Expert Systems with Applications 40 (2013) 4172–4182

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An approach to reservoir computing design and training

Aida A. Ferreira [a,b,*], Teresa B. Ludermir [a], Ronaldo R.B. de Aquino [a]

[a] *Federal University of Pernambuco (UFPE), P.O. Box 7851, Cidade Universitria, Cep: 50.740-530 Recife, PE, Brazil*
[b] *Federal Institute of Science, Technology and Education of Pernambuco, Av Professor Luis Freire, 500, Cidade Universitria, Cep: 50.740-530 Recife, PE, Brazil*

ARTICLE INFO

Keywords:
Reservoir computing
Echo state networks
Evolutionary algorithm

ABSTRACT

Reservoir computing is a framework for computation like a recurrent neural network that allows for the black box modeling of dynamical systems. In contrast to other recurrent neural network approaches, reservoir computing does not train the input and internal weights of the network, only the readout is trained. However it is necessary to adjust parameters to create a "good" reservoir for a given application. In this study we introduce a method, called RCDESIGN (reservoir computing and design training). RCDESIGN combines an evolutionary algorithm with reservoir computing and simultaneously looks for the best values of parameters, topology and weight matrices without rescaling the reservoir matrix by the spectral radius. The idea of adjust the spectral radius within the unit circle in the complex plane comes from the linear system theory. However, this argument does not necessarily apply to nonlinear systems, which is the case of reservoir computing. The results obtained with the proposed method are compared with results obtained by a genetic algorithm search for global parameters generation of reservoir computing. Four time series were used to validate RCDESIGN.

1. Introduction

Theoretically, recurrent neural networks (RNNs) are very powerful tools for solving complex temporal machine learning tasks. Nonetheless, several factors still hinder the larger scale deployment of RNNs in practical applications. There are few learning rules and most suffer from slow convergence rates, thus limiting their applicability (Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007). In 2001, a new approach to RNN design and training was proposed independently under the name of Liquid State Machines (Maass, Natschlager, & Markram, 2002) and under the name of Echo State Networks (Jaeger, 2001). This approach, which had predecessors in computational neuroscience (Dominey, 1995) and subsequent ramifications in machine learning as the Backpropagation–Decorrelation (Steil, 2004) learning rule, is now often referred to as reservoir computing (RC) (Lukosevicius & Jaeger, 2009). The basic concept is to randomly construct an RNN and leave the weights unchanged. A separate linear regression function is trained on the reservoir's response to the input signals using a linear regression. The underlying idea is that a randomly constructed reservoir offers a complex nonlinear dynamic transformation of the input signals which allows the readout to extract the desired output using a simple linear mapping. RC offers an intuitive methodology for using the temporal processing power of RNNs without the hassle of training them (Schrauwen, Defour, Verstraeten, & Campenhout, 2007b).

This study's proposal is an approach to reservoir computing design and training using an evolutionary strategy. Although the RC optimization is a challenge, on the other hand, checking the performance of a result system is relatively inexpensive. This makes evolutionary methods, for reservoir pre–training, a natural strategy for searching the best model for any task (Lukosevicius & Jaeger, 2009). Several evolutionary approaches to ESN reservoir optimization have been presented (Bush & Tsendjav, 2005; Ishii, van der Zant, Becanovic, & Ploger, 2004), however, they used the idea of separating the topology and reservoir weights in order to reduce the search space and they also used the condition of search for a spectral radius smaller than 1 to guarantee the echo state property.

The evolutionary strategy, adopted in this work, simultaneously searches for the best values of the reservoir global parameters, the best topology and the reservoir matrix, without the limitation shown on previous works of reducing the search space and without rescaling the matrices by the spectral radius. All experiments were implemented in MATLAB and RCToolbox (Schrauwen, Verstraeten, & Haene, 2007a). The genetic vector used in the proposed evolutionary algorithm is bigger than the other evolutionary approaches on ESN reservoir optimization.

This study is organized as follows. In Section 2 we provide an overview of reservoir computing. The echo state property is explained in Section 3. Section 4 presents the motivation and the

* Corresponding author at: Federal University of Pernambuco (UFPE), P.O. Box 7851, Cidade Universitria, Cep: 50.740-530 Recife, PE, Brazil. Tel.: +55 81 21268986.

E-mail addresses: aidaaf@gmail.com (A.A. Ferreira), tbl@cin.ufpe.br (T.B. Ludermir), rrba@ufpe.br (R.R.B. de Aquino).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.01.029

proposed evolutionary strategy developed for optimizing the search of global parameters, topology and reservoir weights. Section 5 presents four time series that were chosen to test the search method. The experimental results are presented in Section 6 and conclusions are described in Section 7.

2. Reservoir computing

In RC, the reservoir is a fixed random dynamical system, such as a recurrent network used in this work, that receives time-varying input on which certain computations are to be performed. This study uses the echo state network (ESN) approach as a learning system for time series forecasting. The ESN has K input units, N internal units and L output units (see Fig. 1). All K units are connected to all N internal units which in turn are all connected to the L output units. The key idea of ESNs consists of teaching only the weights of the outgoing connections from N to L. The internal topology and weights remain unchanged during teaching. Also, the output neurons can be fed back to the reservoir, which leads to a fast training procedure of the output weights. Furthermore, the resulting system deals well with time-dependent information (Ishii et al., 2004).

An ESN is composed of a reservoir and of a linear readout output layer which maps the reservoir states to the desired output (Antonelo, Schrauwen, & Stroobandt, 2008). The general state update Eq. (1) and the readout output Eq. (2) are as follows:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{back}\mathbf{y}(n) + \mathbf{W}^{bias}), \quad (1)$$

$$\mathbf{y}(n+1) = \mathbf{f}^{out}(\mathbf{W}^{inout}\mathbf{u}(n+1) + \mathbf{W}^{out}\mathbf{x}(n+1) + \mathbf{W}^{outout}\mathbf{y}(n) + \mathbf{W}^{biasout}), \quad (2)$$

where $\mathbf{u}(n)$ denotes the input at time n; $\mathbf{x}(n)$ represents the reservoir state; $\mathbf{y}(n)$ is the output; and $f()$ = *tanh*() or *sign ()* is the activation function used for ESNs in this study. All weight matrices to the reservoir (denoted by $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{back}$ and $\mathbf{W}^{bias}$) are randomly initialized, in the traditional method, from a uniform distribution with mean 0 and variance 1. They are then rescaled by, first, dividing the matrix by a spectral radius (the largest absolute eigenvalue) and then multiplying by a factor chosen by the specialist (generally near 1). In this work we used a new strategy to find these values; the evolutionary method looks for the best values for the W without rescaling the matrices by the spectral radius.

All connections to the output ($\mathbf{W}^{inout}$, $\mathbf{W}^{out}$, $\mathbf{W}^{outout}$ and $\mathbf{W}^{biasout}$) are trained. Here we have two possibilities: train the readout using pseudo–inverse or using ridge regress (Lukosevicius & Jaeger, 2009).

The leak rate α, can effectively tune the dynamics of the reservoir. If the leak state is chosen correctly, the reservoir dynamics can be adjusted to match the timescale of the input flow, making it possible to achieve an enhanced performance (Antonelo et al., 2008). The leak rate can be chosen empirically, but we chose to use the evolutionary strategy in order to find the best value for each data set. This addition to the basic units is called *leaky integrator neurons* (Lukosevicius & Jaeger, 2009) and Eq. (3) shows the changes.

$$\mathbf{x}(n+1) = \mathbf{f}((1-\alpha)\mathbf{x}(n) + \alpha(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}^{bias})). \quad (3)$$

In the RC Toolbox, which we used in our study, the topology of the system is completely defined by the topology structure. This structure can be depicted graphically as a single large matrix, it is possible to connect inputs, reservoir, outputs and bias on the one hand with reservoir and outputs on the other hand (Schrauwen et al., 2007a).

