Obtaining transparent models of chaotic systems with multiobjective simulated annealing algorithms

Luciano Sánchez, José R. Villar *
Computer Science Department, Universidad de Oviedo
Edificio Departamental 1, Campus de Viesques, 33213 Gijon (Spain)

Abstract

Transparent models search for a balance between interpretability and accuracy. This paper is about the estimation of transparent models of chaotic systems from data, which are accurate and simple enough for their expression to be understandable by a human expert. The models we propose are discrete, built upon common blocks in control engineering (gain, delay, sum, etc.) and optimized both in their complexity and accuracy.

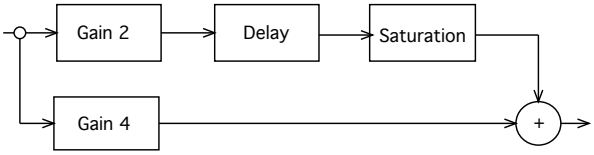
The accuracy of a discrete model can be measured by means of the average error between its prediction for the next sampling period and the true output at that time, or 'one-step error.' A perfect model has zero one-step error, but a small error is not always associated with an approximate model, especially in chaotic systems. In chaos, an arbitrarily low difference between two initial states will produce uncorrelated trajectories, thus a model with a low one-step error may be very different from the desired one. Even though a recursive evaluation (multi-step prediction) improves the fitting, in this work we will show that a learning algorithm may not converge to an appropriate model, unless we include some terms that depend on estimates of certain properties of the model (so called 'invariants' of the chaotic series). We will show this graphically, by means of the reconstructed attractors of the original system and the model. Therefore, we also propose to follow a multiobjective approach to model chaotic processes and to apply a simulated annealing-based optimization to obtain transparent models.

Keywords: Multiobjective Simulated Annealing, Chaotic Systems, Transparent Models, Genetic Programming.

1 Introduction

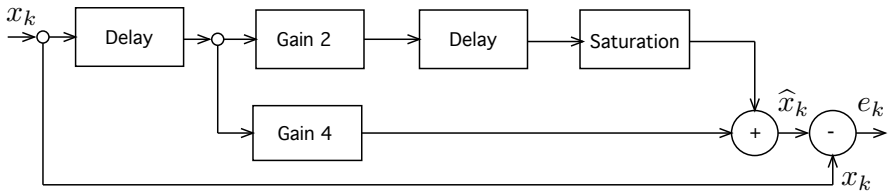
The mathematical models of chaotic systems have to balance complexity and accuracy. We expect a technique that produces a *black box* from data [15, 24, 30, 36,

*Corresponding author. Tel.: +34 985182597; fax: +34 985 181 986. E-mail address: villarjose@uniovi.es



$$\widehat{x}_{k+1} = \sigma(2x_{k-1}) + 4x_k$$

Figure 1: Example of graphical representation of a model comprising common building blocks in control engineering, and its corresponding difference equation. The function σ defines a nonlinear gain, the 'saturation' block.



$$e_k = x_k - \sigma(2x_{k-2}) - 4x_{k-1}$$

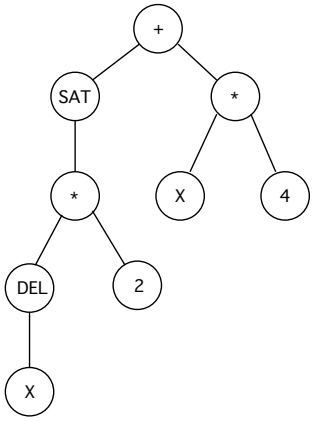
Figure 2: Graphical representation of the one-step error of the model in Figure 1.

39, 47, 7, 29] to produce more accurate results than other procedures that also gain insight into the block structure of the system.

In our opinion, one of the most useful representations of a model comprises a set of building blocks such as the gain, sum, or delay (see Figure 1) which, in turn, is equivalent to a set of discrete, nonlinear difference equations. A difference equations-based model allows the user not only to predict the output of the process, but also to know the dynamics of the model and ultimately to design a control system for it. Thus we will say that Figure 1 displays a *white box* or a transparent model.

However, obtaining a transparent model from data is a problem that has not been solved yet. Our aim is to discover a consistent subset of state variables and the equations that relate them, and also the numerical values of the coefficients in these equations. Some of the most recent approaches to obtain this information are based on evolutionary techniques, combined with a tree-based representation of the model [4, 6, 14, 19, 46, 56]. In Figure 3 there is a simplified example of such a representation, that will be explained in depth in Section 3.2. The use of a tree-based representation permits to define a simultaneous search in both the different families of models, given by the different shapes the tree can adopt, and the parameters that define a model within one of these families (in the example, the parameters are values '2' and '4', which are leaves of the tree in Figure 3).

In some of the mentioned evolutionary algorithms, the objective function measures the discrepancies between the actual data and the prediction of the model,



(+(SAT(*((DEL X) 2)) *(X 4)))

$$\widehat{x}_{k+1} = \sigma(2x_{k-1}) + 4x_k$$

Figure 3: Simplified tree-based representation of the model displayed in Figure 1. 'SAT' stands for 'saturation' and 'DEL' for 'delay'. The prefix expression in the figure is a representation of the tree as a chain of symbols. In Section 3.2 we will show that a grammar can be defined, in order to determine which chains are valid and which ones are not.

the so called *one-step error* (see Figure 2). This technique does not fully take into account the dynamic behavior of the model [47, 54]. As we will show later, if we search for a model on the basis of the lowest one-step prediction error, we have a good chance of ending up with a non-chaotic model. Generally speaking, a zero one-step error proves that the model is exact, but a small error does not mean that the model is a good approximation. In chaotic processes, this effect is stressed, because of the sensitivity to initial conditions; that is to say, a small difference between two initial states produces uncorrelated trajectories past a number of periods. Because of the same reason, the use of greater prediction horizons is not always feasible.

Therefore, we propose to use both the one-step error and the value of the largest Lyapunov exponent of our model. The largest Lyapunov exponent is a measure of the amount of chaos in the signal [28, 30, 55], and it has to match that value estimated from training data. The difference between the maximum Lyapunov exponents of two models also gives us a measure of similarity between the complexities of their dynamics [20, 54]. In particular, our multiobjective problem is designed to minimize the prediction error and the complexity of the model, while restricting the search to those models whose largest Lyapunov exponents are close to the estimated value from the time series we want to analyze.

