2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Reservoir Computing Optimization with a Hybrid Method

Anderson T. Sergio, Teresa B. Ludermir

***Abstract*—Reservoir Computing (RC) is a paradigm of artificial neural networks with important applications in the real world. RC uses similar architecture to recurrent networks without the difficulty of training the network hidden layer (reservoir). However, RC can be computationally expensive and various parameters influence its efficiency, making it necessary to search for alternatives to increase its capacity. This work aims to use a hybrid algorithm between a PSO (Particle Swarm Optimization) extension and Simulated Annealing for optimize the global parameters, architecture and weights of RC, in time series forecasting. The results showed that the Reservoir Computing optimization with the hybrid algorithm achieved satisfactory performance in all databases investigated and outperformed original APSO (Adaptive Particle Swarm Optimization) in some of them.**

Keywords—Reservoir Computing; PSO; optmization; time series forecasting.

I. INTRODUCTION

Reservoir Computing is a paradigm of Artificial Neural Networks with important applications [1] [2] [3]. RC uses similar architecture to Recurrent Neural Networks (RNN) for temporal processing without the difficulty of training the network hidden layer. Reservoir Computing was introduced independently as Liquid State Machines (LSM) [4] and Echo State Networks (ESN) [5]. In general, RC is based on building a random RNN (this layer is called the reservoir) without changing the weights. After this phase, a linear regression function is used to train the system output. Schrauwen et al. [6] show that the dynamic non-linear processing provided by the reservoir is sufficient for the output layer to be able to extract the output signals using a simple linear mapping. Usually, the output layer is called readout.

As in conventional neural networks, Reservoir Computing has some disadvantages. Since RC is an approach of recurrent networks, computational cost can be expensive, even without the training phase on the reservoir layer. Several parameters influence the RC performance; for instance, the number of nodes used and the type of activation function. Setting these parameters without experience is difficult. Lukosevicius and Jaeger [7] say that it is unlikely that the random generation of the weights and the training of the output layer using a simple linear regression function provide the ideal solution for RC.

Several works have sought to use optimization techniques to improve RC performance. For instance, Ferreira proposed a method using Genetic Algorithms [8] to optimize architecture, parameters and initial weights of the network hidden layer [9].

Particle Swarm Optimization [10] is an optimization algorithm that has some advantages over other global search techniques. PSO is based on the social behavior of flocks of birds: a population of solutions is maintained and each individual seeks to improve its performance based on its best experience and the best experience of the group. When compared to Genetic Algorithms, for example, PSO has simpler implementation and, in some cases, relatively fast convergence and low computational cost [11] [12].

Sergio and Ludermir [13] used PSO to optimize RC architecture and initial parameters. Subsequently, based on the work of Ferreira and his first approach, Sergio [14] also sought to optimize the weights of RC (along with the architecture and initial parameters) with PSO, and two of its extensions, applying the method in time series forecasting. The results showed that, taking into account the time series forecast error, the PSO extension known as APSO [15] achieved the best performance.

The literature contains several examples of using optimization techniques to improve performance over traditional architectures of neural networks. Among these, the work of Li and Liu [16] may be mentioned. The authors proposed a hybrid algorithm between a modified PSO, Simulated Annealing (SA) and an RBF network, named MPSO-SA-RNN (Modified PSO-SA algorithm and RBF Neural Network). Simulated Annealing, described independently in [17] and [18], is a probabilistic global search technique, based on an analogy with thermodynamics. The new method was used in the prediction of a variable that indicates the quality of polymers. The PSO was modified with features of SA and used to optimize the parameters of an RBF network. The results validated the proposed model and the confirmed advantages of MPSO-SA against original PSO and SA algorithms.

This work aims to investigate the use of a hybrid algorithm between PSO and SA in the task of optimizing the global parameters, architecture and weights of Reservoir Computing, in the problem of time series forecasting. To validate the proposed method, benchmark time series were tested.

Next, here is the structure of this work. Section II discusses the main concepts of Reservoir Computing and presents the PSO and SA algorithms, as well as the hybridization proposed by Li and Liu. Section III discusses RC optimization and Section IV describes the proposed method. Section V presents numerical simulations and, finally, Section VI describes the conclusions and proposals for future work.

Anderson T. Sergio and Teresa B. Ludermir are with Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Cidade Universitária - 50740-560 - Recife/PE, Brazil (e-mail: ats3@cin.ufpe.br, tpfl2@cin.ufpe.br, tbl@cin.ufpe.br).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

II. BACKGROUND

A. Reservoir Computing

Reservoir Computing is a paradigm of artificial neural networks developed independently as Liquid State Machine [4] and Echo State Network [5]. In common, all the Reservoir Computing approaches use the computational power of recurrent neural networks without training the hidden layer.