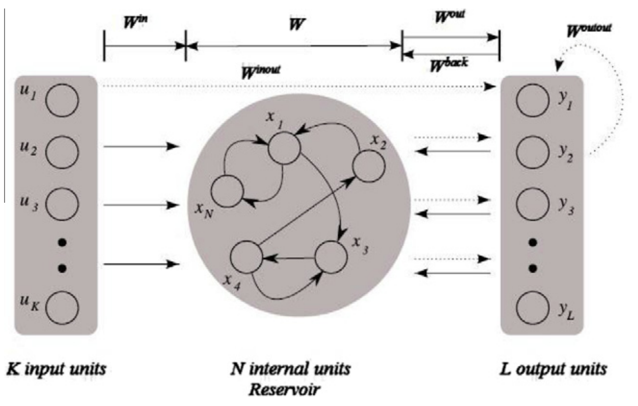


Fig. 1. The ESN. Dashed arrows indicate connections that are possible but not required.

3. Echo state property

Jaeger describes in Jaeger (2001) the conditions necessary for a reservoir, built with sigmoid neurons and output function tanh, that has the *Echo State Property*. The *Echo State Property* says that the activation state of an RNN, $\mathbf{x}(n)$, is a function of input history ($\mathbf{u}(n)$, $\mathbf{u}(n-1)$,...) presented to the network. More specifically, in certain conditions, there is an *echo* function $\mathbf{E} = (e_1,\ldots,e_n)$, where $e_i : U^{-N} \rightarrow \Re$, in which, for all history entries $(\ldots, \mathbf{u}(n-1), \mathbf{u}(n) \in U^{-N})$, the network state is given by Eq. (4):

$$\mathbf{x}(n) = \mathbf{E}(\ldots, \mathbf{u}(n-1), \mathbf{u}(n)). \quad (4)$$

The conditions described by Jaeger in Jaeger (2001) and in Jaeger (2002) are based on the largest singular value, $\sigma = \sigma_{\max}(\mathbf{W})$, and in the spectral radius, $\rho = |\lambda_{\max}|(\mathbf{W})$, of the reservoir matrix ($\mathbf{W}$). The first condition, considered too restrictive by Jaeger, says that in order to have the *Echo State Property*, the largest singular value (σ) must be smaller than 1, $\sigma < 1$. The second condition, known as $\rho > 1$, says that if the spectral radius (ρ) is greater than 1, the network has an asymptotically unstable null state thus, lacking the *Echo State Property* for any input set U containing 0 and admissible state sets.

The second condition is described in Verstraeten et al. (2007) as $\rho < 1$, and means that all the eigenvalues of $\mathbf{W}$ should be inside a circle unit in the complex plane. This condition expresses that the reservoir is locally and asymptotically stable around the origin and is a necessary but not sufficient to guarantee the *Echo State Property*.

The bounds described above supply a measure for the computational quality of a reservoir and can be deduced from the reservoir's weight matrix, without simulating it explicitly on some input signals. However, it is not clear how these bounds are related to the actual network dynamics (Verstraeten et al., 2007).

Only the spectral radius $\rho < 1$ was used in Jaeger (2002) to suggest a heuristic method of constructing reservoirs: the idea is to start with a randomly connected network and to rescale the weights so that $\rho < 1$ is close to, but smaller than 1 (Verstraeten et al., 2007). Although Jaeger has proposed a heuristic method in Jaeger (2002), he also highlighted that networks lacking the *Echo State Property* can sometimes be transformed into networks with *Echo State Properties* through the inputs and useful components that can be observed in biological neural networks.

The spectral radius is an important parameter that controls the reservoir dynamics. According to Verstraeten and Schrauwen (2009), the reservoir can be approximated as a linear time-invariant, discrete-time system:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \quad (5)$$

$$\mathbf{y}[k+1] = \mathbf{C}\mathbf{x}[k+1] + \mathbf{D}\mathbf{u}[k+1], \quad (6)$$

where $\mathbf{x}[k]$ represents the state of the reservoir (the vector of neuron activations) at time k, and $\mathbf{u}[k]$ and $\mathbf{y}[k]$ represent the input and output to the system, respectively. Matrix **A** contains the internal weights of the reservoir (**W**), matrix **B** contains the input-to-reservoir weights ($\mathbf{W}^{in}$), and **C** and **D** contain the (trained) reservoir-to-output ($\mathbf{W}^{back}$) and input-to-output ($\mathbf{W}^{inout}$) weights respectively. According to Verstraeten and Schrauwen (2009), matrix **D** is usually equal to zero. As we can see in the Eqs. (5) and (6), this model does not consider the feedback connection and bias connections (Ozturk, Xu, & Principe, 2007).

4. Motivation and proposed evolutionary strategy

The evolutionary are very efficient in searching optimal solutions (or at least approximately optimal solutions) in a wide variety of problems because they do not impose many of the limitations found in the traditional methods (Holland, 1992). Therefore, we decided to investigate their use in order to optimize the choice of the best ESN for problem solving. Usually, the search for those parameters is carried out in an exhausting way or through random experiments, which in general takes a long time to be accomplished and uses up a lot of computational resource.

The problem of finding a good ESN, using an evolutionary algorithm, can be approached in three different ways: as an evolutionary process that operates directly on the topology of the network; as an evolutionary operator that works on the stochastic parameters which generate the ESN; or both at the same time. If the evolutionary method operates directly on the connections (first and third approaches) the search spaces could be very large. For a network with N neurons the genetic string would be in the order of N^2 neurons large and specially tailored operators would be necessary to combine individuals of different sizes (Ishii et al., 2004). The second approach, called *AG Search* in this work, results in a smaller search space for the evolutionary process. One problem with this approach could be the imprecise investigation for the global parameters, specially because the network interconnections and weights are randomly generated. On the other hand, the strings and the search space are significantly smaller than in the direct approach (Ishii et al., 2004).

Ishii et al. used in Ishii et al. (2004) several evolutionary approaches for optimizing ESNs applied to modeling the motion of an underwater robot. The first approach was an evolutionary search of the parameters for reservoir matrix (**W**) generation: reservoir size, spectral radius and the connection density of **W**. Then an evolutionary algorithm was used on individuals consisting of all the weight matrices of small reservoirs (with 5 neurons). A variant, with a reduced search space, was also tried where the weights, but not the topology, of **W**, were explored, i.e., elements of **W** that were zero initially always stayed zero. According to Ishii et al. (2004) the empirical results showed the method's superiority over other state-of-art methods, and that the first approach was a little better than the others.

Bush and Tsendjav presented in Bush and Tsendjav (2005) another approach for optimizing the reservoir matrix through an evolutionary search applied to predicting the behavior of a mass–spring–damper system. They used the same idea of separating the topology and weight sizes of **W** in order to reduce the search space, but the search was restricted to the connection topology. This approach also demonstrated yielding on average 50% less (and much more stable) error in predicting the behavior of a mass–spring–damper system with small (20 neurons) reservoirs than without the genetic optimization.

Ferreira and Ludermir used the evolutionary approach to reservoirs optimization in Ferreira and Ludermir (2009). Again the idea of separating the topology and the weights of the reservoir to reduce the search space was used. The genetic algorithm was applied to search for the best configuration of some ESN generation parameters (number of neurons in the reservoir, activation function, spectral radius, etc.) to predict different sets of average wind speeds. The results showed that the genetic search spent only 20% of the time needed for an exhaustive search of the same parameters.

Although the strategy of tracking down interconnections and weights has been avoided until now due to the large search space problem, we decided to investigate the third approach and we are obtaining good results.

Another motivation for our proposed method is based on what was said by Ozturk and Principe in Ozturk and Principe (2005). In this work, they claim that the idea to rescale the reservoir matrix to adjust its spectral radius within a unit circle in the complex plane comes from the theory of linear systems. To get useful answers in linear systems this condition is necessary. However, this argument does not necessarily apply to the nonlinear systems, which is the case of reservoir computing. Ozturk and Principe also showed that a nonlinear system can be unstable, due to the condition of the spectral radius greater than 1, and still be able to have its dynamics controlled by the input, when it is applied (Ozturk & Principe, 2005).

