Short-term load Forecast Based on Neuro-evolution Algorithm

A.TIGUERCHA A.A.LADJICI M.BOUDOUR
University of Sciences and Technology Houari-Boumediene, Algiers, Algeria
Laboratory of Industrial and Electrical Systems (LSEI), atiguercha@usthb.dz,

Abstract: *Load forecasting is an important tool in power system planning, operation and control. Load forecasting ensures the equilibrium between consumption and production and, so, helps in maintaining system stability, and optimal operation of the electricity market. Neuroevolution leverage the strengths of two biologically inspired areas of machine learning: artificial neural net works and evolutionary algorithms. The basic idea of Neuro-evolution algorithm is to search the space of neural network policies directly using an evolutionary algorithm, and find the best structure possible for the task at hand. Neuro-evolution can, therefore, improve the effectiveness of Neural Network by optimizing its structure in terms of complexity and efficiency using the optimization capabilities of evolutionary algorithms. The current paper presents a short-term load forecasting methodology, based on neuro-evolution algorithm. A comparative study is conducted between NE and two of the most used machine learning algorithms, artificial neural network (ANN), and Support Vector Regression (SVR).*

Key words: *Short term load forecasting, Neuro-evolution Algorithm, Forecasting Methods*

1. Introduction.

As a matter of fact, the electricity market has been privatized and restructured in many countries around the world. The main reason for such a change lies in the expectation that competition could lead to a reduction in electricity prices and could stimulate the emergence of new technologies. However, the prices consider ably higher than marginal prices have been observed because of the emergence of strategic behavior and the volatility of load in market where the load is stochastic and not known in advance [1,2].

Electricity markets are becoming more sophisticated and load forecasting is gaining importance for market participants to adjust their bids in the day-ahead Electricity market. The knowledge of next day load, is very important to a producer (or a consumer) in a competitive market. Knowing the next day load, leads to a better price forecasting and better generation and consumption scheduling [3,4].Load forecast is made by extrapolating the past load data while taking into account the effect of weather (temperature and humidity) and time events (workdays, holly-days and special events). The relationship between load and these factors is complex, nonlinear [5] and needs specialized tools.

Several interpolation and regression techniques have been proposed and applied to load and price forecasting problems. Those techniques include regression models (ARIMA, SARIMA and GARCH), statistical models and supervised learning algorithms (SVM and artificial neural networks).

Authors In [6], develop an ARIMA and transfer function models applied to the short-term load forecasting by considering weather-load relationship in Taiwan power system.

Authors in [7], performs a comparison study between the results of the three methods: ANN, NFIS and a new stochastic model (called REGARIMA). In [8] the authors made a comparative analysis technique between a Support Vector Machines method and hybrid system that combines the low level of computational neural networks with the high level in the reasoning ability of fuzzy systems.[5] develop a technique based on Neural Network and Rough Set to solve very short-term load forecasting problem, a support vector regression is used to made a comparative study on load forecasting technologies for different geographical distributed loads and in order to reduce the error of load forecasting [9]used an hybrid method based on Fuzzy Logic method and Artificial Neural Network.

In [10] authors develop genetic algorithm (GA) based support vector machine (SVM) forecasting model with deterministic annealing (DA) clustering, SVM parameters are optimized through genetic algorithms, which were used in SVM model. The current paper aims to develop a technique applied to Short Term load forecasting (STLF) using a neuro-evolution (NE) approache. The NE used in this work, is a neural network trained by a coevolutionary algorithm in order to find the best topology of the neural network. As a validation, the obtained results are compared with those of an SVR and ANN using the MATLAB NFTOOL Toolbox.

2. Load forecasting methods.

Load forecasting problem can be divided into three categories: Short-term forecast: this is usually from one hour to a week, medium forecast which is from a week to a year and Long-term forecast longer than a year.

The current paper focuses on short-term load forecasting that gives the load forecast for one day ahead to one week ahead. Such forecast gives valuable information to the System Operator (ISO) and helps to maintaining stability and controlling market, thus leading to better system reliability.

In this work, three load forecasting methods are developed:

• Conventional Artificial Neuron Network (ANN)
• Suport vector Regression (SVR)
• Neuroevolution Algorithm (NE)

2.1 Artificial Neural Networks.

Neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use [11]. Neural network offers the potential to overcome the reliance on a functional form of a forecasting model. The main advantage here is that most of the forecasting methods seen in the literature do not require a load model. However, training usually takes a lot of time. ANNs have been integrated with several other techniques to improve their accuracy. [12].

Neuron Network mimics the brain in two main aspects:

- Knowledge is acquired by the network through learning process.
- Inter neuron connection strengths known as synaptic weights are used to store the knowledge.

