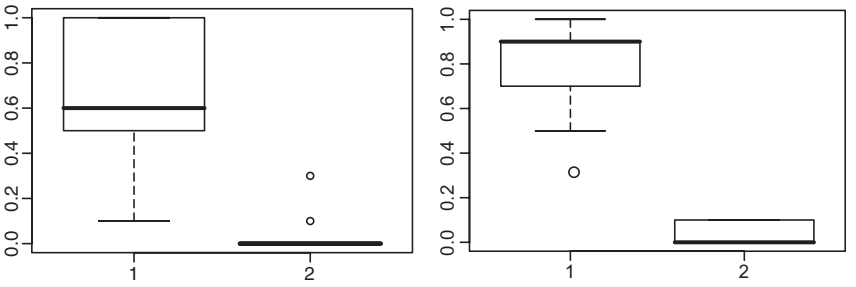

Fig. 9 Boxplots of (1) $\mathbf{P}_{\text{MOSA}}(\text{NSGA-II})$ and (2) $\mathbf{P}_{\text{NSGA-II}}(\text{MOSA})$ for the Henon map (left part) and Logistic map (right part.) This graph shows that the probability of *MOSA* improves $NSGA-II$ is higher than the probability of $NSGA-II$ improves *MOSA* in both problems

	p-value
Henon	0.00020
Logistic	0.00013

We can discard with a confidence greater than 99% that the means of both variables are the same in favor of the alternate hypothesis, thus we can conclude that MOSA is a significant improvement wrt. NSGA-II in this particular application. In Figure 9 the boxplots of $\mathbf{P}_{\text{MOSA}}(\text{NSGA-II})$ and $\mathbf{P}_{\text{NSGA-II}}(\text{MOSA})$ for both problems are also given.

6 Concluding Remarks and Future Work

Modeling systems with chaotic dynamic is a complex task. It is easy to obtain a model with low error in a one-step prediction, but it is not easy to capture their dynamical properties. In this paper we have shown that many of these short-term models are stable, and not chaotic.

If a transparent model is needed, the one-step approach is questionable. However, using a larger horizon in the prediction is not feasible, since chaotic systems show a high dependency on the initial conditions. Therefore, we have decided to combine the one-step error and an invariant of the recursive evaluation of the model, its largest Lyapunov error. Our results have shown that, for simple chaotic systems, we are able to effectively obtain a model whose recursive evaluation converges to an strange attractor very similar to that of the original system. Moreover, we have shown that, for this task, the use of a Simulated Annealing-based search can improve the results of recent multicriteria genetic algorithms in both memory requirements and computational time.

Future work will be devoted to integrate the full spectra of Lyapunov exponents in the learning. This is needed to identify models with more than one positive exponent. In this last case, it is hard for our algorithm to obtain a good model, since most of the search is spent with models where only the largest exponent is similar.

The same can be said about unstable models, that are currently detected by mean of heuristics (i.e., limits in the range of the output of the recursive evaluation.) The full spectra or, at the least, the Kolmogorov entropy of the model should be evaluated and taken into account along with the one step error and the largest exponent.

Acknowledgments The research in this paper has been funded by project TIN2005-08386-C05-05, M.E.C., Spain

References

1. A. Alvarez, A. Orfila and J. Tintore, *DARWIN: An evolutionary programa for nonlinear modeling of chaotic time series.* Computer Physics Communications, 136, pp. 334–349, 2000
2. E.K. Burke and J.D. Landa Silva. *Improving the Performance of Trajectory-Based Multiobjective Optimisers by Using Relaxed Dominance.* In: Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim and Xin Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, 1, pp. 203–207, Nanyang Technical University, Orchid Country Club, Singapore, November 2002
3. H. Cao, L. Guo, Y. Chen and T. Guo, *The Dynamic Evolutionary Modeling of HODEs for Time Series Prediction.*, Computers and Mathematics with Applications, 46, pp. 1397–1411, 2003
4. Y.S. Chang, K.S. Park and B.Y. Kim, *Nonlinear model for ECG R-R interval variation using genetic programming approach.* Future Generation Computer Systems, 21(7), pp. 1117–1123, 2005
5. I-F. Chung, C-J. Lin and C-T. Lin. *A GA-based fuzzy adaptive learning control network.* Fuzzy Sets and Systems, 112, pp. 65–84, 2000
6. C.A. Coello. *List of References on Evolutionary Multiobjective Optimization.* http://www.lania.mx/ccoello/EMOO/EMOObib.html
7. C.A. Coello. *An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends.* In 1999 Congress on Evolutionary Computation, IEEE Service Center, 1, pp. 3–13, Washington, DC, 1999
8. P. Czyzak and A. Jaszkiewicz. *Pareto simulated annealing — a metaheuristic technique for multiple-objective combinatorial optimization.* Journal of Multi-Criteria Decision Analysis, 7, pp. 34–47, 1998
9. K. Deb, Samir Agrawal, Amrit Pratab and T. Meyarivan, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II.* In: Marc Schoenauer, K. Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo and Hans-Paul Schwefel, editors. *Proceedings of the Parallel Problem Solving from Nature VI Conference.* Paris, France. Springer. Lecture Notes in Computer Science No. 1917, p. 849–858, 2000
10. K. Deb and Tushar Goel. *Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence.* In: E. Zitzler, K. Deb, L. Thiele, C.A. Coello and David Corne, editors. *First International Conference on Evolutionary Multi-Criterion Optimization.* Springer-Verlag. Lecture Notes in Computer Science No. 1993, pp. 67–81, 2001
11. K. Downing. *Using evolutionary computational techniques in environmental modelling.* Environmental Modelling and Software, 13, pp. 519–528, 1998
12. C. Evans, P.J. Fleming, D.C. Hill, J.P. Norton, I. Pratt, D. Rees and K. Rodriguez-Vazquez, *Application of system identication techniques to aircraft gas turbine engines.* Control Engineering Practice, 9, pp. 135–148, 2001
13. A.I. Fernandez, L. Sanchez and J. J. Navarro. *Approximating the discrete space equation from chaotic noisy data (IPMU'2000).* In *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Madrid, Spain, pp. 149–156, 2000