Thus, this paper presents a method that simultaneously seeks by global parameters of reservoir computing, by the reservoir topology and weights, where the reservoir dynamic is not only determined by its fixed weights and the input signals influence this dynamic. The method does not consider the approach of reservoir computing with linear systems, because it does not rescale the reservoir matrix. The method is called RCDESIGN (reservoir computing and design training). Ferreira and Ludermir showed in Ferreira and Ludermir (2010) that the utilization of an evolutionary method for simultaneous optimization of parameters, topology and reservoir weights in Echo State Networks was very promising. They applied the idea in two time series. In this study, we present the RCDESIGN method and the results of apply it into four benchmark time series.

4.1. RCDESIGN–proposed method

The evolutionary algorithm incorporates specific knowledge about the problem's domain in order to accomplish the optimization process. It tolerates several non–determinant elements which help the search escape from the local minima. In addition to that, it has an appropriate cost function for each problem, making it widely useful. For our study, we created a fitness function which tries to play the GL criterion presented in Proben1 (Prechelt, 1994), i.e., the fitness function takes into account both the performance in the training set and in the validation set. GL criterion minimizes the chances of overfitting: for two networks trained on the same problem the one with the larger training set error may actually be better since the other one has concentrated on peculiarities of the training set at the cost of losing much of the regularities needed for good generalization. The fitness that we choose for our study allows the method to look for the best network (minor error in the training set) which retains its capability of generalization in the validation set. The fitness function is shown in Eq. (7).

$$f = \overline{MSE}_{Train} + \|\overline{MSE}_{Train} - \overline{MSE}_{Valid}\|, \tag{7}$$

where f is the value to be minimized by the evolutionary algorithm and the MSE (Mean-square Error) is calculated as in Eq. (8). Since RCDESIGN uses 10-fold cross validation, $\overline{MSE}_{Train}$ is the average of MSE in training set and $\overline{MSE}_{Valid}$ is the average of MSE in validation set.

$$\mathbf{MSE} = 100 \cdot \frac{1}{P \cdot N} \sum_{i=1}^{P} \sum_{j=1}^{N} (L_{ij} - T_{ij})^2, \tag{8}$$

where P is the total number of patterns in the set, N is the number of output units of the ESN, L_{ij} and T_{ij} are actual and desired (target) outputs of the $i-th$ neuron in the output layer, respectively.

The search process of the evolutionary algorithm involves a sequence of iterations, where a set of solutions passes through the selection processes and reproduction. To create the next generation, the algorithm selects certain individuals in the current population, called parents, and uses them to create individuals in the next generation, called children. Typically, the algorithm is more likely to select parents that have better fitness values. The selection function chooses parents for the next generation based on their scaled values from the fitness scaling function. An individual can be selected more than once as a parent, in which case, it contributes its genes to more than one child. In this study, the selection option was *stochastic uniform* (Baker, 1987). In this case, the parents are chosen by a line in which each parent corresponds to a section of the line whose length is proportional to its scaled value. The algorithm moves along the line in equal sized steps. At each step, the algorithm allocates a parent from the section it lands on. The first step is an uniform random number, smaller than the step size.

The evolutionary algorithm creates three types of children for the next generation: *Elite children* are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation. *Crossover children* is created by combining the vectors of a pair of parents. *Mutation children* is created by introducing random changes, or mutations, to a single parent.

P^g is a group (population) of s^i vectors, where g represents a genetic algorithm generation and s^i represents an individual in the population. The maximum value for g is the maximum number of generations (N_G) and the size of set P^g is defined by the T_P parameter of the algorithm. The s^i_j notation signals a j characteristic (gene) for the individual denoted as i.

- s^i_1 (η) – Size of $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{bias}$ and $\mathbf{W}^{back}$, varying between [50; 200].
- s^i_2 – If 1, there is $\mathbf{W}^{inout}$, 0, there is no connection.
- s^i_3 – If 1, there is $\mathbf{W}^{biasout}$, 0, there is no connection.
- s^i_4 – If 1, there is $\mathbf{W}^{outout}$, 0, there is no connection.
- s^i_5 – If 1, there is $\mathbf{W}^{bias}$, 0, there is no connection.
- s^i_6 – If 1, there is $\mathbf{W}^{back}$, 0, there is no connection.
- s^i_7 – If 1, the activation function is *tanh ()*; if 2, the activation function is *sign ()*.
- s^i_8 – If 1, the readout training function is *pseudo–inverse*; if 2, it is *ridge regress*.
- s^i_9 – Leak rate value, varying between [0.1; 1].
- s^i_{10} – Regularization parameter value, varying between [10^{-8}; 10^{-1}].
- $s^i_{11} \ldots s^i_{(\eta^2+3\eta+10)}$ – Weights of $\mathbf{W}$, $\mathbf{W}^{bias}$, $\mathbf{W}^{in}$ and $\mathbf{W}^{back}$, varying between [−1; 1].

Fig. 2 shows the conceptual division of s^i.

4.1.1. Pseudo-code of RCDESIGN

Randomized experiments were conducted to define the parameters of genetic algorithm used by RCDESIGN. The configuration parameters that showed a lower error in the experimental tests was chosen to be used in our experiments. The number of generation (N_G) is 10, the population size (T_P) is 120, the selection option is *stochastic uniform*, the number of elite individuals is 2, the crossover fraction is 0.8 (which means 96 individuals, and the remaining 22 individuals are the mutation children).

The following outline summarizes the pseudo-code of the proposed evolutionary method used to find the best global parameters, topology and reservoir weights at the same time:

Algorithm 1. RCDESIGN

```
Create a new random population $P^0$ (size of $T_P$);
Call CreateIndividuals;
$n = 1$;
while ($n \leq N_G$) and (not converge) do
    Select parents from $P^{n-1}$;
    Create population $P^n$ by elite, crossover and mutation;
    Call CreateIndividuals;
    $n = n + 1$;
Return best solution from $P^n$;
Create a network following best solution;
Compute test errors;
```

Algorithm 2. *CreateIndividuals*

```
$i = 1$;
while $i \leq T_P$ do
    Create a network following $s^i$;
    $fold = 1$;
    while $fold \leq 10$ do
        Create training and validation set;
        Simulate network in training set;
        Training readout;
        Compute training errors;
        Compute validation errors;
        $fold = fold + 1$;
    Compute fitness of $s^i$;
    $i = i + 1$;
```

The crossover operator used by RCDESIGN is an adaptation of the *uniform crossover* for populations of individuals of different sizes. For each pair of parents, the vector with the biggest dimension is defined as s^A and the smallest one as s^B. The s^{kid} elements are combined from a randomly generated mask, s^{mask}. The mask

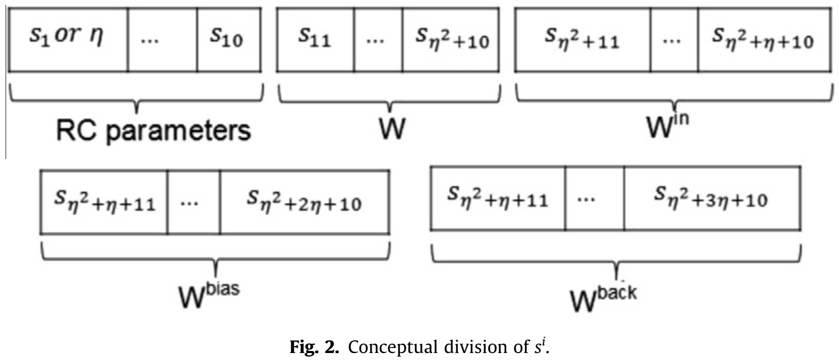


Fig. 2. Conceptual division of s^i.

indicates which s^{kid} characteristics will be inherited from s^A and which characteristics will be inherited from s^B. Fig. 3 illustrates how the crossover operator works.

The mutation operator used by RCDESIGN is also an adaptation from the mutation to different sized individual populations. For each selected individual with a parent (s^A), an s^{mask} mask is created. The s_1 gene is copied to s^{kid}. The mutation may occur starting from the second gene until the last gene with a mutation level defined by the parameter established as T_M.