The figure 1 presents a typical multi-layer neural network work-flow. An elementary neuron with *P* inputs is shown below. Each input is weighted with an appropriate *w*. The sum of the weighted inputs and the bias forms the input to the transfer function f of the hidden layer. Neurons can use any differentiable transfer function *f* to generate their output. The Neural network Outputs are weighted sum of the outputs of the hidden layer neurons.

$$A_j(\bar{x},\bar{\omega}) = \sum_{i=0}^{n} f(x_i\omega_{ij}) + b_i \qquad (1)$$

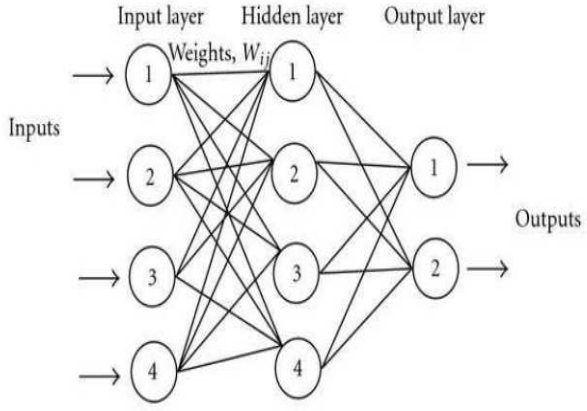

Fig.1 Neural network architecture

To be applied to a specified task, neural networks are trained, so that a particular input leads to a specific target output. The network is trained based on a comparison of the outputs and the targets, until the network output matches the target. Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems.

The neural network is provided with a correct answer (output) for every input. Weights are determined to allow the network to produce answers as close as possible to the target. The error depends on the weights, and we need to adjust the weights in order to minimize the error which is given by:

$$E_j(\bar{x},\bar{\omega},d) = (O_j(\bar{x},\bar{\omega}) - d_j)^2 \qquad (2)$$

With: O_j is the Activation function of neuron j After that we use a gradient descent method to adjust the weight.

2.2 Support Vector Machines.

Support Vector Machines (SVM) are a learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory [13]. This learning methodology introduced by Vapnik has been proven to be very powerful and had outperformed most other machine learning paradigms in a variety of applications [14], and [17].

SVMs were originally designed for classifications problems; they can also be applied to regression problems by the introduction of the concept of loss functions [15, 16]. In Support Vector Regression (SVR), we have to define a function *f(x)* that has at most $\mathcal{E}$ deviation from the actually obtained target y_i for all the training data and in the same time as flat as possible. Flatness in this case means to reduce the model complexity by minimizing $\| (w) \|^2$ so we can write:

$$\min \Phi(w) = \frac{1}{2}\|(w)\|^2 \qquad (3)$$

With the constraints:

$$\begin{aligned} y_i - w^t\varphi(x_i) - b \le \varepsilon \\ w^t\varphi(x_i) - b \le \varepsilon \end{aligned} \qquad (4)$$

This means, we do not care about errors as long as they are less than $\mathcal{E}$, but will not accept any deviation larger than this. To be more realistic, one can add slack variables $\xi_i, \xi_i^*, i = 1,\ldots,N$, to cope with otherwise infeasible constraints of the optimization problem (3); hence we arrive at the formulation stated below:

$$\min \Phi(w,\xi) = \frac{1}{2}\|(w)\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \qquad (5)$$

Subject to:

$$\begin{aligned} y_i - w^t\varphi(x_i) - b &\le \varepsilon - \xi_i \\ w^t\varphi(x_i) + b - y_i &\le \varepsilon + \xi_i^* \\ \xi_i^*, \xi_i &\ge 0 \end{aligned} \tag{6}$$

Where, C is a positive constant as regularization parameter. The optimization formulation can be transformed into a dual problem:

$$\begin{aligned} \min \Phi() &= \frac{1}{2}(\alpha_i - \alpha_i^*)\varphi(x_i)^T \varphi(x_j)(\alpha_i - \alpha_i^*) \\ &-\sum_{i=1}^{N} \alpha_i(y_i + \varepsilon) + \sum_{i=1}^{N} \alpha_i^*(y_i + \varepsilon) \end{aligned} \tag{7}$$

By introducing the kernel trick we can write:

$$\begin{aligned} \min \Phi() &= \frac{1}{2}(\alpha_i - \alpha_i^*)K(x_i, x_j)(\alpha_i - \alpha_i^*) \\ &-\sum_{i=1}^{N} (\alpha_i - \alpha_i^*)y_i + \sum_{i=1}^{N} (\alpha_i + \alpha_i^*)\varepsilon \end{aligned} \tag{8}$$

With constraints:

$$\begin{aligned} 0 \le \alpha_i, \alpha_i^* \le C, i = 1, ..., N \\ \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0 \end{aligned} \tag{9}$$

Solving the problem with corresponding constraints determines Lagrange multipliers, and the regression function is given by:

$$\begin{aligned} f(x) &= \sum_{0 \le \alpha_i, \alpha_i^* \le C} (\bar{\alpha}_i - \bar{\alpha}_i^*)K(x_i, \alpha) + b \\ &\textit{where} \\ \bar{b} &= -\frac{1}{2}\sum_{i=1}^{N}((K(x_i, x_r)K(x_i, x_s)) \end{aligned} \tag{10}$$

The vector alpha is called support vector and is defined by solving the problem stated in (8) and (9) using a quadratic programming optimization, *b* is the bias and *y = f(x)* are the output. SVM are trained in batch mode: in the first phase: the user have to define a set of inputs x and outputs y and solve (8) and (9) to find the support vector. In a second phase, the user provides a new set of inputs and the outputs are calculated using (10).