In general, RC is composed of a recurrent network with a relatively large number of processing units, called the reservoir. The reservoir receives the input signals and sends them to a smaller circuit called the readout. While the reservoir weights are set randomly at the beginning of the process and kept unchanged, the readout is used to train the network output through a function that does linear regression of signals from the hidden layer. Fig. 1 shows a diagram of a simple Echo State Network.

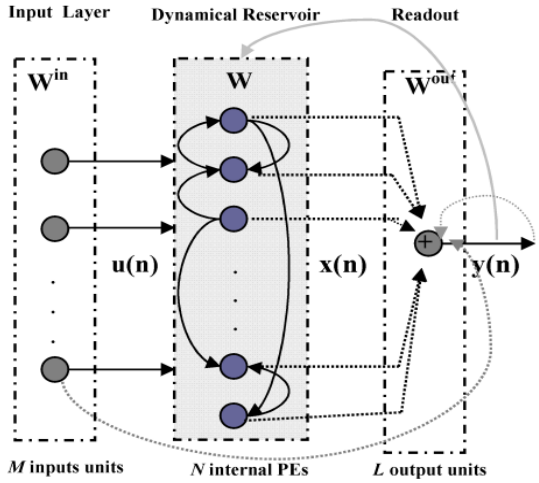

Fig. 1. Echo State Network architecture

As seen in Fig. 1, the reservoir layer receives signal values coming from the input layer, and optionally the signals from the feedback connection and bias. The reservoir, with a certain number of processing units (PEs), is designed with recurrent connections. The weights of these connections are random and do not change. The readout layer makes a simple linear mapping from the reservoir output using, for instance, pseudo-inverse. Also in Fig. 1, dotted lines represent connections that can be trained and shaded lines indicate optional connections.

Rather than trying to achieve a particular transformation by adjusting the weights, RC uses a larger number of neurons (compared to conventional networks) to achieve a diverse set of transformations from the input signals [5]. In general, such changes are not desired, but these changes can be combined in order to achieve these transformations. This action is held by the readout layer. The calculation is made simple since the readout has no feedback connections. It is important to note that the same reservoir can be used to calculate multiple transformations in parallel, since they have different readouts.

B. APSO and SA

PSO is a global optimization technique based on a population of solutions. The algorithm is based on the social behavior of a flock of birds, where an individual mimics the actions of the group's best (or most suitable) individual. The process starts defining the population of solutions. Each individual (particle) is a possible solution. Each particle has a position and speed, and the update process is based on its best experience and the best experience of the group. PSO was introduced by Kennedy and Eberhart in 1995 [10].

Works such as Van den Bergh [19], Clerc et al. [20] and Trelea [21] mathematically analyze the convergence of PSO. Such discussions had led guidance on which parameters affect the convergence, divergence, or oscillation of the algorithm, and these studies have led to several variations of the original PSO.

Adaptive Particle Swarm Optimization (APSO) [15] was developed trying to increase the search efficiency and the convergence speed of the original PSO algorithm. Basically, APSO has two main steps. First, after evaluating the population distribution and the fitness function of each particle, the algorithm identifies to which of the four evolutionary states the current generation belongs: exploration, exploitation, convergence or jump. With this knowledge, there is an automatic control of some variables, such as the term of inertia and acceleration coefficients. Second, an elitist strategy is performed (ELS - Elitist Learning Strategy), activated when the evolutionary state is classified as convergence. To classify the evolutionary state of the current generation, APSO uses an approach called ESE (Evolutionary State Estimation). ESE is based on search behavior and population distribution of the PSO solutions.

The main objective of ELS in APSO is to give the particle better overall performance and leaves out regions of local optima when the search is classified as convergence state. Granting a disturbance to the particle with better performance is important because this individual has no parameters to follow. If this disturbance finds a region with better results, the entire swarm will follow the example of the leading particle.

Simulated Annealing is a probabilistic global search technique, based on an analogy with thermodynamics - the gradual cooling of a physical system to reach a minimum potential energy. The method was described independently in 1983 [17] and 1985 [18].