4.2. AG Search–method for global search

Aiming to compare the results obtained with the RCDESIGN method, we conducted experiments with genetic algorithm search for global parameters generation of reservoir computing. *AG Search* uses the heuristic method of ESNs creation (Jaeger, 2002) and the idea of rescaling the **W** matrix by a spectral radius and a fixed topology with only the required RC connections. *AG Search* uses GA in order to search the principal RC generation parameters, which are the reservoir size, the spectral radius, and the density of interconnections of **W**. Neurons use *tanh ()* and the readout function is the *pseudo-inverse* function. The s_j^i notation signals a j characteristic (gene) for the individual denoted as i.

For *AG Search* experiments we used the same parameters of RCDESIGN, i.e., $N_G = 10$, $T_P = 120$, number of elite individuals equal 2, crossover fraction equal 0.8 and selection option was the *stochastic uniform*.

- s_1^i (η) – Reservoir size, varying between [50; 200].
- s_2^i – Spectral radius, varying between [0.7; 1].
- s_3^i – Density of interconnections, varying between [10%, 100%].

The pseudo-code for *AG Search* is equal in Subsection 4.1.1, the difference is in ***CreateIndividuals***. The following outline summarizes ***CreateIndividuals*** used by *AG Search*:

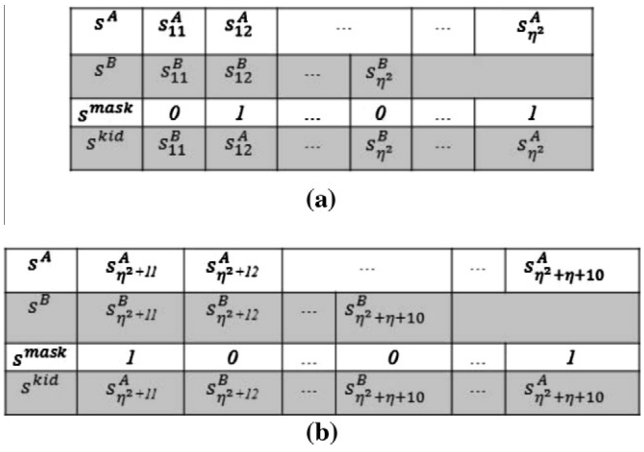
Algorithm 3. *CreateIndividuals*

```
$i = 1$;
while $i \leq T_P$ do
    Create a network following $s^i$;
    Compute all eigenvalues of $\mathbf{W}$;
    Divide $\mathbf{W}$ by max (eigenvalues);
    Multiply $\mathbf{W}$ by spectral radius $s_2^i$;
    $fold = 1$;
    while $fold \leq 10$ do
        Create training and validation set;
        Simulate network in training set;
        Training readout;
        Compute training errors;
        Compute validation errors;
        $fold = fold + 1$;
    Compute fitness of $s^i$;
    $i = i + 1$;
```

4.3. Comparison of computational complexity

Through the pseudo-code analysis proposed by RCDESIGN and *AG Search* method, three basic differences were noted among the two methods:

- Size of the individuals (T_i).
- Genetic operators' levels (T_{op}).
- Rescaling of the *Reservoir* matrix (**W**) by the spectral radius.



s^A	s^A_{11}	s^A_{12}	···		···	$s^A_{\eta^2}$
s^B	s^B_{11}	s^B_{12}	···	$s^B_{\eta^2}$		
s^{mask}	0	1	...	0	...	1
s^{kid}	s^B_{11}	s^A_{12}	···	$s^B_{\eta^2}$	···	$s^A_{\eta^2}$

(a)

s^A	$s^A_{\eta^2+11}$	$s^A_{\eta^2+12}$	···		···	$s^A_{\eta^2+\eta+10}$
s^B	$s^B_{\eta^2+11}$	$s^B_{\eta^2+12}$	···	$s^B_{\eta^2+\eta+10}$		
s^{mask}	1	0	...	0	...	1
s^{kid}	$s^A_{\eta^2+11}$	$s^B_{\eta^2+12}$	···	$s^B_{\eta^2+\eta+10}$	···	$s^A_{\eta^2+\eta+10}$

(b)

Fig. 3. Crossover of RCDESIGN. (A) Crossover's first part. (B) Crossover's second part.

A genetic algorithm's complexity depends on the size of the individual (T_i), on the number of generations (N_G), on the population size (T_P) and on the probability of the genetic operators (T_{op}). The complexity of the population generation procedure can be defined, in a simplified way, as O (T_PT_i) and the reproduction procedure's complexity as O ($N_GT_PT_iT_{op}$). The difference in the size of the individuals in the two methods implies a difference in complexity of the initial population generation procedures and also in the reproduction operations (crossover and mutation).

Rescaling the *Reservoir* matrix by the spectral radius incurs a high computational cost since the process involves the calculation of all eigenvalues of the *Reservoir* matrix, the dividing of *Reservoir* matrix by the highest absolute value among its eigenvalues and finally the multiplying of *Reservoir* matrix by the spectral radius. For the calculation of all eigenvalues, the complexity is equal to O (n^4) (Anderson et al., 1999). So we defined, in a simplified manner, that the complexity of rescaling the *Reservoir* matrix is O ($N_GT_Pn^4$). Table 1 presents a summary of the basic differences among the three methods in terms of time complexity.

According to Table 1, assuming that a weights search method for RC was less costly in computational terms (Ishii et al., 2004), due to the large size of the search space, than the cost for RC generating parameter search methods is misguided. The cost associated to the large search space is compensated by the lack of cost associated to the rescaling of the *Reservoir* matrix by the spectral radius. The time complexity for each method will be presented with the average time spent by the method to create each network. The average time ($\bar{t}$) is calculated by Eq. (9):

$$\bar{t} = \left(\frac{1}{N_GT_P}\sum t_i\right), \tag{9}$$

where N_G stands for the number of all the generations executed by the algorithm, where T_P is the size of the population and t_i is the time spent by the algorithm since its creation until the last individual's evaluation.

Table 1
Computational Complexity.

	RCDESIGN	*AG search*
Creating the initial population	O (T_iT_P)	O (T_i T_P)
Reproduction	O ($N_GT_PT_iT_{op}$)	O ($N_GT_PT_i$ T_{op})
Rescale *Reservoir* matrix by spectral radius	–	O (N_GT_P n^4)

4.4. Persistence method

The persistence forecast method was used to compare the results obtained by the RCDESIGN and *AG Search* methods. The persistence method assumes that the conditions at the time of forecasting will not change. For example, if it is sunny and 40 °C today, the persistence method predicts that it will be sunny and 40 °C tomorrow. The persistence method works well when weather patterns change very little and features on the weather maps move very slowly. It also works well in places like Northeast Brazil, where summertime weather conditions vary little from day to day. The persistence forecast method tends to be used as a benchmark against which all other models are compared (Walson, 2005).

5. Data sets

The Echo State Networks, created in this study, were driven by input/output sequences. The experiments are performed in four prediction problems:

- A NARMA system.
- Mackey–Glass chaotic attractor time series (MGS).
- Average hourly wind speeds from the city of Triunfo (TRI).
- Average hourly wind speeds from the city of Belo Jardim (BJD).

5.1. A NARMA system

The NARMA (Nonlinear Autoregressive Moving Average) task was described in Jaeger (2003). A NARMA system is a discrete time system of the following configuration, showed in Eq. (10)

$$y(n) = f(y(n-1), y(n-2), \ldots, y(0), u(n), u(n-1), \ldots, u(0)), \quad (10)$$

where $y(n)$ is the system output at time n,$u(n)$ is the system input at time n, and f is an arbitrary vector-valued, possibly non-linear function. The distinctive property of these systems is that the current output depends both on the input and output history. Modeling these systems is, in general, quite difficult due to the arbitrary non-linearity and the possibility that the long memory (i.e. $y(n)$ might depend on many former inputs and outputs) (Liebald, 2004). As inputs we use $u(n)$ as random samples drawn from a uniform distribution over the interval [0, 1].