The main drawback with our approach is the time needed to estimate the Lyapunov exponents. Multicriteria genetic algorithms are known to require a large

number of evaluations of the fitness function. In this respect, we propose to use an ad hoc evolutionary algorithm that combines a tree-based representation with a population-based, multiobjective extension of the Simulated Annealing. This algorithm is able to produce a set of difference equations that reproduces the dynamics of a sampled chaotic process, and improves the results of modern multiobjective evolutionary algorithms like NSGA-II [12, 13] when the number of evaluations of the objective function is limited. Observe that the comparison of two multiobjective evolutionary algorithms is a current research area itself. We have used a methodology of our own, based on binary indicators [58].

In the next section, we focus on the open problems in the transparent modeling of chaotic systems, and introduce our approach, which is fully explained in Section 3. Experiments and results are shown in Section 4, and the paper finishes with the concluding remarks and highlights some future work.

2 Issues with evolutionary transparent modeling of chaotic systems

2.1 Evolutionary models of chaos

Evolutionary algorithms (genetic algorithms, genetic programming and evolutionary programming) have been applied to identify and control nonlinear and chaotic systems. The reader can refer to [34, 49], where genetic algorithms are compared with different identification techniques, or review the results in [2, 7, 8, 27, 43, 50, 53]. The control problem is less studied. For instance, in [43], a genetic algorithm was used to find the optimal control signals sequence in a chaotic cutting process. Fuzzy controllers have been used in [5, 7, 53].

The most often used fitness function is scalar, and it measures the accuracy of the prediction, but there have also been some transparent models for chaos. None of them is based on the building blocks we propose, though. For instance, linguistic fuzzy rules were combined with genetic algorithms in [8] and in [27], and applied to analyze chaotic time series. Wavelet coefficients are also considered to provide a certain degree of interpretability, as they were used in [50], where genetic algorithms were applied to select wavelet threshold parameters in an exchange-rate forecasting problem.

Other approaches for non-linear modeling use polynomial models, as can be seen in [15, 44]. Many other different, problem specific, analytical modeling approaches have been developed. For example, in [1], evolutionary algorithms were used to obtain nonlinear models for a satellite based ocean forecasting system. In [14], evolutionary computing was used for extracting mathematical models, and this proposal was analyzed with three different applications. In [19] genetic programming was used to find difference equations models of non-linear processes, as we propose in this article. Finally, in [2] genetic-neural experts are combined and used for stock index forecasting.

2.2 Instantaneous error and recursive predictions

In our opinion, transparent models should not be designed only on the basis of the one-step error, although the fitness of an individual is based only on instantaneous error measures in all the preceding methods.

Understandable models are intended to obtain knowledge about the structure of the physical process. We can lose some accuracy in the prediction if it helps to obtain a simpler model and gain a better insight, but if the learning produces a set of non chaotic equations for a chaotic system, no matter how simple they are, they do not provide the best information. In Figure 4 there is an example of a model with a good one-step error but with the wrong structure, which helps to make our point clearer.

There are few publications where the information about the dynamics of the system is used. For example, in [17], 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. Nevertheless, chaotic models are sensitive to initial conditions. The recurrent outcomes of a chaotic model are very different under small differences of the initial state. Thus, this measure of error can be discussed, but our own approach shares properties with this method. In the following section, we propose to evaluate the dynamical properties of a candidate model by means of its recursive evaluation, 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.

2.3 Multicriteria design of models

Multiobjective techniques have been previously used to develop models for non linear and chaotic systems. In some of our own previous works [16], we have chosen using a linear combination of the quadratic error and the largest Lyapunov exponent for the fitness function, and have optimized it by using a genetic algorithm. In [15, 44] a Pareto based approach is used instead of scalar functions, in combination with the MOGA algorithm described in [18]. There are some different Pareto based multiobjective strategies that could also be applied to the same problem, as can be seen in [9]. Later, we will evaluate a more recent approach, the NSGA-II algorithm [12, 13].

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 [57]. However, in our opinion, the experimentations that support this assertion were intended to solve problems based on a linear genotype, and cannot be immediately extrapolated to tree-based representations. We will show that some metaheuristics can improve the results of multiobjective genetic algorithms.

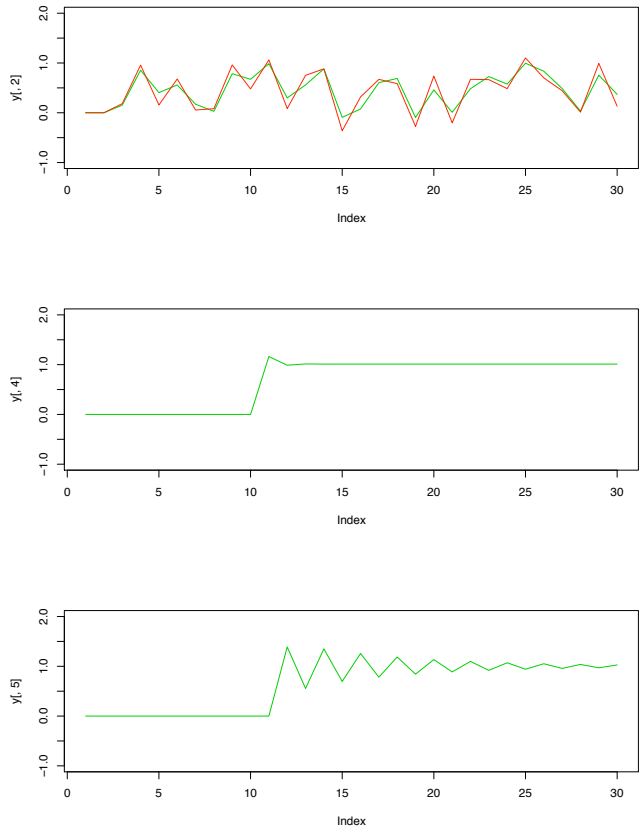


Figure 4: Upper part: One-step prediction of two linear models, given by the transference functions $\frac{1.39z}{z^2+0.6z-0.21}$ and $\frac{1.16z}{z+0.15}$. The one-step error is small, although the structure of both models is different. Central and lower parts: recursive predictions of both models for a step input (0 before the tenth sample, 1 for samples between 10 and 30). The step response shows the differences in the dynamics of these models, that were not clear from the graph in the upper part. The same thing happens with chaotic models: many non-chaotic models have a very good one-step prediction error in chaotic series, but their recursive prediction does not produce a strange attractor, as it will be shown later.

2.3.1 Multicriteria Simulated Annealing

In previous works [48], we have combined a simulated annealing (SA) global search with a grammar-tree based representation, in the context of the learning of fuzzy rules. The results of the genetic algorithm (GA) were improved by a 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. Taking this into account, in this paper we will extend our own algorithm to multiobjective problems, and propose a new population-based, multi-objective SA (MOSA) search, able to elicit a set of non-dominated 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.