In general, the algorithm starts with an initial solution S_0. T is the parameter that controls temperature, starting at T_0. The temperature of the system decreases until it reaches a thermal equilibrium which is a better solution. In this process, a new solution S_n is created in the neighborhood of the previous solution S. If the new solution fitness $f(S_n)$ is better than the prior value $f(S)$, the new solution is accepted. Otherwise the new solution is accepted according to a probability P given by equations 1 and 2.

$$P = exp\left(-\frac{\Delta}{T}\right) \tag{1}$$

$$\Delta = \frac{f(S_n) - f(S)}{f(S_n)} \, x \, 100 \tag{2}$$

Applying the P factor, the search tends to leave local minima. The algorithm repeats this process described in the previous paragraph until it reaches a desirable state.

If the initial solution is a point close to the global optimal, SA generally gives good results. However, sometimes it is impossible to start from a good initial solution. Then, a trial and error process is necessary. Also, the user must set certain heuristics. For example, the user must consider the way that new solutions will be proposed and what the appropriate stopping criterion will be.

Trying to achieve better efficiency in finding optimal solutions to a given problem, Li and Liu [15] proposed a hybrid algorithm between PSO and SA. One of the disadvantages of PSO is that after a few generations, the diversity of the swarm is reduced and the population may converge to a local optimal. Since SA can be more effective when an appropriate initial state is taken into account, the authors proposed to run this method after iterations of a modified PSO algorithm. The new algorithm is called MPSO-SA (Modified PSO-SA).

Regarding the original PSO, MPSO-SA has different update equations for the inertia term (momentum) and acceleration coefficients. The authors propose that the momentum should not vary linearly from a maximum value w_{max} to a minimum value w_{min}. For the acceleration coefficients, c_1 also varies nonlinearly from c_{1max} to c_{1min}, while c_2 is fixed at 0.1. Let $iter_{max}$ be the maximum number of iterations, k the current iteration and α and β constants; the term inertia and acceleration coefficients are given by equations 3, 4 and 5. These modifications were made in order to encourage the particles to seek solutions in the whole space instead of being trapped in local optima.

$$w(k) = w_{min} + \left(\frac{iter_{max} - k}{iter_{max}}\right)^{\alpha} (w_{max} - w_{min}) \tag{3}$$

$$c_1(k) = c_{1min} + \left(\frac{iter_{max} - k}{iter_{max}}\right)^{\beta} (c_{1max} - c_{1min}) \tag{4}$$

$$c_2(k) = 0.1 \tag{5}$$

With such modifications, the standard PSO algorithm is executed to find the best set of individual positions of the population $OP = (p^1, p^2, \ldots, p^p)$. Then, the SA algorithm initializes taking this set as a starting point, according to the following steps:

1. Initialize the sequence number of initial solutions: $i = 1$.
2. For the initial solution p^i, initialize the temperature controlling parameter: $T = T_0$.
3. Generate a new solution p^{temp} according to p^i.
4. Do the test and decide if we should accept the new solution. If $f(p^{temp}) < f(p^i)$, accept p^{temp} to replace p^i; Otherwise, calculate P from equations 1 and 2; generate a random number $rand$ between 0 and 1; if $rand < P$, accept p^{temp} to replace p^i. A uniform distribution was used in order to follow the original MPSO-SA.
5. Decrease T. If $T > T_{min}$ go back to 3. Else go to next step.
6. Choose the solution with the best objective function.

MPSO-SA was applied in the parameters optimization of an RBF network, in prediction of a variable that indicates the quality of polymers. The results confirmed the validity of the proposed model and the advantages of MPSO-SA against original PSO and SA algorithms.

III. Reservoir Computing Optimization

As previously noted, Reservoir Computing primary architectures (ESN and LSM) work with a recurrent neural network with fixed and randomly generated weights. However, Lukosevicius and Jaeger [7] say that it is unlikely that the random generation of weights and output layer training with a linear regression function is the optimal solution for computing RC. Then, it's necessary to seek alternatives for reservoir generations and readout training.

As an alternative to the reservoir layer generation, one can initialize the hidden layer weights in an unsupervised way or even from a supervised pre-training. The unsupervised adaptation involves optimizing some measure defined in the reservoir for a given input, not taking into consideration the desired output. In contrast, supervised pre-training also takes into account the desired output.

Besides reservoir adaptation, other ways to optimize Reservoir Computing can be considered. Ferreira and Ludermir [22] presented a method to optimize the global parameters' choice using genetic algorithms, in time series forecasting with practical application. The optimization took up 22.22% of the time required to perform an exhaustive search for the parameters and achieve similar results. In an exhaustive search, all parameters are combined without using any optimization method.