5.2. The Mackey–Glass chaotic attractor time series

The Mackey–Glass time series (MGS) is a standard benchmark test in the time series prediction community. This system has no inputs and only one output, which is computed as in Eq. (11):

$$y(n+1) = y(n) + \delta\left(\frac{0.2y\left(n-\frac{\tau}{\beta}\right)}{\left(1+y\left(n-\frac{\tau}{\beta}\right)\right)^{10}} - 0.1y(n)\right), \quad (11)$$

where δ is a stepsize parameter that arises from the discretization of the original continuous-time MG equation, and τ is a delay parameter that influences the degree of chaos the MGS features. We used the standard choice, $\tau = 17$, which produces a "mildly chaotic" behavior, and $\delta = 0.1$ with subsequent sub sampling by 10. It is important to note that the evolution of the MGS depends on the initial values of $y\left(y(0), \ldots y\left(\frac{\tau}{\beta}\right)\right)$ (Liebald, 2004). These were simply random samples drawn from a uniform distribution. Although the MGS output does not depend on any input sequence, we created a bias input of equal length, which is simply used by the network in its update procedure state.

5.3. Average hourly wind speeds from Triunfo and Belo Jardim Cities

In this research real data from the project SONDA (system of national organization of ambient data – http://sonda.ccst.inpe.br/) have been used in order to create the model. SONDA is a project from the National Institute of Space Research (INPE) for the implementation of physical infrastructure and human resources in order to gather and improve the resources database of solar and wind energy in Brazil.

The wind power model in energy planning is based on statistical operation of the wind farms, considering the wind regime. For the Northeast Region of Brazil, the most important characteristic is the wind speed, as shown in Aquino et al. (2010). We chose two wind speed time series to carry out the experiments. The series were used previously in Ferreira and Ludermir (2009); Ferreira, Ludermir, de Aquino, Lira, and Neto (2008); Ferreira and Ludermir (2008). The first series consists of the average hourly wind speeds obtained by the wind headquarters in Triunfo. This series is made up of the average hourly wind speeds in the period from January 01, 2006 to April 30, 2007, which amounts to 11,640 patterns. The second series consists of the average hourly wind speed obtained by the wind headquarters of Belo Jardim in the period from July 31, 2004 to August 31, 2005, which amounts to 10,248 patterns.

Before creating the system, the base was preprocessed and the values of the average hourly speeds was transformed. The values were transformed as in Eq. (12):

$$x_{norm} = \frac{(y_{\max} - y_{\min}) \cdot (x - x_{\min})}{(x_{\max} - x_{\min})} + y_{\min}, \quad (12)$$

were, x_{norm} is the value transformed in a limited range [0,1]; $y_{\max}$, is the maximum value of the interval, 1; $y_{\min}$ is the minimum value of the interval, 0; $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of the series and; x is the original value. Through statistical analysis of the average hourly wind speed, we observed a high standard deviation of this variable, so we decided to apply an increase of 20% to the maximum value of the series, like we did in previous works Ferreira and Ludermir (2010); Ferreira and Ludermir (2009); Ferreira et al. (2008). Thus, the maximum value accepted by the model is equal to the maximum value found in the database increased by 20% and the minimum value accepted by the model is zero. The aim of the increase in speed supported by the forecasting model of average wind speeds is to make it more robust to support speeds higher than the speed shown by the historical series.

As in previous works Ferreira and Ludermir (2009); Ferreira et al. (2008); Ferreira and Ludermir (2008), a twenty-four-step-forward predictor of the average hourly wind speed was chosen. Since the series presented a good correlation index, it can be concluded that 24-step-forward is a good interval for the operation system planning. A twenty-four-step-forward predictor is actually 24 h predictor and thus predicting 1 day into the future.

6. Experimental results

In this section, we present experimental results that provide an overview of the proposed RCDESIGN efficiency. Four time series were used for evaluating the RCDESIGN method. We used the *AG Search* and *Persistence Method* to compare with the RCDESIGN results. All series were normalized to lie within the interval [0, 1] and divided in three sets: training set, validation set (which together make up 75% of the points), and test set (25% of the points). Because of the random characteristics of the methods, we performed 30 initializations the same in all data sets, so, we consider that an experiment consists of 30 initializations. The following outline summarizes the pseudo-code that controls the experiments:

Algorithm 4. Experiment

```
Create database;
Create database into two sets: training (75%) and testing (25%);
Create 10 fold (cross-validation) from training set;
Save folds and test set;
n = 1;
while (n ≤ 30) do
    Call RCDESIGN;
    Call AG Search;
Compute RCDESIGN performance;
Compute AG Search performance;
```

Table 2 presents the average number of generations, the average spectral radius and the average number of neurons in the reservoir for the two methods in all data sets for the test set. The values presented in the table is the average of each 30 initialization for each data set and the standard deviation is shown in parentheses.

The system performance was measured by the percentage of the MSE specified in Eq. (8), the NRMSE (normalized root mean squared error) specified in Eq. (13), the NMSE (normalized mean-square error) specified in Eq. (14), and for the wind speed data sets by the MAPE (mean absolute percentage error) specified in Eq. (15) and the MAE (mean absolute error) specified in Eq. (16):

$$\mathbf{NRMSE} = \frac{1}{N \cdot P} \sum_{i=1}^{P} \sum_{j=1}^{N} sqrt\left(\frac{(T_{ij} - L_{ij})^2}{var(T)} \right), \tag{13}$$

$$\mathbf{NMSE} = \frac{1}{N \cdot P} \sum_{i=1}^{P} \sum_{j=1}^{N} \frac{(T_{ij} - L_{ij})^2}{var(T)}, \tag{14}$$

$$\mathbf{MAPE} = 100 \cdot \frac{1}{P \cdot N} \sum_{i=1}^{P} \sum_{j=1}^{N} \left| \frac{T_{ij} - L_{ij}}{T_{ij}} \right|, \tag{15}$$

$$\mathbf{MAE} = \frac{1}{P \cdot N} \sum_{i=1}^{P} \sum_{j=1}^{N} (T_{ij} - L_{ij}), \tag{16}$$

where P is the total number of patterns in the set; N is the number of output units of the ESN; T_{ij} and L_{ij} are actual and desired output (target) of the $i - th$ neuron in the output layer; and $var(T)$ is the variance of the values in the target data set.

Table 3 presents MSE, NRMSE and NMSE of the two methods in all data sets for the test set. The values presented in the table is the average of each 30 initialization for each data set and the standard deviation is shown in parentheses. The errors obtained by RCDESIGN were much smaller than the errors obtained by the *AG Search*.

Table 4 shows the MAPE and MAE in the test for RCDESIGN, *AG Search* and *Persistence Method* only to data sets of average hourly wind speeds. The performance of RCDESIGN was well above the performance of the other two methods.

Table 2
Method's Parameters.

Data set	Generations	Spectral radius	Reservoir size
RCDESIGN			
NARMA	8.5	1.15(0.08)	174(16.72)
MGS	4	1.59(0.28)	82(31.20)
TRI	4.5	1.59(0.26)	86(25.91)
BJD	7.4	1.53(0.22)	79(23.09)
AG search			
NARMA	11	0.97(0.05)	196(3.50)
MGS	4	1.00(0.00)	189(8.15)
TRI	10.5	1.00(0.00)	173(21.46)
BJD	10.4	1.00(0.00)	177(23.11)

Table 3
Test – MSE, NRMSE and NMSE – 30 initializations.