It is interesting to mention that a pure Pareto-based MOSA search has not been previously defined, to the best of our knowledge. The most recent approaches weigh the different criteria into a scalar function [23, 35, 51]. Otherwise, in [11] it was proposed to use the dominance to decide the evolution of the simulated annealing. That approach was also used in [21], where fuzzy numbers and uncertainty in dominance are managed to decide if an individual is better than another or not. Similarly, in [40, 41], Pareto dominance is studied to decide how the multiobjective simulated annealing evolves. But, in all of these cases, an weighted sum of objectives is still used to evaluate each individual. A different approach to Pareto based MOSA, nearer to ours, is introduced in [3], where a comparison of a Pareto-based evolutionary algorithm and a population-based simulated annealing with dominance control approach is shown. In each simulated annealing iteration, a new individual is obtained by means of a heuristic and included in the population if there is a non dominance 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, weighing the different objectives into a scalar function and therefore, it can be argued that the algorithm does not homogeneously sample the Pareto front. In the following sections we will introduce a different algorithm that does not pose this problem.

3 Multicriteria design of models with Simulated Annealing

In this section we explain our proposal for obtaining a model which balances accuracy, interpretability and dynamic behavior, as stated in the introduction. The algorithm we are about to introduce is a multiobjective extension of the SA-P algorithm defined in [48].

Needs:
Initial and final temperatures: $T_{initial}$, T_{final}
Cooling factor : C

Produces:
A set of nondominated models: PARETO

```
Initialize the population of models: X = {x_0}
Initialize the set of elites : PARETO = X
T ← T_initial
while T ≥ T_final
    // X' is the intermediate population
    X' ← ∅
    for each x ∈ X
      x_mutated ← mutation(x)
      if x_mutated ≺ x then
        X' = X' ∪ {x_mutated}
      else if x ≺ x_mutated then
        if rnd() < exp(-distance(x, x_mutated)/T) then
          X' = X' ∪ {x_mutated}
        else X' = X' ∪ {x}
      else
        X' ← X' ∪ {x, x_mutated}
      end if
    end for
    PARETO ← nondominated models of the joint set PARETO ∪ X'
    X ← selection(X')
    T ← T · C
end while
```

Figure 5: Pseudocode of the MOSA algorithm

3.1 Outline of the algorithm

The pseudocode of the Multi-Objective Simulated Annealing (MOSA) algorithm is shown in Figure 5. This algorithm is based on a variable sized search points population. 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.

Our strategy 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.

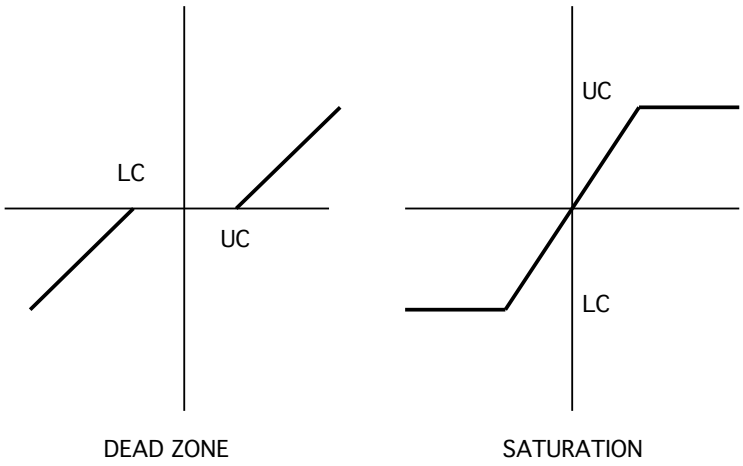


Figure 6: Graphical representation of the transference functions of the building blocks used in the models.

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 an SA search, the probability of admitting the mutated point depends on the cooling pattern and decreases with both the distance between the individuals and the time. The distance we have used is explained in Section 3.5.

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 (see Section 3.6) to form the following generation. An elitist set of non-dominated solutions is also kept aside the population. This set is the current sample of the Pareto front and will eventually be the output of the algorithm.

3.2 Representation of an individual

As we have mentioned before, our models are built upon common building blocks in control engineering. Following [31], our catalog comprises the dead zone, saturation, subtraction, product and delay. In Figure 6 the transference functions of the less evident blocks are displayed. The 'delay' operator produces the value that a given variable has had a certain number of periods ago.

The phenotype of an individual is a list of equations. Each equation assigns an expression to one state variable, and it is assumed that the first state variable is also the output of the model. Any expression has to be a valid chain of the grammar that follows:

EXP $\mapsto$ ArithOp | NonLinearOp | DelayOp | Param | Variable
ArithOp $\mapsto$ (**+** EXP, EXP) | (**-** EXP, EXP) | (***** EXP, EXP)
NonLinearOp $\mapsto$ (**Saturation** Param Param EXP) | (**DeadZone** Param Param EXP)
DelayOp $\mapsto$ (**Delay** Param Variable)

for instance, the expression

$$\sigma(2x_{k-1}) + 4x_k$$

where $\sigma(x) = \text{sign}(x) \cdot \min\{|x|, 1\}$, is codified by the chain

```
(+ (SATURATION -1 1 (* 2 (DELAY 1 X))) (* 4 X)).
```

We use a tree-shaped genotype, (a simplified example has already been given in Figure 3). The complete representation also includes labels in the nodes and the edges of the tree, which will be used in the mutation operator defined in the next section. The genotype is the parse tree of the expression. An actual example of it is shown in Figure 7, where the following set of equations is codified:

$$\begin{array}{rcl} x^1_{k+1} & = & \sigma(4x^1_{k-1}) + 5x^2_k \\ x^2_{k+1} & = & x^3_k \\ x^3_{k+1} & = & x^1_{k-1} \end{array}$$

Each node of this tree encodes the name of the production rule that originated each subtree. This information will be used later to define a typed crossover. Note that we do not codify numerical parameters in the leaves of the tree, but rather references to a chain of named parameters.

3.3 Mutation and crossover operators

The Simulated Annealing algorithm relies on the mutation operator to produce new models, therefore the crossover is not needed. However, we will implement the mutation by means of a subtree crossover with a randomly generated individual [25][42] to define the mutation in terms of the crossover operator. Notice that the crossover operator defined in this section will be reused in the genetic algorithm to which we will compare our numerical results in Section 4.

Since the numerical parameters are encoded in a separate chain rather than in the leaves of the tree, our crossover operator has two different expressions, that we will call *parametric* and *structural*. The parametric crossover takes place between the chains of parameters, and the structural one between the trees. Leaving apart the differences in the grammar, the same operators proposed in [48] were used:

- *Parametric crossover:* Extended intermediate crossover [37] between the chains of parameters.