14. D.B. Fogel and L.J. Fogel. *Preliminary Experiments on Discriminating between Chaotic Signals and Noise using Evolutionary Programming.* In: J.R. Koza, D.E. Goldberg, D.B. Fogel and R. L. Riolo, editors. *Genetic Programming 96.*, MIT Press, Cambridge, MA, 1996
15. C.M. Fonseca and Peter J. Fleming. *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms — Part I: A Unified Formulation.* IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 28(1), pp. 26–37, 1998
16. G.J. Gray, D.J. Murray-Smith, Y. Li, K.C. Sharman and T. Weinbrenner. *Nonlinear model structure identification using genetic programming.* Control Engineering Practice, 6, pp. 1341–1352, 1998
17. N.F. Guler, E.D. Ubeyli and I. Guler. *Recurrent neural networks employing Lyapunov exponents for EEG signals classification.* Expert Systems with Applications, 29, pp. 506–514, 2005
18. M. Hapke, Andrzej Jaszkiewicz and Roman Slowinski. *Pareto Simulated Annealing for Fuzzy Multi-Objective Combinatorial Optimization.* Journal of Heuristics, 6(3) pp. 329–345, August 2000
19. M. Hernández-Guía, R. Mulet and S. Rodrguez-Prez, *A New Simulated Annealing Algorithm for the Multiple Sequence Alignment Problem: The approach of Polymers in a Random Media.* Physical Review E, 72 (3), 2005
20. W. Jiang, Q. Guo-Dong and D. Bin. *Observer-based robust adaptive variable universe fuzzy control for chaotic system.* Chaos, Solutions and Fractals, 23, pp. 1013–1032, 2005
21. T. Jones. *Crossover, macromutation and population-based search.* In: *6th Internatonal Conference on Genetic Algorithms.* San Francisco, July 15–19, Morgan Kaufmann, 1, pp. 73–80, 2005
22. H. Kantz. *A robust method to estimate the maximal Lyapunov exponent of a time series.* Phys. Lett. A, 185, pp. 77–87, 1994
23. D. Kim. *Improving the fuzzy system performance by fuzzy system ensemble.* Fuzzy Sets and Systems, 98, pp. 43–56, 1998
24. D. Kugiumtzis, B. Lillekjendliey and N. Christophersen. *Chaotic time series. Part I: Estimation of some invariant properties in state space.* Identification and Control, 4(15), pp. 205–224, 1995
25. B. Lillekjendlie, D. Kugiumtzis and N. Christophersen. *Chaotic time series part II: System identification and prediction.* Identification and Control, 4(15), pp. 225–243, 1995
26. A. Lopez, H. Lopez and L. Sanchez. *Graph based GP applied to dynamical system modeling.* In: *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence, 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001*. Lecture Notes in Computer Science, 2084, pp. 725–732, 2001
27. S. Luke. *Two Fast Tree-Creation Algorithms for Genetic Programming.* IEEE Transactions on Evolutionary Computation, 4(3), pp. 274–283, September 2000
28. S. Luke and Liviu Panait. *A Survey and Comparision of Tree Generation Algorithms.* In: Lee Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001).* San Francisco, California, Morgan Kaufmann, pp. 81–88, 2001
29. M.T. Rosenstein, J.J. Collins and C.J. De Luca. *A practical method for calculating largest Lyapunov exponents from small data sets.* Physica D, 65, pp. 117-134, June, 1993
30. M.W. Mak, K.W. Ku and Lu. *On the improvement of the real time recurrent learning algorithm for recurrent neural networks.* Neurocomputing, 24, pp. 13–36, 1999
31. M.A. Matos and Paulo Melo. *Multiobjective Reconfiguration for Loss Reduction and Service Restorating Using Simulated Annealing.* In: *International Conference on Electric Power Engineering, Budapest 99.* IEEE, pp. 213–218, 1999
32. Y. Mei-Ying and W. Xiao-Dong. *Chaotic time series prediction using least squares support vector machines.* Chinese Physics, 13, pp. 454–458, 2004
33. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd ed. Springer-Verlag, 1996
34. S. Mukherjee, E. Osuna and F. Girosi. *Nonlinear Prediction of Chaotic Time Series Using Support Vector Machines.* in: *Proceedings of IEEE NNSP'97, Amelia Island, FL, USA*, IEEE Service Center, pp. 24–26, September 1997

