Single-step-ahead and Multi-step-ahead Prediction with Evolutionary Artificial Neural Networks

V. Landassuri-Moreno[1], C. Bustillo-Hernández[2], J.J. Carbajal-Hernández[2]

[1]Centro Universitario UAEM Valle de México
Universidad Autónoma del Estado de México, C.P. 54500, México.
[2]Centro de Investigación en Computación
Instituto Politécnico Nacional, C.P 07738, México.
vmlandassurim@uaemex.mx, chbustillo004@cic.ipn.mx, juancarvajal@sagitario.cic.ipn.mx
Paper received on 25/07/12, Accepted on 11/09/12.

Abstract. In recent years the Evolutionary Algorithms have been remarkably useful to improve the robustness of Artificial Neural Networks. This study will introduce an experimental analysis of the FS-EPNet algorithm to understand how neural networks are evolved with a steady-state algorithm and compare the Single-step-ahead (SSP) and Multiple-step-ahead (MSP) methods for prediction tasks over two test sets. It was decided to test an inside-set during evolution and an outside-set after the whole evolutionary process has been completed to validate the generalization performance with the same method (SSP or MSP). Thus, the networks may not be correctly evaluated (misleading fitness) if the single SSP is used during evolution (inside-set) and then the MSP at the end of it (outside-set). The results show that the same prediction method should be used in both evaluation sets providing smaller errors on average over the generalization performance.

Keywords: Evolutionary Algorithms, Artificial Neural Networks, EANNs, Single-step-ahead prediction, Multi-step-ahead prediction

1 Introduction.

Artificial Neural Networks (ANNs) are mathematical models inspired by the structural and functional organization of biological neural networks. They are characterized by having input, hidden and output units with interconnection between them, where each connection has an associated weight which is updated during the training phase to allow the network to learn a given task. Since their origin, they have been used to solve control [1], classification [2, 3] and prediction [4] tasks, showing a performance and adaptability superior to those of conventional mathematical models, as it is actually known that one hidden layer with sufficient nodes can approximate any classification decision boundary [5]. Even though neural networks have proved to be a robust method for solving different kinds of problem, they involve several different parameters that need to be chosen appropriately to obtain a functional network. Early studies used to select many of those parameters

Advances in Intelligent and Information Technologies.
Research in Computing Science 60, 2012, pp. 3 -12.

by trial and error [5]. Another difficulty is that some of these parameters may change over time, and thus more elaborate methods are needed to adjust them.

ANNs and Evolutionary Algorithms (EAs) have been widely inspired by biological organisms, usually giving them superior performance when both are applied together to solve a problem than when they are applied in separate stages. Thus, Evolutionary Artificial Neural Networks (EANNs), sometimes called neuroevolution, have been remarkably useful at adapting the ANNs' parameters during evolution [2, 6, 7]. Furthermore, EAs can improve the robustness of networks because they are less likely to be trapped on local minima than traditional gradient-based search algorithms [6].

The usage of EAs over ANNs require an extra error-evaluation, e.g. Normalized Root Mean Squared Error (NRMSE) to determine the fitness of each individuals in the population (this metric is used in this work). Therefore, there may be different evaluation sets within an EANN: validation set to discover overtraining; inside-set to obtain the fitness of an individual during evolution, and a final test set called outside-set to evaluate the generalization performance after the evolution has finished. In this way, inside and outside terms are used to make reference to performance evaluation during and after the evolutionary process has been completed. Besides test sets, a prediction metric is needed, e.g. the Single-step-ahead (SSP) and Multiple-step-ahead prediction (MSP) methods. Note that in classification tasks, an extra test set (validation set) is not required like in the prediction case because the validation set uses the same method to evaluate the fitness during evolution as in the generalization test, i.e. like the SSP method. Consequently, the validation set and inside-set use to be the same in this case.

Thus, this paper is aimed to compare SSP and MSP procedures over both test sets (inside and outside) to forecast two chaotic time series (TS): *Lorenz* and *Mackey-Glass*, usually tested in prediction tasks. Thus, the networks may not be correctly evaluated (misleading fitness) if the single SSP is used during evolution (inside-set) and then the MSP at the end of it (outside-set), i.e. both evaluations may be perform in the same terms. Moreover, no previous study has been found explaining such scenario, which should be tested empirically.