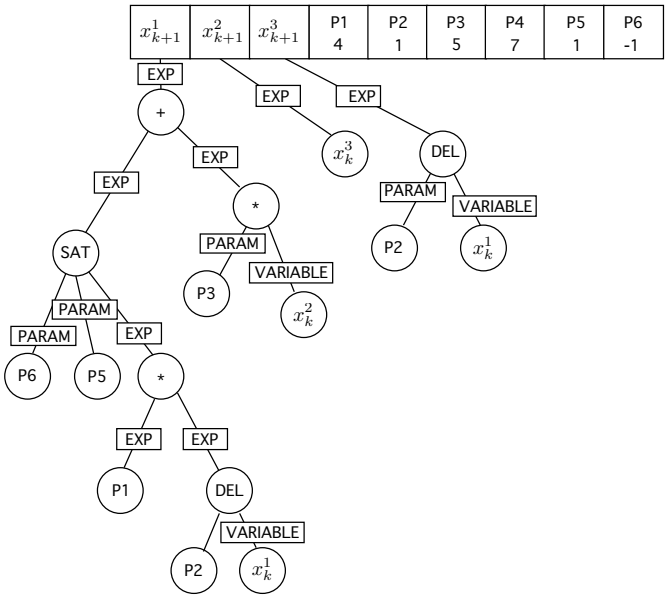


Figure 7: Genotype of an individual in the MOSA algorithm. An individual comprises a list of equations and a chain of parameters. Each equation is, in turn, a labeled tree.

- *Structural crossover:* Typed subtree exchange [38]. The grammar proposed before is used to determine the compatibility of subtrees, e.g. an edge with the label 'EXP' can receive a subtree of type 'ArithOp' but an edge labeled 'Param' can not receive a 'Variable' type tree .

3.4 Random generation of genotypes

The PTC2 algorithm is used to generate random trees [32, 33]. This algorithm allows to specify the maximum number of nodes, the types of nodes, the probability distribution of each type of node, the maximum height and the probability distribution for each tree height, conditioned to our grammar.

3.5 Distance between individuals

The distance function is used in Simulated Annealing to check that the mutated individuals are close to the initial individual.

In previous works [48], we postulated the use of an *edition distance* between trees. This distance is the number of edition operations (add, remove or replace a node) needed to transform the current into the new model. Moreover, in the same paper we also checked that similar individuals (in terms of the edition distance) 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.

3.6 The selection operator

The size of the intermediate population can be twice as high as the current population size, in the worst case.

Our selection operator is a variation of that used in the NSGA-II algorithm [12, 13]. Firstly, the set of nondominated search points is computed by pairwise comparisons of all individuals in the intermediate population X'. 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 first those points in less dense areas.

3.7 Fitness function

The fitness function comprises two numbers: the average error of the one-step prediction of the model, and the absolute difference between the estimations of the largest Lyapunov exponents of the model and the training data.

Different procedures have been proposed to compare this kind of compound values [10]. We will use a Pareto multiobjective evaluation, and guide the search towards obtaining a set of non dominated individuals. In the most general case, it is said that an individual x dominates another individual y ($x \prec y$,) when 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, irrespective of their Lyapunov value.

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

We evaluated some different numerical algorithms. Our first choice was the well-known Wolf algorithm [55], that we had already used in previous works. The number of samples that this algorithm needs is rather high; this, altogether 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. There are other algorithms, in particular those of Rosenstein and Kantz [45][26], 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

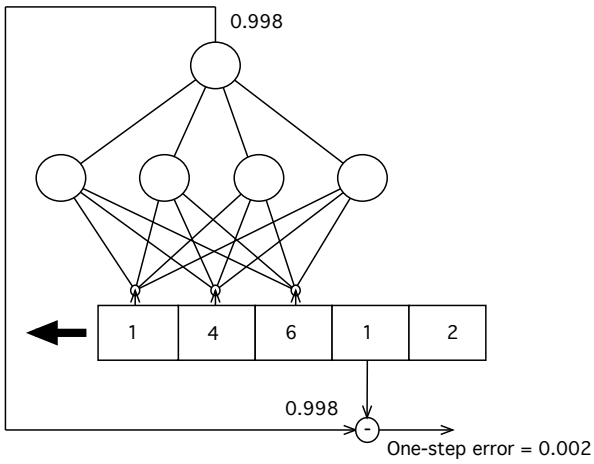


Figure 8: Multilayer perceptron as a model a chaotical time series. The embedding dimension is 3, and the error is computed as the average of the squared one-step prediction error. Neural networks can approximate the series with very high accuracy, nevertheless the model they learn can be non-chaotic.

point where the slope of the curve time vs. divergence changes. 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 [52]: we follow the divergence of two close trajectories. One of them is retained for reference. The other is repeatedly renormalized so that the distance between both is kept short. The maximum Lyapunov exponent is then estimated by the average value of the logarithm of the initial distance between the trajectories, divided by the distance between the predictions, before renormalizing.

4 Experimental results

4.1 One-step error compared to the proposed fitness function

As mentioned in the introduction, a good error in the one-step prediction does not mean that the dynamic behavior of the system has been captured. In Figure 4 we gave an example of a linear model with the wrong structure, but a low one-step error. In this section we will provide a new example where a chaotical time series will be modeled by a multilayer perceptron.

The neural network will be trained to minimize the squared difference between the one-step prediction and the true output of the system, e.g. let us suppose that the chaotic series is

$$1, 4, 6, 1, 2, 6, 5, \ldots$$

and we choose an embedding dimension of size 3. Then, we train a neural network with three inputs and one output (see Figure 8), using the training set that follows:

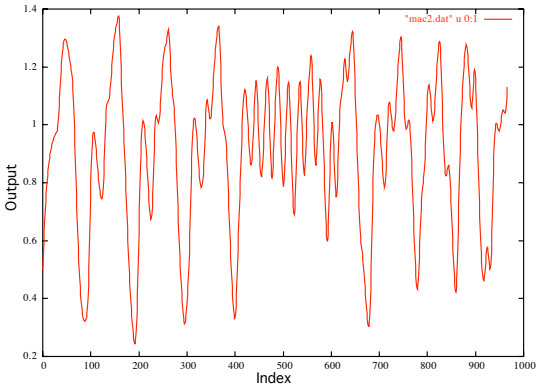


Figure 9: Chaotic time series used to analyze the one-step prediction error based fitness function

Input	Output
1, 4, 6	1
4, 6, 1	2
6, 1, 2	6
...	