In 2011, Ferreira [9] developed a method to find the best reservoir in time series forecasting, called RCDESIGN. The method simultaneously searches the best values of the global parameters of the network topology and weights, combining an evolutionary algorithm with Reservoir Computing. Two other optimization methods were implemented to compare the results. RS Search tries to optimize the reservoir size, the spectral radius and the connection density. TR Search simultaneously searches the spectral radius and the network topology. Along with RCDESIGN, TR Search does not consider the approach of linear systems with RC. RCDESIGN showed satisfactory results in all databases studied, being better than the other methods used for comparison. All approaches were also applied to the prediction of wind speeds, which is an important task for

wind power generation.

PSO is an important candidate to perform Reservoir Computing optimization, due to its advantages over genetic algorithms [11] [12]. Also, extensions of this algorithm can increase efficiency, since such changes have been developed for this purpose. Sergio and Ludermir used PSO and two of its extensions (EPUS-PSO [23] and APSO) to optimize the overall parameters of RC. They concluded that, in the five series used, the proposed method reduced the number of training cycles needed to train the system [13]. The parameters used in the optimization were the number of nodes in the reservoir, the activation function of neurons in the reservoir, spectral radius of the weight matrix in the reservoir layer and the presence or absence of optional connections.

Based on RCDESIDGN, Sergio [14] proposed a method to optimize global parameters, topology and weights of Reservoir Computing, using PSO and two extensions, EPUS-PSO and APSO. The APSO algorithm achieved the best results according to the forecast errors in the time series used. EPUS-PSO obtained the best results when the criterion of training cycles required to reach the optimal values was taken into account.

Since APSO achieved greater improvement in the forecast errors, this paper proposes a hybridization between SA and APSO in the task of optimizing architecture, global parameters and weights of Reservoir Computing, for time series forecasting. A hybridization of these two algorithms showed satisfactory results when applied to a more traditional topology of neural networks, RBF [16].

IV. Reservoir Computing Optimization with APSO and SA

Hereafter, the proposed method will be presented. The representation of solutions will be described, followed by how the fitness function was calculated, algorithm optimization, the parameters involved and the numerical simulation. The optimization is based on RCDESIGN [9] and the method proposed by Sergio [13]. ESN was used as the Reservoir Computing architecture.

A. Solutions Representation

Each particle is represented by a vector s^i. Notation s_j^i denotes dimension j of the particle s^i. Next, each of these dimensions is described.

- s_1^i (η) – Nodes number in reservoir, integer between 50 and 200. This parameter defines the weight matrix size.
- s_2^i – Connection between input and output layers. Float between 0 and 1. If larger than 0.5, there is connection, else, there is not.
- s_3^i – Connection between bias and output layer. Float between 0 and 1. If larger than 0.5, there is connection, else, there is not.
- s_4^i – Feedback connection in output layer. Float between 0 and 1. If larger than 0.5, there is connection, else, there is not.
- s_5^i – Connection between bias and reservoir layer. Float between 0 and 1. If larger than 0.5, there is connection, else, there is not.
- s_6^i – Connection between output and reservoir layers. Float between 0 and 1. If larger than 0.5, there is connection, else, there is not.
- s_7^i – Neurons activation function. If 1, hyperbolic tangent, if 2, sigmoid.
- s_8^i – Readout training function. If 1, pseudo-inverse [7], if 2, ridge-regress [24].
- s_9^i – Leak rate. Float number between 0.1 and 1. Leak rate is a parameter that enables RC dynamic adaptation.
- s_{10}^i – Regularization parameter of the training functions. Float number between 10^{-8} and 10^{-1}. Regularization parameter is a noise that can be added to the reservoir responses.
- $s_{11}^i \dots s_{(\eta^2+3\eta+10)}^i$ – Reservoir weights W, input weights W^{in}, bias weights W^{bias} and feedback weights W^{back}. Float number between -0.1 and 0.1.

Vector size s^i is variable. This happens because the last positions depend on the nodes number in the reservoir layer, defined by dimension j = 1. Interval [50, 200] was set empirically, in order to create large reservoirs big enough to compute data and in which it is feasible to perform simulations. If η = 50, vector s^i has 2660 positions. If η = 200, vector s^i has 40610 positions.

B. Fitness Function

Fitness function is based on MSE (Mean Square Error) generated by the network. Due to the overfitting phenomenon, common in artificial neural networks, the fitness function is also based on the validation phase. Equation 6 shows the fitness function used.

$$f = \overline{MSE}_{Training} + \left\|\overline{MSE}_{Training} - \overline{MSE}_{Validation}\right\| \quad (6)$$

C. Optimization Algorithm

Reservoir Computing optimization with APSO and SA is presented in Algorithm 1. Network training was performed with k-fold cross-validation. Cross-validation with 10 partitions has proven to be a suitable value for most problems [25].