The reminder of this paper is organized as follows: In Sec. 2 is presented an overview of the FS-EPNet algorithm aimed at evolving ANNs using a steady-state algorithm based on Evolutionary Programming. Sec. 3 then gives a description of SSP and MSP for TS forecasting. Thereafter, Sec. 4 presents the experimental set-up with the Lorenz and Mackey-Glass TS, while Sec. 5 describes our empirical prediction results for the comparison between SSP and MSP over the inside-set and outside-set. Finally, we provide our conclusions in Sec. 6.

2 The FS-EPNet Algorithm

The EPNet algorithm [3, 8] is based upon the standard Evolutionary Programming (EP) approach, aimed at evolving ANN architectures and weights at the same time as obtaining smaller network topologies. The original algorithm [3] does not tackle the feature evolution, i.e. input adaptation in the same evolutionary process. However, further improvements consider their evolution [8], i.e. Feature Selection

EPNet algorithm (FS-EPNet), being the algorithm used du-ring this empirical study. The FS-EPNet algorithm emphasizes the evolution of ANN behaviours by EP, like node-splitting, which maintains the behavioural (i.e. functional) link between the parent and its offspring. It does not have a crossover operator, nor a genotype to represent the individuals. Instead it carries out the evolutionary process by performing only nine different mutation operations directly on the phenotype as shown in Fig. 1: (1) hybrid training composed of training with the Modified Back Propagation (MBP) algorithm and Simulated Annealing (SA); (2) node deletion; (3) connection deletion; (4) input deletion; (5) delay deletion; (6) connection addition; (7) node addition; (8) input addition; and (9) delay addition. The algorithm performs only one such mutation on the selected individual in each generation. The training in the EPNet algorithm is only a partial training, i.e. the networks are not trained until they converge. This is motivated by computationally efficiency, which lets the evolution advance faster, with the individuals improving their fitness through the generations. For a more detailed description of the EPNet algorithm see [3, 8].

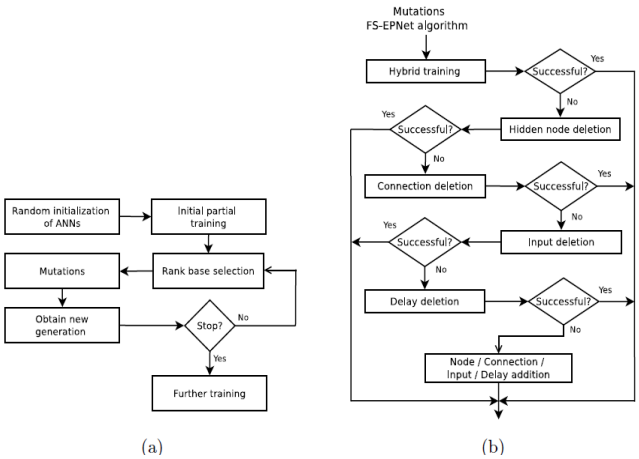

Figure 1. Feature Selection EPNet algorithm (FS-EPNet). General procedure (Fig.1a) and FS-EPNet mutations (Fig. 1b)

Table 1. Single-step prediction

Forecasting	*Inputs*
y_{t+1}	x_t, x_{t-1}, x_{t-2}
y_{t+2}	x_{t+1}, x_t, x_{t-1}
y_{t+3}	x_{t+2}, x_{t+1}, x_t
y_{t+4}	$x_{t+3}, x_{t+2}, x_{t+1}$

3 Time Series Prediction

For the TS prediction problem with ANNs, it is common to try to use a small subset of recent TS information to perform the prediction. Therefore, we are aiming

to obtain accurate predictions using only a finite segment of previous values up to the point to be predicted. Thus we have:

$$x_{t+1} = F[x_t, x_{t-k}, x_{t-2k}, \ldots, x_{t-(d-1)k}] \quad (1)$$

Where d is the number of inputs, k is the time delay and F is the method or algorithm that performs the prediction (the network for this work). There is one condition that needs to be satisfied: given an attractor of dimension D, we must have $d \geq 2D+1$ [9]. There are two general ways to perform the prediction of a TS in terms of the desired number of values to be forecast. Thus, assume the TS X is $[x_1, x_2, ..., x_t]$, the number of points ahead to predict is n, the test set is $[x_{t+1}, x_{t+2}, ..., x_{t+n}]$, and the forecast in the same interval is $[y_{t+1}, y_{t+2}, ..., y_{t+n}]$. In the following examples, we are assuming that the number of inputs (past information) is 3, delays are set at 1 and the prediction step is $\Delta t=1$.