In particular, we have modeled the chaotic series shown in Figure 9 with four multilayer perceptrons, using the conjugate gradient method [22]. The nets have a sigmoidal activation in the hidden layer and a linear activation in the output layer. The first 80% of the data was used to train the net, and the last 20% was used to test it. Their respective sizes and the order of magnitude of the errors are shown in the table below:

Multilayer Perceptron		
Embedding dimension	Nodes in each layer	Err
1	1 - 3 - 1	10^{-3}
2	2 - 5 - 1	10^{-4}
3	3 - 10 - 1	10^{-5}
4	4 - 10 - 1	10^{-5}

The errors are very low. If we plotted the prediction of the net over the real data, there was virtually no difference. Let us compute recursive trajectories for these neural networks, as shown in Figure 10, applying the same procedure that

we used in the example in Figure 4. The results are plotted in Figure 11. All these models are stable and therefore, there are no strange attractors.

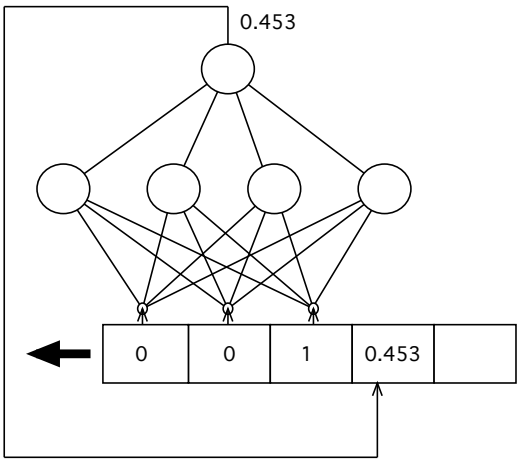


Figure 10: Recursive evaluation of a multilayer perceptron to check the properties of the attractor. The output of the net is added to the input series and the data is shifted to obtain the next prediction.

The neural networks have been selected because they are one of the most accurate black-box models in the literature. There is nothing intrinsically wrong with the use of neural networks to obtain short term predictions of chaotic series. For instance, if we use a genetic model instead, it happens the same. In Figure 12, a genetic algorithm was used, with the same representation and operators described in the text except for a 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 central part of the figure. And, in the lower part of the same figure, the recursive evaluation of a model that optimizes the fitness function proposed in this work is shown. In the next section we will also plot some reconstructed attractors, where the similarities are more evident.

4.2 Benchmark problems

In this section we will compare the results of MOSA and NSGA-II with some benchmark problems. The NSGA-II is an implementation of the Pareto based multiobjective genetic algorithm detailed in [12, 13].

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 experiment. The statistical (quantitative) comparison of the results of multiobjective Evolutionary Algorithms is a current research field. There are many different measures of how much a Pareto front improves the results of another one, but

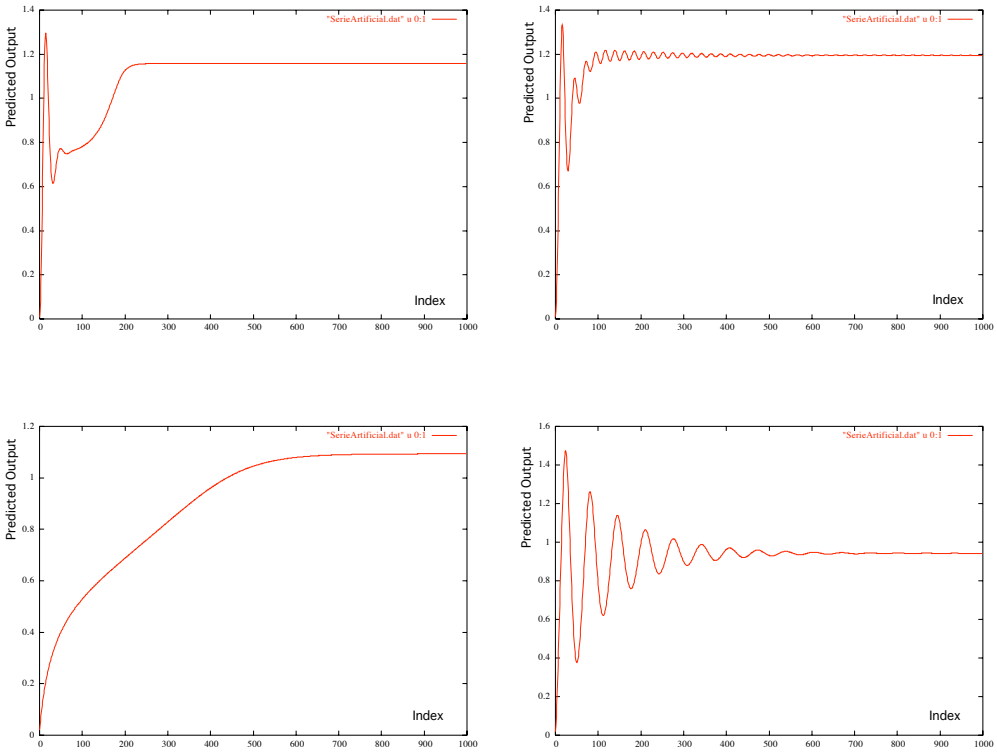


Figure 11: Multilayer perceptrons with one hidden layer, trained on the series in Figure 9 to minimize the one-step error. Upper part: Recursive evaluation of the neural networks 1-3-1 and 2-5-1. Lower part: Networks 3-10-1 and 4-10-1. Despite the low values in the objective function shown in the text, all trajectories converge to a point in the space state, thus none of the models is chaotic.