As seen above, the hybridization between APSO and Simulated Annealing is based on work by Li and Liu [16]. However, there are two differences in the process.

In Li and Liu's work, the authors propose a new way to calculate inertia term and acceleration coefficients. However, compared with standard PSO, APSO also presents a different way to calculate these variables. Since the approach used in APSO (ESE - Evolutionary State Estimation) to update these values is relatively more complex than that used in Li and Liu, ESE was selected. Another difference is how the final solution is selected. In MPSO-SA, SA receives as input the individual best solutions set, and, according to temperature, these solutions are improved. However, these solutions are not compared with the best global solution. This paper proposes to use the best global solution in this phase. When a new solution based on the best individual solutions is found, it is compared with the best global solution.

In MPSO-SA, Li and Liu do not say how a new solution must be proposed in the neighborhood, at the SA execution

phase. In this work, using the concept already used in APSO, the new solution is proposed according to the ELS (Elitist Learning Strategy) approach.

ELS randomly selects a particle dimension with better global performance and enforces a Gaussian perturbation, according to equation 7 (being *d* the *d-ith* dimension and *X* the minimum and maximum limits).

Algorithm 1: RC Optimization with APSO and SA

Input: database, swarm size (*s*), iterations number in APSO (*iterMax),* initial temperature (T_0), final temperature (T_f), temperature decreasing rate (*Tx)*
Select database
Randomly initialize swarm with size s
while (iteration ≤ *iterMax*) **do**
Create RC according particle position
while *fold* ≤ 10 (cross-validation) **do**
Create training sets (nine partitions) and validation set (one partition)
Simulate network with training set
Train readout
Calculate training set errors
end while
Calculate fitness function
Update swarm positions and velocities according APSO
end while
temperature = T_0
while (temperature ≤ T_f) **do**
Execute SA algorithm,
According to *Tx*, execute SA algorithm, being the best individual solutions set resulting from APSO the initial search space.
Update, when applicable, best global solution.
end while
Return best global solution
Create RC according to best solution
Calculate test set errors

$$P^d = P^d + \left(X^d_{max} - X^d_{min}\right) \times Gaussian(\mu, \sigma^2) \tag{7}$$

In Gaussian distribution with mean μ = 0, standard deviation (called elitist learning rate in APSO) is given by equation 8. g is the generation of the current iteration and *max_gen* denotes the maximum number of generations. σ is linearly decreased according to the number of generations.

$$\sigma = \sigma_{max} - (\sigma_{max} - \sigma_{min}) \frac{g}{\max\_gen} \tag{8}$$

D. Parameters

For APSO, the following parameter set was tested, based on [13]: swarm size = 50; number of iterations = 20; initial momentum term = 0.9. Regarding SA, the parameters were set after extensive initial testing, as follows: initial temperature = 100; final Temperature = 10, temperature decreasing rate = 0.85. The weights values of the reservoir was set in interval [-0.1 01]. These values were set empirically.

E. Experimental Method

Five benchmark time series were tested, including two variations:

- Narma order 10 and Narma order 30 (NAR10 and NAR30)

Narma is a discrete time series given by equation 9:

$$(y+1) = 0.3y(t) + 0.05y(t)\left[\sum_{i=0}^{k-1} y(t-i)\right] + 1.5u\big(t-(k-1)\big) * u(t) + 0.1 \tag{9}$$

The series input is a uniform random noise $u(t)$, t is time, k is the system order and y is the output. Two values for the system order were used: k = 10 and k = 30.

- Mackey-Glass average chaos and Mackey-Glass moderate chaos (MGS17 and MGS30)

Mackey-Glass, a continuous and unidimensional time series, is given by equation 10.

$$y(t+1) = \frac{0.2y(t-\tau)}{1 + y(t-\tau)^{10}} - 0.1y(t) \tag{10}$$

y(t) is the output in t time and τ is a delay parameter that leverages the chaos level. Two values for the τ parameter were used: τ = 17 (average chaos) and τ = 30 (moderate chaos). The first value is less chaotic than the second one.

- Multiple Sinewave Oscillator (MSO)

MSO series is used to create a system for generating multiple sines. It is given by equation 11:

$$y(t+1) = \sin(0.2 * t) + \sin(0.311 * t) \tag{11}$$

- Natural shining star series (STAR)

Available in [26], STAR series has 600 consecutive numerical observations of star light at midnight.

- Dow Jones Industrial Average (DJIA)

Available in [27], the Dow Jones Industrial Average financial series consists of daily observations of the index of the same name. Data used in this work contains 1444 records, with observations from January 2nd 1998 to August 26th 2003.