3.1 Single-step-ahead prediction (SSP)

The simplest method is just to predict a value in the future, and we may call this method One-step or Open-loop or Single-step-ahead prediction (SSP). It is called Open-loop forecasting because a pattern is used to predict a value and no feedback is used to continue the predictions as in an autoregressive method. Table 1 shows the single-step prediction method. A sample of previous works that have used (SSP) are [10, 11, 12, 13], where [10, 11] predict the *Lorenz* TS and [12, 13] the *Mackey-Glass* TS.

3.2 Multi-step-ahead prediction

Another interesting prediction method is the Multi-step-ahead prediction (MSP) which uses closed-loop forecasting through an autoregressive method as shown in Table 2.

Table 2. Multiple-step prediction

Forecasting	*Inputs*
y_{t+1}	x_t, x_{t-1}, x_{t-2}
y_{t+2}	y_{t+1}, x_t, x_{t-1}
y_{t+3}	y_{t+2}, y_{t+1}, x_t
y_{t+4}	$y_{t+3}, y_{t+2}, y_{t+1}$

Note that in Table 2 the predictions are used as input values in subsequent predictions, i.e. it is repeated one-step prediction several times, using the actual prediction to predict the next value. The input vector from the SSP (Table 1) and MSP (Table 2) methods may be seen as a window of d values with k delays that is moved one position ahead every time a value is predicted, to be ready to predict the next value. The real difference between both methods is that the SSP moves the window

input vector over the original data available, meanwhile the MSP starts with the original data, overlap original and predicted data, and finish with predicted values in the window input vector. Previous publications that used the MSP method are [3, 14, 15], where [14, 15] are focused on the Lorenz TS and [3] predicting the Mackey-Glass TS.

3.3 MSP and SSP comparison

For the previous section, it can be said that prediction tasks with SSP are similar to classification tasks as one input vector produce one output vector and there is no feedback in the output as in MSP. Having said that, a standard procedure in the literature is to evaluate an inside-set with the SSP method for classification and prediction tasks to obtain the fitness of individuals. Moreover, any publication has been found so far that uses MSP over the inside-set, that kind of evaluation is never said and it may be assumed to be SSP as it is the standard.

4 Experimental set-up and data sets

As previously remarked, it was decided to use an extra test set during evolution as tasks solved with MSP require to validate the generalization performance with the same method (MSP) when the evolution finish, i.e. validation, inside-set and outside-set. Different to classification task where only the validation set and outside-set is required.

Thus, the networks may not be correctly evaluated (misleading fitness) if the single SSP is used during evolution and then the MSP at the end of it. For example, it can be assume that SSP method is easier than MSP by the feedback in the later, therefore, if a prediction task requiring MSP is evaluated with SSP during evolution (inside-set), it will probably produce a different fitness than if the MSP is used in the same inside-set, which could produce a bias in the selection process with networks not so fit. For that reason it was needed to use an extra test set in prediction tasks, so the validation set is used mainly to evolve the learning rate and the inside-set to measure the fitness as it were a real prediction. However, it may be expected to obtain a smaller fitness error during evolution with SSP than MSP.

There are some common parameters that were fixed for the experiments throughout this study: population size 30, generations of evolution 3000, initial connection density 30%, initial learning rate 0.15, minimum learning rate 0.01, epochs for learning rate adaptation 5, number of mutated hidden nodes 1, number of mutated connections 1-3, temperatures in SA 5, iterations per temperature in SA 100, 1500 epochs of training inside the FS-EPNet, and 2000 of further training at the end of the algorithm. The only stopping criteria was the number of generations. For all the experiments, 30 independent runs were performed to ensure statistical validity of the results. The inside-set was set-up with 30 patterns to perform the prediction and the MSP is performed on the outside-set in all TS tested. All these parameters were set at convenient traditional values and are not intended to be optimal.