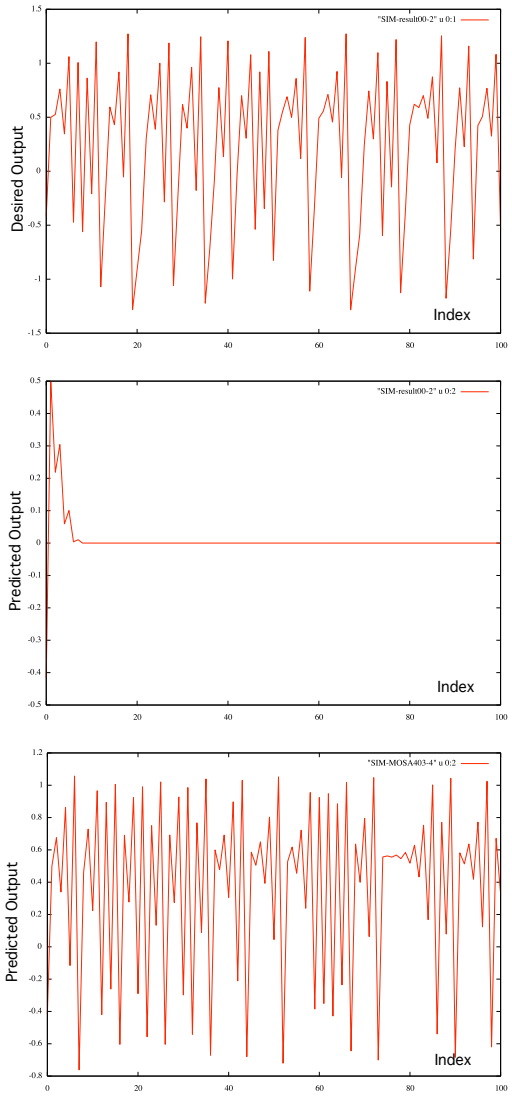


Figure 12: Graphical analysis of experimental results, Henon map. Upper part: Train data. Center: Typical recursive evaluation of a transparent model obtained by an evolutionary algorithm when the maximum Lyapunov exponent is *not* included in the fitness function; in this case, the optimization has converged to a stable model (Lyapunov exponent < 0). Lower part: recursive evaluation of one of the models found with the MOSA algorithm.

it is acknowledged that there are problems derived from the stochastic nature of evolutionary algorithms that are still unsolved [57, 58, 59]. We propose to use a statistical test about the probability that one algorithm dominates the other, based on the binary ϵ-indicator described in [58]. Both the qualitative and quantitative analysis will be explained in the next sections.

4.2.1 Experimental setup

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

NSGA-II			
Parameter	Value	Parameter	Value
Structural crossover	0.5	Parametric crossover	0.5
Structural mutation	0.01	Embedding dimension	2
Population size	100	Evaluations of fitness	5000
Minimum value of a parameter	-5	Maximum value of a parameter	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
Minimum value of a parameter	-5	Maximum value of a parameter	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 5000 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	+; -; *; Delay; Saturation; Dead Zone
Node type probability distribution	.21 .21 .21 .21 .09 .07

Each experiment was repeated 10 times. The time series used for training and validation has a size of 1000. The chaotic systems that have have been used are the

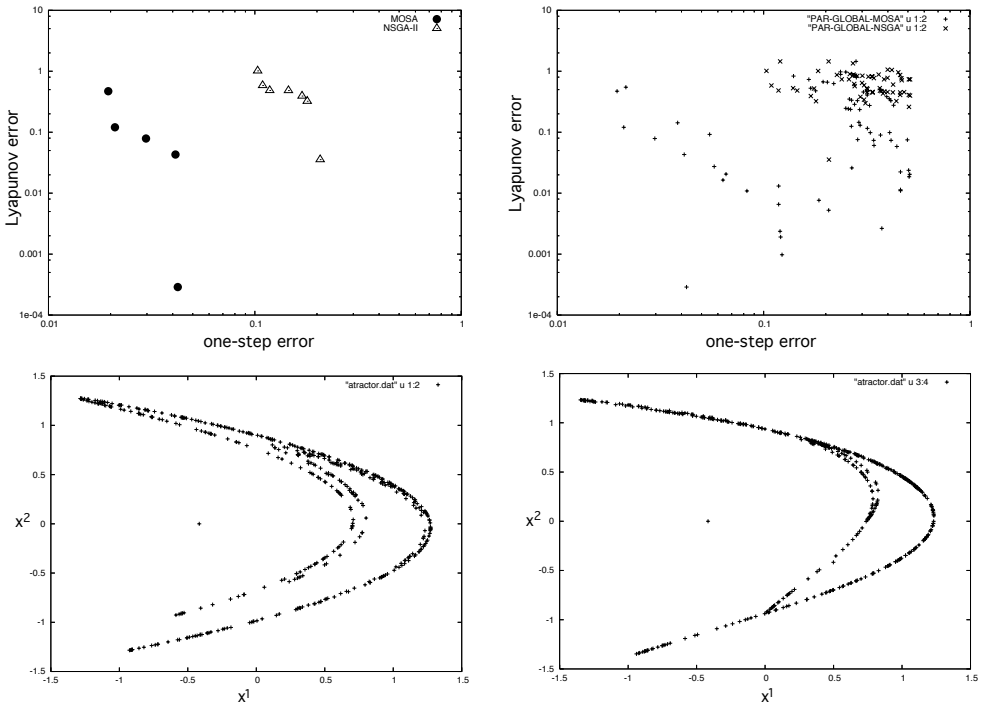


Figure 13: Graphical analysis of experimental results, Henon map. Upper part, left: Combined Pareto front of ten repetitions of the algorithms NSGA-II (triangles) and MOSA (circles). All the models in the Pareto front of the NSGA-II algorithm are dominated by at least one element in the Pareto front of the MOSA. To enhance the differences, a logarithmic scale is used. The vertical axis 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.

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

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

$$\text{Henon map:} \quad \begin{cases} x^1_{k+1} = x^2_k \\ x^2_{k+1} = 1 + 0.3 * x^1_k - 1.4 * x^2_k * x^2_k \end{cases} \tag{2}$$

4.2.2 Commented graphical results

The graphical results are displayed in Figures 13 and 14. In both cases, we have obtained the combined Pareto front (upper left part) after 10 repetitions of each algorithm. This combined Pareto front is formed by selecting all the non dominated individuals of the 10 runs. In the upper right part, all the elements of the 10 Pareto