35. D. Nam and Cheol Hoon Park. *Multiobjective Simulated Annealing: A Comparative Study to Evolutionary Algorithms.* International Journal of Fuzzy Systems, 2(2), pp. 87–97, 2000
36. D. Nam and Cheol Hoon Park. *Pareto-Based Cost Simulated Annealing for Multiobjective Optimization.* In: Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim and Xin Yao, editors. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02).* Nanyang Technical University, Orchid Country Club, Singapore, 2, pp. 522–526, November 2002
37. R. Poli and Nicholas F. McPhee. *Exact GP Schema Theory for Headless Chicken Crossover and Subtree Mutation.* In: *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001 COEX,* World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, IEEE Press, pp. 1062–1069, 2001
38. P. Potocnik and I. Grabec. *Nonlinear model predictive control of a cutting process.* Neurocomputing, 43, pp. 107–126, 2002
39. K. Rodríguez-Vázquez, C.M. Fonseca and P.J. Fleming. *Identifying the Structure of NonLinear Dynamic Systems Using Multiobjective Genetic Programming.* IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans, 34(4), pp. 531–545, July 2004
40. J.J. Rowland. *Model selection methodology in supervised learning with evolutionary computation.* BioSystems, 72, pp. 187–196, 2003
41. A.E. Ruano, P.J. Fleming, C. Teixeira, K. Rodriguez-Vazquez and C.M. Fonseca. *Nonlinear identification of aircraft gas-turbine dynamics.* Neurocomputing, 55, pp. 551–579, 2003
42. L. Sanchez, I. Couso and J.A. Corrales. *Combining GP operators with SA search to evolve Fuzzy Rule based classifiers.* Information Sciences, 1–5, pp. 175–192, 2001
43. R.S. Sexton and J.N.D. Gupta. *Comparative evaluation of genetic algorithm and backpropagation for training neural networks.* Information Sciences, 129, pp. 45–59, 2000
44. T. Shin and I. Han. *Optimal signal multi-resolution by genetic algorithms to support artificial neural networks for exchange-rate forecasting.* Expert Systems with Applications, 18, pp. 257–269, 2000
45. Kevin I. Smith, Richard M. Everson and Jonathan E. Fieldsend. *Dominance Measures for Multi-Objective Simulated Annealing.* In: *2004 Congress on Evolutionary Computation (CEC'2004).* Portland, Oregon, USA. IEEE Service Center, 1, pp. 23–30, June, 2004
46. J.C. Sprott. *Chaos and Time-Series Analysis.* Oxford University Press, 2003
47. Z. Wei, W. Zhi-ming and Y. Gen-ke. *Genetic programming-based chaotic time series modeling.* Journal of Zhejiang University SCIENCE, 5(11), pp. 1432–1439, 2004
48. A. Wolf, J.B. Switf, H.L. Swinney and J. A. Vastano. *Determining Lyapunov Exponents from a Time Series.* Physica D, 16, pp. 285–317, 1985
49. A.M. Woodward, R.J. Gilbert and D. B. Kell. *Genetic programming as an analytical tool for non-linear dielectric spectroscopy.* Bioelectrochemistry and Bioenergetics, 48, pp. 389–396, 1999
50. E. Zitzler, K. Deb and L. Thiele. *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.* Evol. Comput., 8(2), pp. 173–195, 2000
51. E. Zitzler, M. Laumanns, L. Thiele, C.M. Fonseca and V. Grunert da Fonseca, *Why Quality Assessment of Multiobjective Optimizers Is Difficult.* In: W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli and K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke and N. Jonoska, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002).* San Francisco, California, pp. 666–673, July 2002
52. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca and V. Grunert da Fonseca. *Performance Assessment of Multiobjective Optimizers: An Analysis and Review.* IEEE Transactions on Evolutionary Computation, 7(2), pp. 117–132, April 2003