Two chaotic TS were used to test the insights presented in this study: a) the first one is the *Lorenz* TS [16] generated with the fourth order Runge-Kutta method as done in [14¡**Error! No se encuentra el origen de la referencia.**], i.e. the following values are used to generate the TS: Δt=1, σ=10, *r*=28, β=8/3 and *time step*=0.05 using 1000 values to train and 100 to test; and b) the *Mackey-Glass* TS usually generated with fourth-order Runge-Kutta method as Lorenz TS, where the parameters used here to generate it are: *x*(0)=1.2, τ=17, β=0.2 and β=-0.1 as done in [3] using 500 values to train and 500 to test. Therefore, the outside-set for *Lorenz* TS was set to 100 and for *Mackey-Glass* to 500 patterns.

5 Experimental Results

In this section is presented the results from a set of experiments developed to determine if the usage of SSP in the inside-set may degrade the performance of task requiring MSP on the outside-set.

To illustrate this, consider Fig. 2 for the best predictions found for the Lorenz with MSP (Figs. 2a and 2b) and SSP (Figs. 2c and 2d) on the inside-set during evolution and using MSP on the outside-set. Interestingly to note (and as previously expected), the average fitness of the networks evaluated with SSP on the inside-set have a lower error during all the evolutionary process than the fitness obtained from the MSP as can be seen in Fig. 3. The best prediction error on the inside-set at the end of the 3000 generations of evolution were smaller with the SSP than with the MSP as expected too (Table 3, NRMSE Inside-set row). It was also obtained a smaller error with the SSP on the average fitness over all independent trials as shown at the end of generations in Fig. 3.

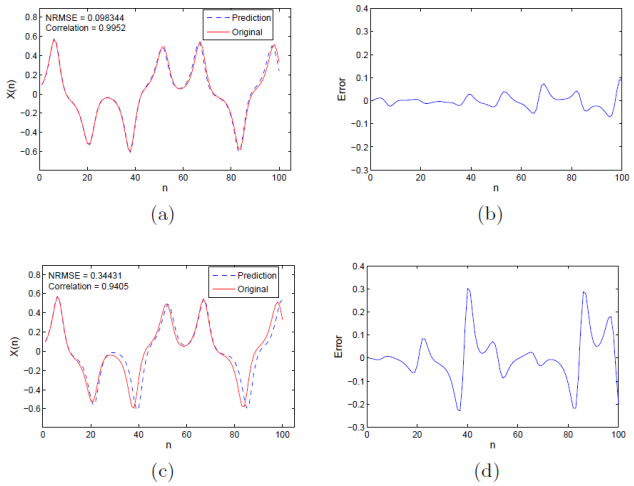

Figure 2. Best predictions found for Lorenz TS with MSP (Figs. 2a and 2b) and SSP (Figs. 2c and 2d) on the inside-set during evolution and using MSP on the outside-set after 3000 generations. Figs. 2b and 2d present the error in terms of $y_i(t) - z_i(t)$, where $y_i(t)$ is the prediction

at time t and $z_i(t)$ is the original data to be predicted (outside-set).

At the end of the evolution, even the network that uses MSP have a bigger fitness error in the inside-set, it obtained the smallest generalization error, with statistical significant having a *p-value <0.01* (Fig. 2a), because the selection mechanism during evolution was in the same terms as the generalization measurement.

Table 3 presents the individual parameters evolved for the Lorenz TS, showing how the NRMSE over the inside-set is smaller when the MSP is used during evolution than SSP, but the generalization performance is smaller when the MSP is used in both test sets. Also note that using SSP produce the convergence of delays (Table 3 and graphically in Fig. 4d) as it is easier to predict with SSP than MSP for the feedback in the latter as previously remarked. Fig. 4 shows the evolution of hidden nodes (Fig.4a), connections (Fig. 4b), input nodes (Fig. 4c) and delays (Fig. 4d) for the Lorenz TS. There can be seen that evolving those parameter were similar in both cases for hidden, input nodes and connections, i.e. using the SSP or MSP on the inside-set. It may be worth to say too, that an incorrect prediction method over the inside-set may produce poorer networks by the deviation of it parameters as can be seen in Fig. 4d.