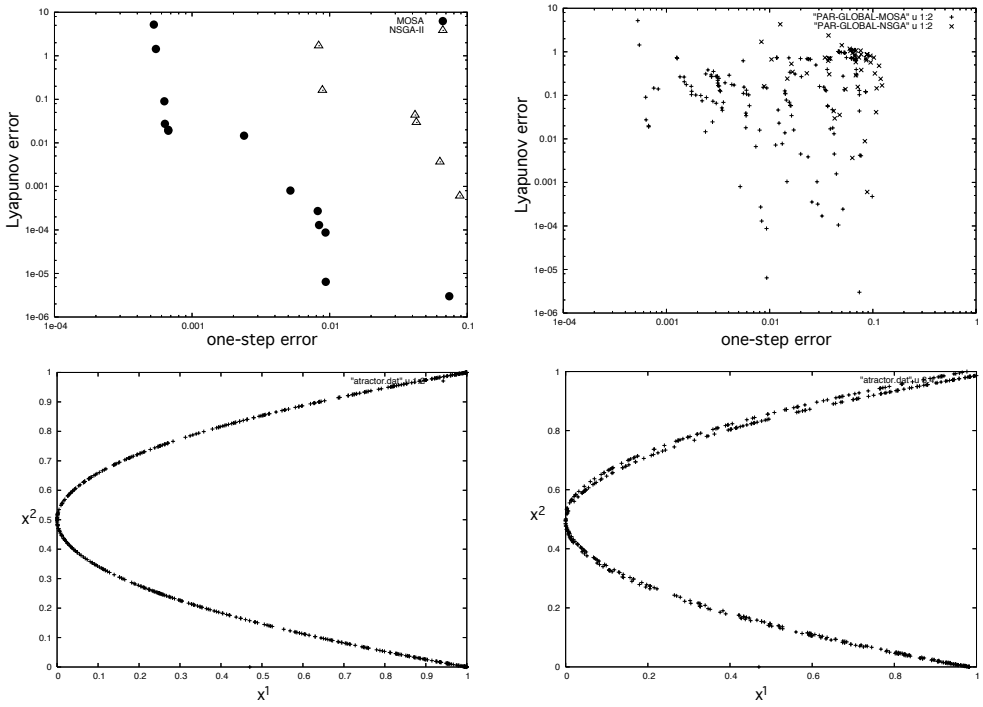


Figure 14: Graphical analysis of experimental results, Logistic map. Upper part, left: Combined Pareto front of ten repetitions of the algorithms NSGA-II (triangles) and MOSA (circles). All but one of the models in the Pareto front of the NSGA-II algorithm are dominated by at least one element in the Pareto front of the MOSA. To enhance the differences, a logarithmic scale is used. The vertical axis 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.

fronts of each algorithm are displayed together, in the same graph. Lastly, in the right lower part of the figures we have displayed a couple of reconstructed attractors that show the similarities between the dynamic behavior of the models (right part) and that of the original system (left part).

There is a clear difference between the combined fronts, because all of the points in the NSGA-II front are dominated by those of the MOSA. The difference is not so clear in the combined clouds (upper right part), since some of the executions of MOSA were dominated by NSGA-II and vice versa. In the next section, we will study the extent to which, on average, one algorithm is better than the other.

4.2.3 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 cannot show all the dominance relations that can occur between Pareto fronts [59]. Therefore, to assess the average improvement between one algorithm and the other, we will use a method based on a binary indicator, namely the binary ϵ-indicator defined in [58].

Two different definitions of this last indicator are possible: the standard (multiplicative) I_ϵ and the additive indicator $I_{\epsilon+}$. Given two fronts A and B, if $I_\epsilon(A,B) < 1$ and $I_\epsilon(A,B) > 1$, or if $I_{\epsilon+}(A,B) < 0$ and $I_{\epsilon+}(A,B) > 0$, 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

	$I_{\epsilon+}$(MOSA,NSGA)	$I_{\epsilon+}$(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 those of NSGA-II. These results are not conclusive, though, since one exceptionally good result of either algorithm could be responsible for the dominance of the combined Pareto front. So we propose to apply the ϵ-indicator to perform a full set of comparisons between all pairs of fronts.

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

$$P_A(B) = \frac{1}{10} \sum_{i=1}^{10} p_A(B_i). \tag{3}$$

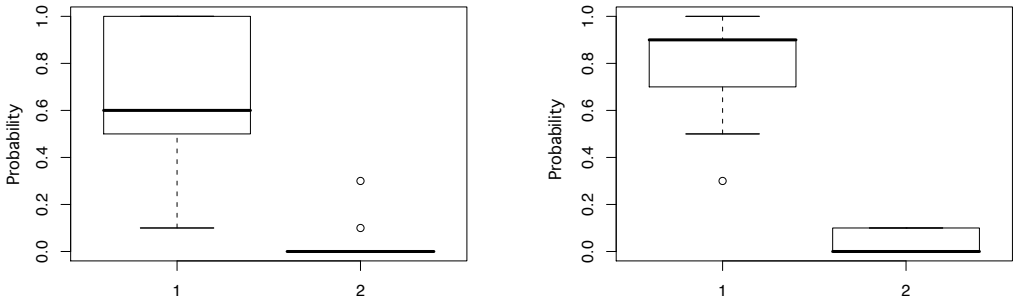


Figure 15: Boxplots of (1) $\mathbf{P}_{\text{MOSA}}(\text{NSGA-II})$ and (2) $\mathbf{P}_{\text{NSGA-II}}(\text{MOSA})$ for the Henon map (left part) and Logistic map (right part). This graph shows that the probability of MOSA improving NSGA-II is higher than the probability of NSGA-II improving MOSA in both problems.

Given other 10 repetitions $A_1, \ldots, A_{10}$ of an algorithm A, let

$$\mathbf{P}_A(B) = (P_{A_1}(B), P_{A_2}(B), \ldots, P_{A_{10}}(B)). \tag{4}$$

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 than the expectation of $\mathbf{P}_B(A)$, then we can state that the algorithm A is better than the algorithm B, since it is more likely that results of the former improve those 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:

	p-value
Henon	0.00020
Logistic	0.00013

We can surely discard 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 with respect to NSGA-II in this particular application. In Figure 15 the boxplots of $\mathbf{P}_{\text{MOSA}}(\text{NSGA-II})$ and $\mathbf{P}_{\text{NSGA-II}}(\text{MOSA})$ for both problems are also given.

5 Concluding remarks and future work

Modeling chaotic dynamic systems 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. We have already shown that many of these short term models are 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 a 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 algorithm. 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 means of heuristics (i.e., limits in the range of the output of the recursive evaluation). The full spectra or, at least, the Kolmogorov entropy of the model should be evaluated and taken into account along with the one-step error and the largest exponent.

Acknowledgements

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 program for nonlinear modeling of chaotic time series. *Computer Physics Communications*, 136:334–349, 2000.

[2] G. Armano, M. Marchesi, and A. Murru. A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences*, 170:3–33, 2005.

[3] 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)*, pages 203–207, 2002.

[4] H. Cao, L. Guo, Y. Chen, and T. Guo. The dynamic evolutionary modeling of HODEs for time series prediction. *Computers and Mathematics with Applications*, 46:1397–1411, 2003.

[5] P. C. Chang and C. H. Liu. A TSK type fuzzy rule based system for stock price prediction. *Expert Systems with Applications*, 34:134–144, 2008.

[6] 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):1117–1123, July 2005.

[7] B. Chen, X. Liu, and S. Tong. Adaptive fuzzy approach to control unified chaotic systems. *Chaos, Solitons and Fractals*, 34:1180–1187, 2007.