In order to compare the results with other methods in the literature, the performance of the proposed optimization in this work was calculated according to various forecast error indices. They are: Mean Square Error (MSE), Normalized Square Root Mean Square Error (NRMSE) and Normalized Mean Square Error (NMSE). These errors are given respectively by equations 12, 13 and 14.

$$MSE = \frac{1}{N * P} \sum_{i=1}^{P} \sum_{j=1}^{N} (T_{ij} - L_{ij})^2 \tag{12}$$

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

$$NMSE = \frac{1}{N * P} \sum_{i=1}^{P} \sum_{j=1}^{N} \frac{\left(T_{ij} - L_{ij}\right)^2}{var(t)} \tag{14}$$

In these equations, P is the patterns number in the data set,

N is the output units number and T_{ij} and L_{ij} are respectively the output and the desired values calculated by *i-th* neuron of the output layer. $var(t)$ is the variance of the values in the desired outputs set. Considering that MSE, NMSE and NRMSE raise the squared error, larger differences penalize the final assessment more sharply.

V. Numerical Simulations

Table I shows the NRMSE provided by the proposed algorithm in the training phase, along with the results reached by Sergio using APSO as optimization algorithm [13]. Table II shows the same information regarding the testing phase. Because of the random characteristics of the numerical simulations, and to perform hypothesis testing, the performance measures are represented by averages of 30 startups in each database. The values in parentheses are the standard deviations. The best performances among the investigated configurations are bold.

In Table I and II, the third column shows the comparison between the models according to the Student's t test at 5% of significance (95% of confidence). In this table, the "=" sign says that the null hypothesis was not rejected (the difference between the mean errors is not statistically significant) and the models have the same performance. The sign "<" says that the null hypothesis was rejected and that the algorithm provided by this work has the worst performance and the ">" sign says the opposite.

According to Table I, one can observe that, during the training phase, APSO-SA outperforms APSO in two databases, Narma 30 and Mackey-Glass 17. In remaining databases both algorithms reached the same performance. In the test phase, according to Table II, APSO-SA had better performance. This algorithm was not outperformed by APSO in any databases, and, in three of them (Narma 30, Mackey-Glass 17 and MSO), there were decreased forecasting errors. Regarding DJIA database, this series is more difficult than others and the results are not much different between distinct techniques. Furthermore, in most databases in which the differences between models were not statistically significant, APSO-SA reached the best absolute results. With more empirical studies about parameters set in APSO-SA, this algorithm can achieve even better results. This point of view is stronger with Narma 10 and Mackey-Glass 30, since APSO-SA outperforms APSO in NAR30 and MGS17.

In prior works, other algorithms were used to optimize Reservoir Computing in the same way as APSO-SA in this paper. Table III shows an absolute comparison between APSO-SA, PSO, EPUS-PSO and RCDESIGN (with Genetic Algorithms), in the test phase.

In absolute terms, since best results are bold, one can see that APSO-SA was outperformed in just one database. Reservoir Computing optimization with a hybrid algorithm between APSO and SA was better than the original PSO, two PSO extensions, and Genetic Algorithms. APSO-SA's better performance can be explained by the fact that this algorithm uses SA after APSO to find out the best solutions set. This way, the search is positioned in a good region, increasing the chances of finding global optimal solutions. The proposed algorithm works with the advantages of both PSO and Simulated Annealing. That is the main idea when an algorithm is hybridized with another one.

TABLE I
NRMSE IN TRAINING PHASE, 30 STARTUPS

Database	APSO-SA	APSO	Student's t Test
NAR10	**0.0436145368** (0.04361)	0.0477106861 (0.0222)	=
NAR30	**0.0429269667** (0.006823)	0.0910574919 (0.0195)	>
MGS17	**0.0000967796** (0.00001)	0.0001082409 (0.00001)	>
MGS30	**0.0003573238** (0.00001)	0.0003643903 (0.00002)	=
MSO	**0.0000000031** ($5*10^{-10}$)	0.0000000033 ($6*10^{-9}$)	=
STAR	**0.0307661037** (0.00473)	0.0318399628 (0.0027)	=
DJIA	**0.1318084901** (0.00013)	0.1318174990 (0.0001)	=