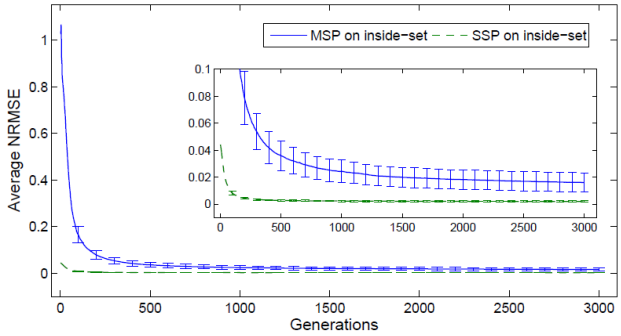

Figure 3. Average fitness value of *Lorenz* TS with MSP and SSP over the inside-set

Table 3. Average fitness value of *Lorenz* TS with MSP and SSP over the inside-set

Parameter	*MSP − MSP*				*SSP − MSP*			
	Mean	*Std Dev*	*Min*	*Max*	*Mean*	*Std Dev*	*Min*	*Max*
Number of Inputs	6.70000	1.91455	3	10	6.13333	1.79526	3	9
Number of Delays	2.46666	0.93710	1	4	1	0	1	1
Number of Hidden Nodes	13.3333	4.25346	5	26	12.8	3.28423	6	18
Number of Connections	108.466	56.1989	39	351	92.9333	35.5614	43	172
NRMSE Validation Error	0.02475	0.02848	0.00062	0.12563	0.00145	0.00104	0.00035	0.00447
NRMSE Inside-set	0.01657	0.00775	0.00676	0.03480	0.00189	0.00084	0.00123	0.00477
NRMSE Outside-set	0.52301	0.26162	0.09834	0.92639	0.73256	0.23340	0.34431	1.20320

Comparing these results against results found in the literature, Dudul [14] obtain a NRMSE for the best individual with a State-space 8th order of NRMSE = 0.1822 and a NRMSE = 0.7325 with a Regularized ANN-ARMA, while the FS-EPNet obtain a NRMSE = 0.09834 (Table 3) for the best individual found.

Table 4. *Mackey-Glass* time series individual results

Parameter	*MSP − MSP*				*SSP − MSP*			
	Mean	*Std Dev*	*Min*	*Max*	*Mean*	*Std Dev*	*Min*	*Max*
Number of Inputs	8.33333	2.39732	5	13	8.10000	1.26899	5	9
Number of Delays	3.56667	0.85836	2	6	3.16667	0.74664	2	4
Number of Hidden Nodes	16.0666	6.17522	6	31	12.5666	3.94517	7	23
Number of Connections	150.600	78.1499	52	376	114.866	51.2214	55	263
NRMSE Validation Error	0.00567	0.00835	0.00049	0.04147	0.00182	0.00112	0.00061	0.00466
NRMSE Inside-set	0.00330	0.00152	0.00119	0.00706	0.00377	0.00191	0.00163	0.01075
NRMSE Outside-set	0.16810	0.04626	0.07397	0.30965	0.17183	0.31555	0.01696	1.41379

To finalize the results of this work, in Table 4 is presented the individual results of evolving the Mackey-Glass TS. It can be seen how the usage of MSP over the inside-set produced smaller average errors than SSP, however in this case there where no statistically significance in the results. Comparing these results against previous studies, e.g. [17], it was found that the FS-EPNet could not improve them, requiring a more analysis and further research.