	RCDESIGN	*AG search*
MSE		
Narma	0.00009201(0.0001)	0.00229437(0.0009)
MGS	0.00000001(0.0000)	0.00000161(0.0000)
TRI	0.00099199(0.0000)	0.00357787(0.0001)
BJD	0.00396731(0.0000)	0.01137915(0.0001)
NRMSE		
Narma	0.08462155(0,0354)	0.42436505(0.0438)
MGS	0.00050628(0,0001)	0.00584469(0.0006)
TRI	0.43600947(0,0016)	0.82799176(0.0103)
BJD	0.49230377(0,0025)	0.83375926(0.0039)
NMSE		
Narma	0.00837240(0.0089)	0.18194143(0.0395)
MGS	0.00000028(0.0000)	0.00003454(0.0000)
TRI	0.19010669(0.0014)	0.68567226(0.0170)
BJD	0.24236893(0.0024)	0.69516923(0.0065)

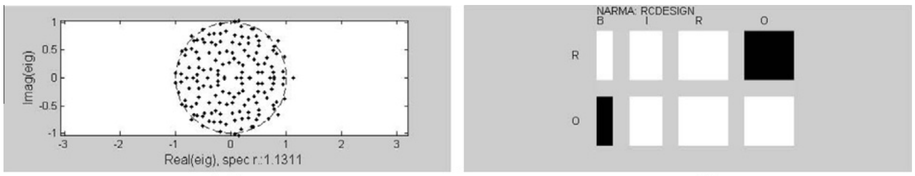
Table 4
Test – MAPE and MAE – 30 initializations.

	RCDESIGN	*AG search*	*Persistence Method*
MAPE (%)			
TRI	9.86(0.04)	20.63(0.24)	22.46(0.23)
BJD	12.08(0.11)	20.27(0.11)	23.99(0.25)
MAE (m/s)			
TRI	0.87(0.00)	1.71(0.02)	1.92(1.61)
BJD	0.62(0.00)	1.08(0.01)	1.22(0.96)

6.1. Narma

The Narma database has been used as a benchmark task of time series forecasting. Fig. 4 presents the topology and poles of the reservoir to reservoir connection matrix, in complex planes, for one of the systems created by RCDESIGN for the NARMA data set. In the topology matrix, the white square indicates the existence of a connection, on the one hand, between bias (b), inputs (i), reservoir (r) and outputs (o), on the one hand, to reservoir and, on the other hand, to outputs. The black square indicates the absence of a connection. As we can see in this figure, the network has connections (white rectangle) between bias and reservoir, input and reservoir, reservoir and reservoir, input and output, reservoir and output and feedback connection in the output layer, which makes this system very non-linear. Although RCDESIGN does not perform a search of the spectral radius, we can see it in this figure, from the view of the poles of the reservoir weights in the complex plane. The spectral radius was equal to 1.1311, staying above the limit suggested by Jaeger (2001). In spite of that, the created system has an excellent performance.

Table 5 presents the performance of networks created by RCDESIGN and a summary of the results obtained by other authors with the same data set. Comparisons between these methods must be made with caution, as the results are obtained with different experimental model setups. This table shows the best results in the test set in bold. Jaeger (2003) used a scheme to perform the online adaptation of output weights (*readout*), for NARMA data set, based on the RLS algorithm, and obtained a NMSE equal to 0.0081 in the test set in a network with a reservoir of 400–neuron reservoir. Steil (2004) performed experiments with APRL and BPDC algorithms for training systems for this base. The best result achieved by him in the test set was NMSE equal to 0.142 in a network with a 40–neuron reservoir, trained by BPDC. Verstraeten et al. presented in Verstraeten et al.



(a) (b)

Fig. 4. RCDESIGN – NARMA. (A) Poles of the matrix *W* matrix. (B) Topology (white square indicates the existence of a connection and the black square indicates the absence).

Table 5
NARMA: method's performance.

	NMSE	Spectral radius	Reservoir size
RCDESIGN	0.0084	1.15(0.08)	174(16.72)
Jaeger (2003)	0.0081	–	400
Steil (2004)	0.1420	–	40
	NRMSE	Spectral radius	Reservoir size
RCDESIGN	0.085	1.15(0.08)	174(16.72)
Verstraeten et al. (2007)	0.400	1	200

Table 6
MGS: method's performance.

	NMSE	Spectral radius	Reservoir size
RCDESIGN	0.00000028	1.59(0.28)	82(31.20)
Steil (2004)	0.0340	–	40
	NRMSE	Spectral radius	Reservoir size
RCDESIGN	0.00051	1.59(0.28)	82(31.20)
Jaeger and Haas (2004)	0.000063	–	1.000

(2007) a study on the relationship between the parameters of reservoir computing and its dynamics. They performed many different configurations of reservoir computing for the task of predicting the base NARMA. The best system created by them, obtained a NRMSE of approximately 0.4, in a network with a 200–neuron reservoir, linear active function and spectral radius close to 1. The RCDESIGN performance may not have been the best for the problem, but the result shows that it is possible to build ESNs with different recipes from the traditional ones which use spectral radius to rescale the reservoir's weight matrix.

6.2. Mackey–Glass

The task of predicting the Mackey–Glass base is also used as a benchmark in several studies and systems created with reservoir computing have produced excellent results for this base. Fig. 5 shows the architecture and the weights distribution for one of the systems created by RCDESIGN for this data set. The spectral radius of this system is equal to 1.651, which means it is well above the limit suggested by in Jaeger (2001)].

Table 6 presents the performance of networks created by RCDESIGN and a summary of the results obtained by other authors with the same data set. Comparisons between these methods must be made with caution, as the results are obtained with different experimental model setups. This table shows the best results in the test set in bold.

Steil (2004) performed experiments with APRL and BPDC algorithms in order to train systems for this base. The best result he achieved in the test set was NMSE equal to 0.034 in a network trained by BPDC and with 40 neurons in the reservoir. In Jaeger and Haas (2004) Jaeger and Haas performed experiments with this series and obtained a NRMSE of 0.000063 for reservoirs with 1000 neurons.

6.3. Triunfo

We also applied the proposed method in two real data sets (the average hourly wind speeds from the cities of Triunfo and Belo Jardim) in order to verify the feasibility of the proposed method applied to real data. Fig. 6 presents the architecture and the weights distribution for one of the systems created by RCDESIGN for this data set. The spectral radius is equal to 1.57.

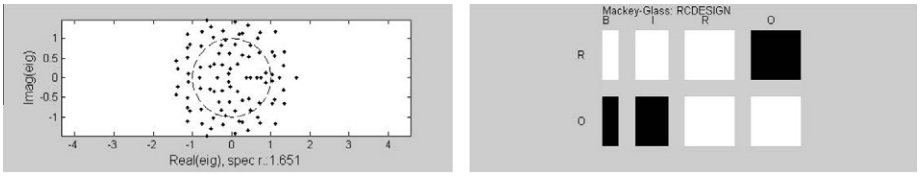
Tables 3 and 4 show the performance for forecasting the Triunfo data set. These tables show the results in the tests sets. As we can see in these tables, the MSE, the NRMSE, the NMSE, the MAPE and the MAE of the systems created by RCDESIGN were much lower than all of the sets in the system created by the *AG Search*. They were also smaller than the errors of the *Persistence Method*.

If we compare the results obtained by the model created by RCDESIGN with the results obtained by Ferreira et al. in Ferreira and Ludermir (2008), for the same data set, we can conclude that the RCDESIGN method was better than the one adopted in this study. The model created in Ferreira and Ludermir (2008) presented MSE and MAE equal to 0.95 and 0.88 in the test set. The model created in Ferreira and Ludermir (2008) used a network with 400 neurons. The model created in Ferreira and Ludermir (2009) presented MSE equal to 0.96 in the test set, with 547 neurons and spectral radius equal to 0.947.

6.4. Belo Jardim

The Fig. 7 presents the architecture and the weights distribution for one of the systems created by RCDESIGN for this data set. The spectral radius is equal to 1.5986.

Tables 3 and 4 show the performance for forecasting the Belo Jardim data set. These tables show the results in the tests sets.



(a) (b)

Fig. 5. RCDESIGN – Mackey–Glass. (A) Poles of the matrix *W* matrix. (B) Topology.

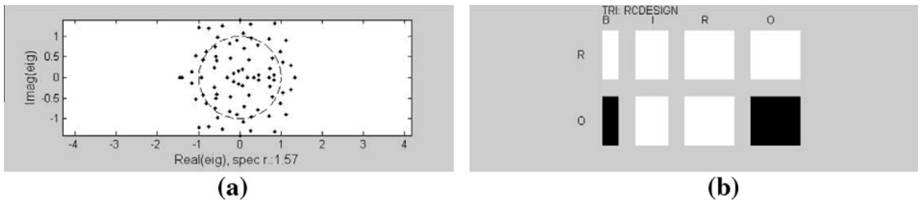


Fig. 6. RCDESIGN – Triunfo. (A) Poles of the matrix *W* matrix. (B) Topology.