TABLE II
NRMSE IN TESTING PHASE, 30 STARTUPS

Database	APSO-SA	APSO	Student's t Test
NAR10	**0.0457983264** (0.00516)	0.0503082103 (0.0242)	=
NAR30	**0.0452129088** (0.00737)	0.0948630262 (0.0208)	>
MGS17	**0.0000986223** (0.00001)	0.0001099572 (0.00001)	>
MGS30	**0.0003707519** (0.00001)	0.0003778040 (0.00002)	=
MSO	**0.0000000032** ($5*10^{-10}$)	0.0000000034 ($7*10^{-10}$)	>
STAR	**0.0496100461** (0.00091)	0.0494933321 (0.0001)	=
DJIA	0.3787291947 (0.00084)	**0.3783935519** (0.0006)	=

TABLE III
NRMSE IN TESTING PHASE, 30 STARTUPS

Database	APSO-SA	PSO	EPUS-PSO	RCDESIGN
NAR10	**0.0457983264** (0.00516)	0,05030821 (0,0242)	0.05626217 (0,0301)	0.08462155 (0,0354)
NAR30	**0.0452129088** (0.00737)	0,09486302 (0,0208)	0.12576019 (0,0371)	0.20424126 (0,0469)
MGS17	**0.0000986223** (0.00001)	0,00010995 (0,00001)	0.00012705 (0,00001)	0.00050628 (0,0001)
MGS30	**0.0003707519** (0.00001)	0,00037780 (0,00002)	0.00039609 (0,00002)	0.00099067 (0,0002)
MSO	**0.0000000032** ($5*10^{-10}$)	$3.4*10^{-9}$ ($7*10^{-10}$)	$4.5*10^{-9}$ ($1*10^{-9}$)	0.00000120 (0,0000)
STAR	**0.0496100461** (0.00091)	0,04949333 (0,0001)	0.05068854 (0,0010)	0.08555901 (0,2088)
DJIA	0.3787291947 (0.00084)	0,37839355 (0,0006)	0.37877361 (0,0006)	**0.23938190** (0,0156)

Even with experiments with different settings, one can compare APSO-SA performance with other works that used

some databases studied, taking into account that the experiments were not reproduced. This comparison is performed in tables IV, V and VI. Since the standard deviation is not available in some of these works, this information was suppressed. Reservoir Computing optimization with APSO-SA outperformed all these works.

TABLE IV
MSE COMPARISON

Algorithm	NAR10	NAR30	MGS30
APSO-SA	**0.0000248907**	**0.0000254061**	**0.0003707519**
Sergio and Ludermir [13]	0.00023951	0.00013143	0.0000000092

TABLE V
NMSE COMPARISON

Algorithm	MGS30
APSO-SA	**0.0000001389**
Steil [28]	0.0340

TABLE VI
NRMSE COMPARISON

Algorithm	MGS17	MGS30	MSO
APSO-SA	**0.0000986223**	**0.0003707519**	**0.0000000032**
RS Search [9]	0.00584469	0.01114130	0.01347762
TS Search [9]	0.00168976	0.00260901	0.00068574
Jaeger [5][5]	0,00012	0,032	-
Wyffels et al. [29]	0,0065	0,0065	-
Jaeger and Haas [30]	0,000063	-	-
Schmidhuber et al. [31]	-	-	0,0103

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented a method to optimize the global parameters, the topology, and the weights of Reservoir Computing using a hybrid algorithm between APSO and Simulated Annealing. This hybridization was inspired by the MPSO-SA-RBF, proposed by Li and Liu [16].

Numerical simulations were performed and compared with the optimization done only with the APSO algorithm, according to [13]. Some benchmark time series were used as a database. According to the forecast errors, APSO-SA had better performance. This can be explained by the fact that this algorithm used SA after APSO to find out the best solutions set, positioning search in a better region.

The results were compared with other works in the literature. According to the forecast errors, the proposed optimization proved to be better than the others in all databases investigated. However, it is important to note that numerical simulations of the works used for comparison were not reproduced.

Reservoir Computing optimization with APSO-SA achieve good results, but its performance can be improved. This can be accomplished through a fine-tuning of the parameters used.

Other future works: use distinct algorithms to optimize Reservoir Computing; use the optimization proposed by this work in practical databases; test the proposed algorithm in benchmark optimization problems.

REFERENCES

[1] K. Vandoorne, M. Fiers, D. Verstraeten, B. Schrauwen, J. Dambre, P. Bienstman. "Photonic Reservoir Computing: A New Approach to Optical Information Processing". 12th International Conference on Transparent Optical Networks (ICTON), Munich, German, 2010.

[2] P. Buteneers, D. Verstraeten, P. van Mierlo, T.Wyckhuys, D.Stroobandt, R. Raedt, H. Hallez, B. Schrauwen. "Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing". Artificial Intelligence in Medicine, 53, pp. 215-223, 2001.