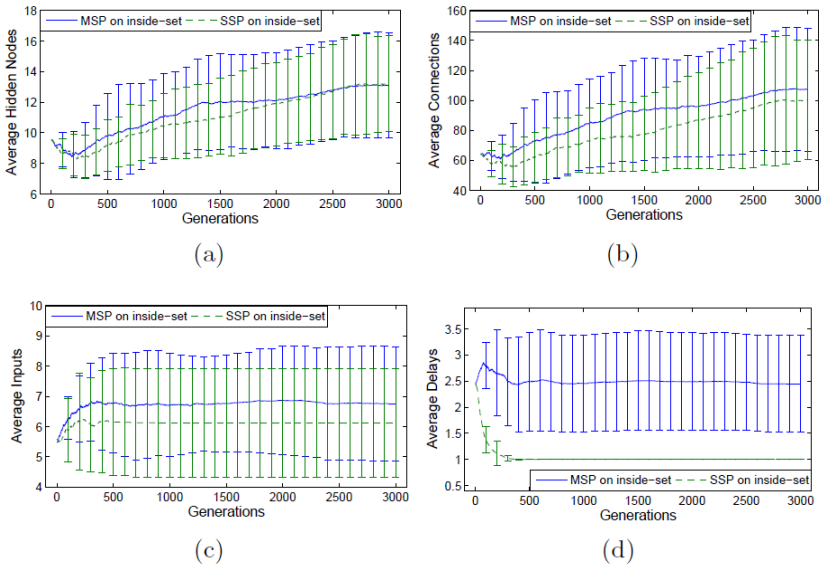

Figure 3. Best predictions found for the *Lorenz* TS with MSP and SSP over the inside-set for hidden (Fig. 4a, connections 4b), input (Figs. 4c and delays 4d) during 3000 generations

6 Conclusions and Discussion

In this work was presented a comparison between two prediction methods, Single-step-ahead (SSP) and Multi-step-ahead, during the evolution of Artificial Neural Networks (ANNs) for time series (TS) prediction. The experiments were carried out using the EPNet algorithm designed to evolve ANNs architectures and weights simultaneously through and steady-state procedure. From two chaotic TS tested (Lorenz and Mackey-Glass), it was determined that tasks that use SSP will use SSP for the fitness during evolution and to evaluate the generalization performance. Contrary, tasks that use MSP will use the same MSP method in both parts of the process. Further research is required to test a broad range of TS to generalize these results, however, from this study it can be expected that the same method should be used in both stages.

References

1. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2) (2002) 99–127
2. Bullinaria, J.A.: Understanding the emergence of modularity in neural systems. Cognitive Science 31(4) (2007) 673–695
3. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks 8(3) (1997) 694–713
4. Cholewo, T.J., Zurada, J.M.: Sequential network construction for time series prediction. International Conference on Neural Networks 4 (Jun 1997) 2034–2038
5. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (November 1995)
6. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9) (1999) 1423–1447
7. Bullinaria, J.A.: Evolving neural networks: Is it really worth the effort? In: Proceedings of the European Symposium on Artificial Neural Networks, Evere, Belgium: d-side (2005) 267–272
8. Landassuri-Moreno, V., Bullinaria, J.A.: Feature selection in evolved artificial neural networks using the evolutionary algorithm EPNet. In: Proceedings of the 2009 UK Workshop on Computational Intelligence. UKCI '2009, Nottingham, UK: University of Nottingham. (July 2009)
9. Belaire-Franch, J., Contreras, D.: Recurrence plots in nonlinear time series analysis: Free software. Journal of Statistical Software 7(9) (2002)
10. Rojas, I., Pomares, H., Bernier, J.L., Ortega, J., Pino, B., Pelayo, F.J., Prieto, A.: Time series analysis using normalized pg-rbf network with regression weights. Neurocomputing 42(1-4) (2002) 267–285
11. Gholipour, A., Araabi, B.N., Lucas, C.: Predicting chaotic time series using neural and neurofuzzy models: A comparative study. Neural Processing Letters 24(3) (2006) 217–239
12. Müller, K.R., Smola, A.J., Rätsch, G., Schökopf, B., Kohlmorgen, J., Vapnik, V.: Using support vector machines for time series prediction. (1999) 243–253
13. Müller, K.R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks, London, UK, Springer-Verlag (1997) 999–1004

14. Dudul, S.V.: Prediction of a lorenz chaotic attractor using two-layer perceptron neural network. Applied Soft Computing 5(4) (2005) 333–355
15. Guerra, F.A., dos S. Coelho, L.: Multi-step ahead nonlinear identification of lorenz's chaotic system using radial basis neural network with learning by clustering and particle swarm optimization. Chaos, Solitons & Fractals 35(5) (2008) 967 – 979
16. Lorenz, E.N.: Deterministic nonperiodic flow. Journal of atmospheric Science 20 (1963) 130–141
17. Assaad, M., Boné, R., Cardot, H.: A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Information Fusion 9(1) (2008) 41–55 Special Issue on Applications of Ensemble Methods.2. De Mol, L.: Tracing Unsolvability: A Mathematical, Historical and Philosophical analysis with a special focus on Tag Systems. PhD thesis, University Gent (2007)