[8] I. F. Chung, C. J. Lin, and C. T. Lin. A GA-based fuzzy adaptive learning control network. *Fuzzy Sets and Systems*, 112:65–84, 2000.

[9] C. A. Coello. List of references on evolutionary multiobjective optimization. http://www.lania.mx/˜ccoello/EMOO/EMOObib.html.

[10] C. A. Coello. An updated survey of evolutionary multiobjective optimization techniques : State of the art and future trends. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999)*, pages 3–13, Washington, USA, 1999. IEEE Press.

[11] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

[12] K. Deb, S. Agrawal, A. 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*, pages 849–858. Springer. Lecture Notes in Computer Science No. 1917, 2000.

[13] K. Deb and T. 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*, pages 67–81. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[14] K. Downing. Using evolutionary computational techniques in environmental modelling. *Environmental Modelling and Software*, 13:519–528, 1998.

[15] 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*, 9:135–148, 2001.

[16] 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*, pages 149–156, Madrid, Spain, 2000.

[17] 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*, pages 512–520. The MIT Press, Cambridge, MA, 1996.

[18] C. M. Fonseca and P. 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):26–37, 1998.

[19] 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*, 6:1341–1352, 1998.

[20] N. F. Guler, E. D. Ubeyli, and I. Guler. Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Systems with Applications*, 29:506–514, 2005.

[21] M. Hapke, A. Jaszkiewicz, and R. Slowinski. Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, 6(3):329–345, August 2000.

[22] S. Haykin. *Neural Networks. A Comprehensive Foundation*. Prentice Hall, 1999.

[23] M. Hernandez-Guia, R. Mulet, and S. Rodriguez-Perez. A new simulated annealing algorithm for the multiple sequence alignment problem: The approach of polymers in a random media. *Physical Review E*, 72(3):1–7, 2005.

[24] W. Jiang, Q. Guo-Dong, and D. Bin. Observer-based robust adaptive variable universe fuzzy control for chaotic system. *Chaos, Solutions and Fractals*, 23:1013–1032, 2005.

[25] T. Jones. Crossover, macromutation and population-based search. In *Proceedings of the 6th Internatonal Conference on Genetic Algorithms (ICGA95)*, pages 73–80, San Francisco, USA, 2005. Morgan Kaufmann.

[26] H. Kantz. A robust method to estimate the maximal Lyapunov exponent of a time series. *Phys. Lett. A*, 185:77–87, 1994.

[27] D. Kim. Improving the fuzzy system performance by fuzzy system ensemble. *Fuzzy Sets and Systems*, 98:43–56, 1998.

[28] 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):205–224, 1995.

[29] L. W. Lee, L. H. Wang, and S. M. Chen. Temperature prediction and TAIFEX forecasting based on high-order fuzzy logical relationships and genetic simulated annealing techniques. *Expert Systems with Applications*, 34:328–336, 2008.

[30] B. Lillekjendlie, D. Kugiumtzis, and N. Christophersen. Chaotic time series. Part II: System identification and prediction. *Identification and Control*, 4(15):225–243, 1995.

[31] 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*, volume 2084, pages 725–732, 2001.

[32] S. Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 4(3):274–283, September 2000.

[33] S. Luke and L. Panait. A survey and comparison of tree generation algorithms. In Lee Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 81–88, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[34] M.W. Mak, K.W. Ku, and Y. L. Lu. On the improvement of the real time recurrent learning algorithm for recurrent neural networks. *Neurocomputing*, 24:13–36, 1999.

[35] M.A. Matos and P. Melo. Multiobjective reconfiguration for loss reduction and service restorating using simulated annealing. In *IEEE Budapest Power Tech'99*, pages 213–218, Budapest, Hungary, 1999. IEEE.

[36] Y. Mei-Ying and W. Xiao-Dong. Chaotic time series prediction using least squares support vector machines. *Chinese Physics*, 13:454–458, 2004.

[37] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Third edition, 1996.

[38] D. J. Montana. Strongly typed genetic programming. Technical Report #7866, Bolt Benanek and Newman, Inc., 10 Moulton Street, Cambridge, MA 02138, USA, 1993.

[39] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Proceedings of IEEE NNSP'97*, pages 24–26, Amelia Island, FL, USA, September 1997. IEEE Service Center.

[40] D. Nam and C. H. Park. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97, 2000.

[41] D. Nam and C. H. 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)*, volume 2, pages 522–526, Singapore, November 2002. Nanyang Technical University.

[42] R. Poli and N. F. McPhee. Exact GP schema theory for headless chicken crossover and subtree mutation. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1062–1069, Korea, 2001. IEEE Press.

[43] P. Potocnik and I. Grabec. Nonlinear model predictive control of a cutting process. *Neurocomputing*, 43:107–126, 2002.

[44] K. Rodriguez-Vazquez, 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):531–545, July 2004.

[45] 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*, 65:117–134, June 1993.

[46] J. J. Rowland. Model selection methodology in supervised learning with evolutionary computation. *BioSystems*, 72:187–196, 2003.

[47] A. E. Ruano, P. J. Fleming, C. Teixeira, K. Rodriguez-Vazquez, and C. M. Fonseca. Nonlinear identification of aircraft gas-turbine dynamics. *Neurocomputing*, 55:551–579, 2003.

[48] L. Sanchez, I. Couso, and J.A. Corrales. Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences*, 136(1–4):175–191, 2001.

[49] R. S. Sexton and J. N. D. Gupta. Comparative evaluation of genetic algorithm and backpropagation for training neural networks. *Information Sciences*, 129:45–59, 2000.

[50] 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*, 18:257–269, 2000.

[51] K. I. Smith, R. M. Everson, and J. E. Fieldsend. Dominance measures for multi-objective simulated annealing. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 1, pages 23–30, Portland, Oregon, USA, June 2004. IEEE Service Center.

[52] J. C. Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, 2003.

[53] C. S. Ting. An observer-based approach to controlling time-delay chaotic systems via Takagi-Sugeno fuzzy model. *Information Sciences*, 177(20):4314–4328, 2007.

[54] Z. Wei, W. Z. Ming, and Y. G. Ke. Genetic programming-based chaotic time series modeling. *Journal of Zhejiang University SCIENCE*, 5(11):1432–1439, 2004.

[55] A. Wolf, J. B. Switf, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica D*, 16:285–317, 1985.

[56] A. M. Woodward, R. J. Gilbert, and D. B. Kell. Genetic programming as an analytical tool for non-linear dielectric spectroscopy. *Bioelectrochemistry and Bioenergetics*, 48:389–396, 1999.

[57] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, 2000.

[58] 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, 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)*, pages 666–673, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[59] 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):117–132, April 2003.