[3] A. Smerieri, F.Duport, Y.Paquot, M.Haelterman, B.Schrauwen, M.Massar. "Towards Fully Analog Hardware Reservoir Computing For Speech Recognition". International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), 1479, pp.1892-1895, 2012.

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

[5] H. Jaeger. "The echo state approach to analyzing and training recurrent neural networks". Tech. Rep. GMD 148, German National Resource Center for Information Technology, 2001.

[6] B. Schrauwen, J. Defour, D. Verstraeten, J. Van Campenhout. "The introduction of time-scales in reservoir computing, applied to isolated digits recognition". LNCS, 4668(1), pp. 471–479, 2007.

[7] M. Lukosevicius, H. Jaeger. "Reservoir computing approaches to recurrent neural network training". Computer Science Review, 3(3), pp. 127–149, 2009.

[8] Holland, H.J. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.

[9] A. Ferreira, T. B. Ludermir, R. Aquino. "An Approach to Reservoir Computing Design and Training. Expert Systems with Applications, v. 40, p. 181-195, 2013.

[10] J. Kennedy, R. Eberhart. "Particle swarm Intelligence". Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942-1948, 1995.

[11] R. Hassan, B. Cohanim, O. De Weck, G. Venter. "A Comparison Of Particle Swarm Optimization And The Genetic Algorithm". 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, pp. 1-13, 2005.

[12] S. Panda, N. P. Padhy. "Comparison of Particle Swarm Optimization and Genetic Algorithm for TCSC-based Controller Design". International Journal of Computer Science & Engineering, 1(1), pp. 41, 2007.

[13] A. Sergio, T. B. Ludermir. "PSO for reservoir computing optimization". ICANN 2012, Part I, LNCS 7552, pp. 685-692, 2012.

[14] A. Sergio. "Reservoir Computing Optimization with PSO - Otimização de Reservoir Computing com PSO". Master's Thesis. Centro de Informática, Universidade Federal de Pernambuco, 2013.

[15] Z-H. Zhan, J. Zhang, Y. Li, H. Chung, H. "Adaptive Particle Swarm Optimization". IEEE Transactions, Man and Cybernetics, 39, pp. 1362 -1381, 2009.

[16] J. Li, X. Liu. Melt index prediction by RBF neural network optimized with an MPSO-SA hybrid algorithm. Neurocomputing 74 (2011), pp. 735-740, 2010.

[17] S. Kirkpatrick, C. Gelatt, M. Vecchi. Optimization by Simulated Annealing. Science 220 (4598): 671–680, 1983.

[18] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45: 41–51.

[19] F. Van den Bergh. “An Analysis of Particle Swarm Optimizers”. PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science, 2001.

[20] M. Clerc, J. Kennedy. “The particle swarm - explosion, stability, and convergence in a multidimensional complex space”. IEEE Transactions on Evolutionary Computation, 6(1), pp. 58–73, 2002.

[21] I. C. Trelea. “The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection”. Information Processing Letters, 85, pp. 317–325, 2003.

[22] A. A. Ferreira, T. B. Ludermir. “Genetic algorithm for reservoir computing optimization”. International Joint Conference on Neural Networks, pp. 811-815, 2009.

[23] S. Hsieh, T. Sun, C. Liu, S. J. Tsai. “Efficient Population Utilization Strategy for Particle Swarm Optimizer”. IEEE Transactions, Man and Cybernetics, 30, pp. 444-456, 2009.

[24] C. M. Bishop. Pattern recognition and machine learning. In Information Science and Statistics. Springer, 2006.

[25] I. H Witten, E. Frank. Data Mining, Practical Machine Learning Toolsand Techniques with Java Implementations. Morgan Kaufmann Publishers, 2000.

[26] http://datamarket.com/data/list/?q=provider:tsdl. Last access in December 14th, 2013.

[27] http://www.djindexes.com/. Last access in December 14th, 2013.

[28] J. J. Steil. “Backpropagation–decorrelation: Online recurrent learning witho(n) complexity”. In International Joint Conference on Neural Networks, IJCNN 2004, 2, pp. 843–848, 2004.

[29] F. Wyffels, B. Schrauwen, D. Verstraeten, D. Stroobandt. “Band-pass reservoir computing”. In International Joint Conference on Neural Networks 2008, pp. 3203–3208, 2008.

[30] H. Jaeger, H. Haas. “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication”. Science, 308, pp. 78–80, 2004.

[31] J. Schmidhuber, D. Wierstra, M. E. Gagliolo, F. Gomez. “Training recurrent networks by evolino”. Neural Computation, 19(3), pp. 757–779, 2007.