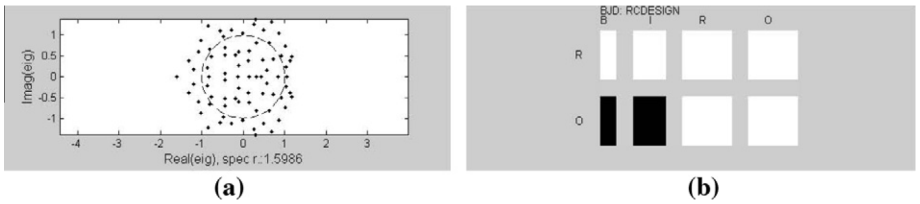


Fig. 7. RCDESIGN – Belo Jardim. (A) Poles of the matrix *W* matrix. (B) Topology.

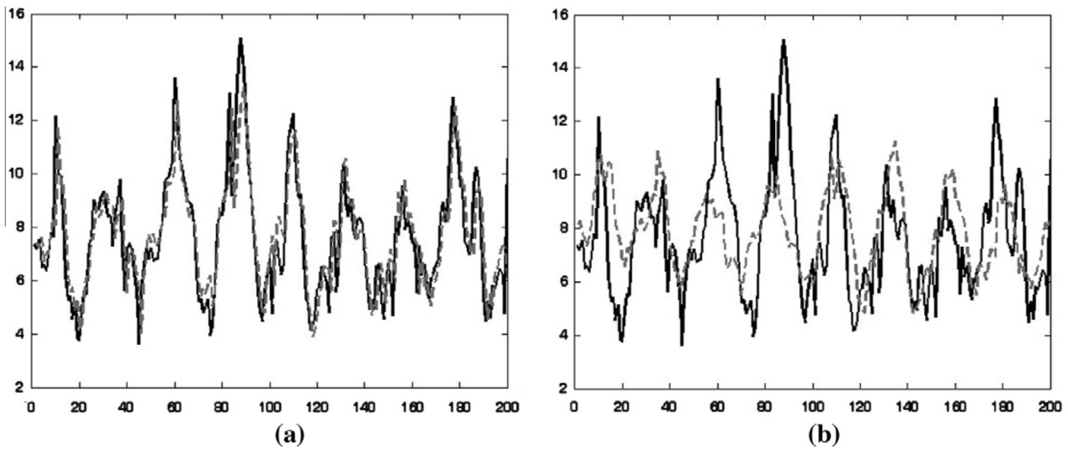


Fig. 8. Test patterns of Triunfo database. (A) Actual (solid) × RCDESIGN (dashed) and (B) Actual (solid) × *AG Search*(dashed). The vertical axis represents the average hourly wind speeds and the horizontal axis the sequence of the last 200 patterns.

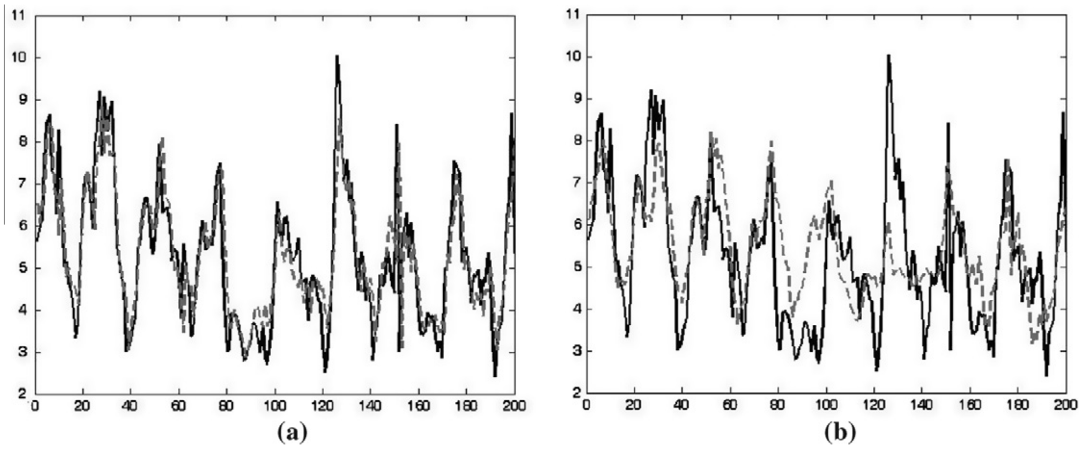


Fig. 9. Test patterns of Belo Jardim database. (A) Actual (solid) × RCDESIGN (dashed) and (B) Actual (solid) × *AG Search*(dashed). The vertical axis represents the average hourly wind speeds and the horizontal axis the sequence of the last 200 patterns.

Again, as we can see in these tables, the MSE, the NRMSE, the NMSE, the MAPE and the MAE of the systems created by RCDESIGN were much smaller than those in the systems created by the *AG Search*. They were also smaller than the errors of *Persistence Method*. The model created in Ferreira et al. (2008) presented, in the test set, MSE, MAE and MAE respectively equal to 4.11, 18.21 and 0.71

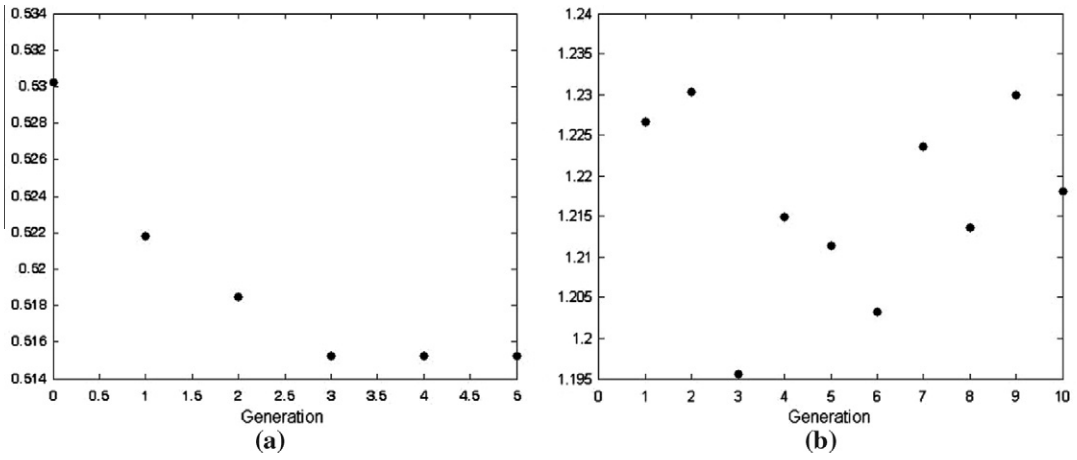


Fig. 10. Comparison of the fitness evolution for the Belo Jardim data set. RCDESIGN × *AG Search*. The vertical axis represents the fitness of the best individual in each generation and the horizontal axis represents the sequence of each generation.

Table 7
Average generating time for each network (in seconds) in each method.

	RCDESIGN	*AG search*
Narma	4.54(0.43)	5.83(0.45)
MGS	3.89(0.21)	5.58(0.47)
TRI	5.94(0.17)	7.76(0.16)
BJD	2.57(0.09)	3.62(0.11)

in a network with 200 neurons. The performance of the model created by RCDESIGN was much higher in the test sets with MSE, MAPE and MAE equal to 0.00396731, 12.08 and 0.62 in a much smaller network containing approximately 79 neurons.

Figs. 8 and 9 present the last 200 patterns, in the test set, referring to the prediction made by the RCDESIGN created model and the *AG Search* created model for the Triunfo and Belo Jardim database respectively. The solid line refers to the actual data, while the dashed line is the data predicted by the RCDESIGN created model or by the *AG Search* created model. As we can see, the model created by RCDESIGN was able to learn and identify most of the background presented in this series, while the model created by the *AG Search* had a much lower performance.

As we can see in Fig. 10 the best individual evolution in the RCDESIGN method is decreasing, but this decreasing behavior is not seen in the *AG Search*. The irregular behavior presented by the *AG Search* reinforce what Ozturk et al. demonstrated in Ozturk et al. (2007). In this article they proved that the reservoir dynamics is not determined by its fixed weights and that the input signals influence this dynamic. They showed that there are various weight matrices with the same spectral radius, and unfortunately, they do not display the same performance when the MSE is calculated. The same behavior was observed in the NARMA, Mackey–Glass and Triunfo bases.

6.5. Results of time complexity

Table 7 presents the average generating time for each network (in seconds) in each method. The values presented in the table is the average of each 30 initialization for each data set and the standard deviation is shown in parentheses. The average generating time for the RCDESIGN networks was consistently lower compared to the average *AG Search*, which means that the strategy of searching the *Reservoir* matrix weights has a lower computational cost than the strategy of searching the RC generation parameters, when this involves the rescaling of the *Reservoir* matrix by the spectral radius.

7. Conclusion

The RCDESIGN method combines the advantages of an evolutionary strategy with reservoir computing in order to generate an automatic process for producing ESNs. The most encouraging result proves which method is the best option since the choice does not depend on the analyst's experience. Through this method, it's possible to investigate a different topology, as well as evaluate many parameter combinations. Furthermore, the task's time–consumption and the computational effort in finding the parameters, topology and reservoir weights, were reduced.

According to Lukosevicius and Jaeger (2009), separating the RNN reservoir and readout provides a good platform for trying out several kinds of RNN methods in the reservoir. It is also useful for observing how much they can actually improve the performance on created RNNs. This is particularly well–suited for testing various biology-inspired RNN adaptation mechanisms, on how they can improve learning of a supervised task. Considering this, we developed the RCDESIGN method in order to analyze the effect that the search, for various parameters of the ESN in a single step, for a given task, could have on the system's final performance.

We showed that it is possible to optimize reservoir global parameters, topology and reservoir weights simultaneously, without the limitation (shown in previous works) of reducing the search space and without the spectral radius rescaling of the reservoir matrix. The proposed method displays very good performance in two real time series and good results in two benchmarks series. The search with RCDESIGN takes less time than *AG Search* in all the bases studied.

The results presented by the RCDESIGN created systems show that there is not a need for uniform distribution of the poles of the weights within a unit circle in the complex plane in order for the reservoir to perform well, since the real dynamics of the reservoir can only be found with the system in use. This is in line with what had been said previously in Ozturk and Principe (2005). Ozturk and Principe introduced in Ozturk and Principe (2005), a computational model for nonlinear systems with sigmoidal nonlinearity, which does not require global stability. In this model, although the autonomous system is unstable, the input signal forces the system dynamics to become "transiently stable".

We verified that the proposed method is good for forecasting four distinct time series. In future works, we plan to investigate the method in different kinds of data sets and analyze in more detail the effect of not rescaling the weight matrix by the spectral radius.

Acknowledgments

The authors thank CNPq, CAPES and FACEPE (Brazilian Research Agencies) for their financial support.

References

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., et al. (1999). *LAPACK user's guide*. SIAM<http://www.netlib.org/lapack/lug/>.

Antonelo, E. A., Schrauwen, B., & Stroobandt, D. (2008). Event detection and localization for small mobile robots using reservoir computing. *Neural Networks, 21*(6), 862–871.

Aquino, R. R. B., Junior, M. A. C., Lira, M. M. S., Neto, O.N., Almeida, G.J., & Tiburcio, S. N. N. (2010). Recurrent neural networks solving a real large scale mid-term scheduling for power plants. In *Proceedings of the international joint conference on neural networks – IJCNN 2010, Barcelona* (pp. 3439–3444).

Baker, J. E. (1987). Reducing bias and inefficienry in the selection algorithm. In *Proceedings of the 2nd international conference on genetic algorithms, Cambridge, MA, USA* (pp. 14–21).

Bush, K., & Tsendjav, B. (2005). Improving the richness of echo state features using next ascent local search. In *Proceedings of the artificial neural networks in engineering conference* (pp. 227–232).

Dominey, P. F. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics, 73*(3), 265–274.

Ferreira, A. A., & Ludermir, T. B. (2008). Using reservoir computing for forecasting time series: brazilian case study. In *Proceedings of the international conference on hybrid intelligent systems – HIS 2008,Los Alamitos, CA, USA* (pp. 602–607).

Ferreira, A.A., & Ludermir, T.B. (2009). Genetic algorithm for reservoir computing optimization. In *Proceedings of the international joint conference on neural networks – IJCNN 2009, Atlanta* (pp. 811–815).

Ferreira, A. A., & Ludermir, T. B. (2010). Evolutionary strategy for simultaneous optimization of parameters, topology and reservoir weights in echo state networks. In *Proceedings of the international joint conference on neural networks – IJCNN 2010, Barcelona* (pp. 1870–1876).

Ferreira, A. A., Ludermir, T. B., de Aquino, R. R. B., Lira, M. M., & Neto, O. N. (2008). Investigating the use of reservoir computing for forecasting the hourly wind speed in short-term. In *Proceedings of the international joint conference on neural networks – IJCNN 2008, Hong Kong* (pp. 1950–1957).

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology control and artificial intelligence*. Cambridge, MA, USA: MIT Press.

Ishii, K., van der Zant, T., Becanovic, V., & Ploger, P. (2004). Identification of motion with echo state network. In *Proceedings of the OCEANS 2004 MTS/IEEE–TECHNO-OCEAN 2004 conference* (Vol. 3, pp. 1205–1210).

Jaeger, H. (2001). The echo state approach to analyzing and training recurrent neural networks., Tech. rep., GDM 148, German national resource center for information technology.

Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the echo state network approach., Tech. rep., GDM 159, German national resource center for information technology.

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. *Advances in Neural Information Processing Systems, 15*, 593–600.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. *Science, 308*, 78–80.

Liebald, B. (2004). Exploration of effects of different network topologies on the ESN signal crosscorrelation matrix spectrum, *Master Dissertation*, Jacobs University Bermen, 2004.

Lukosevicius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review, 3*(3), 127–149.

Maass, W., Natschlager, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation, 14*(11), 2531–2560.

Ozturk, M. C., & Principe, J. C. (2005). Computing with transiently stable states. In *Proceedings of the international joint conference on neural networks, IJCNN 2005* (Vol. 3, pp. 1467–1472).

Ozturk, M. C., Xu, D., & Principe, J. C. (2007). Analysis and design of echo state networks. *Neural Computation, 19*(1), 111–138.

Prechelt, L. (1994). Proben1 – a set of neural network benchmark problems and benchmarking rules., Tech. rep., 21/94, Fakultt fr Informatik, Universitt Karlsruhe, Germany.

Schrauwen, B., Verstraeten, D., & Haene, M. D. (2007a). Reservoir computing toolbox manual, http://reslab.elis.ugent.be/rctoolbox.

Schrauwen, B., Defour, J., Verstraeten, D., & Campenhout, J. V. (2007). The introduction of time-scales in reservoir computing, applied to isolated digits recognition., Proceedings of the 17th international conference on artificial neural networks. In *Lecture notes in computer science* (Vol. 4668, Part I 471–479).

Steil, J. J. (2004). Backpropagation–decorrelation: Online recurrent learning with o(n) complexity. In *Proceedings of the international joint conference on neural networks, IJCNN 2004* (Vol. 2, pp. 843–848).

Verstraeten, D., & Schrauwen, B. (2009). On the quantification of dynamics in reservoir computing, Proceedings of the 19th International Conference on Artificial Neural Networks. In *Lecture notes in computer science* (Vol. 5768, pp. 985–994).

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks, 20*(3), 391–403.

Walson, S. (2005). Fresh forecasts [wind power forecasting]. *Power Engineer, 19*(2), 36–38.