2.3 Neuro-evolution.

The basic idea of neuro-evolution algorithm is to search the space of neural network policies directly by using an evolutionary algorithm. Therefore, a NE combines the learning capability of a neuron network with the global optimization capabilities of evolutionary algorithms. Figure 2 shows a typical NE Algorithm based on ANN and Evolutionary strategies algorithm.

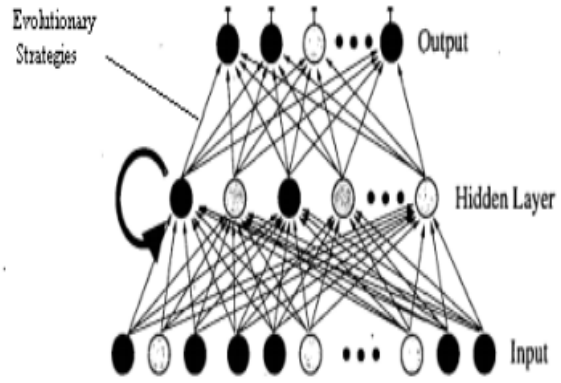

Fig.2 Neuro-Evolution Algorithm.

There are two mostly known architectures of NE, Symbiotic adaptive neuro-evolution (SANE) [18] and neuro-evolution of Augmented Topologies (NEAT).

In SANE, neurons compete on the basis of how well, on average, the networks in which they participate perform. A high average fitness means that the neuron contributes to forming successful networks and, consequently, suggests that it cooperates well with other neurons. Over time, neurons will evolve that result in good networks. The system breaks the problem down to that of finding the solution to smaller, interacting sub-problems.

The NEAT method is classified as a Topology and Weight Evolving Artificial Neural Networks (TWEANN), which are used in combination with evolutionary algorithms and neural networks in order to evolve weights and topologies [19-22].

3. Cooperative co-evolution approach neuro-evolution.

This section provides detailed implementation of the proposed neuro-evolution algorithm, where a NN is trained by a cooperative co-evolutionary algorithm (CCEA) in order to perform a good forecasting performance.

3.1 Cooperative co-evolutionary algorithm.

Cooperative co-evolutionary algorithms (CCEA) are evolutionary algorithm, where, instead of evolving a single population, several populations co-evolve simultaneously [23]. The most common definition of Co-evolutionary Algorithms, in the community of Evolutionary Computation, refers to an algorithm in which two populations or more are evolved using an Evolutionary Algorithm and in which individual fitness

depends on interaction with individuals of other populations.

In CCEA the optimization problem is explicitly decomposed into simpler sub-problems, and assigns each sub-problem to a population. Save for evaluation, each population evolve independently of one another. Therefore, an individual of a particular population represents only a component of a potential solution; collaborators are selected, randomly, from the other population to represent the remaining components of a solution [24].

In the same way as in traditional evolutionary algorithms, individuals of each population have to be reproduced and evaluated and the fittest individuals are selected to be part of the next generation. The main difference is that in CCEA, individuals of one population have to be evaluated against his collaborators.

Special care must be taken when applying CCEA to train a neuron network:

- **Representation:** the functionality of each population is different from another. The first population acts as input layer, the second as hidden layer and the third as an output layer:
  - In the input layer: to inputs coded as a two columns real valued vector.
  - The hidden layer a one column real valued vector.
  - The output layer a one column real valued vector.
- **Reproduction:** to each individual a crossover and mutation are applied
- **Evaluation:** each individual is evaluated against a set of evaluators according to equation (11) or (12)
- **Selection:** the most fitted individuals of each population are selected to be part of the next generation.

The main difference between conventional EA and CCEA resides on the process of evaluation, depending on the set of evaluators the objective function may be evaluated many times [24]. The evaluation of an individual is taken as the average value of the whole of the interactions of this individual with the whole of his evaluators.

$$Fit\left(a^K\right)=\frac{1}{q}\sum_{i=1}^{q}F\left(P_i\left(a_i^1\right),...P_K\left(a^K\right),...,P_N\left(a_i^N\right)\right) \quad (11)$$

The simplest manner to carry out the evaluation is to choose the *'best'* individual of each population:

$$Fit\left(a^K\right)=\frac{1}{q}\sum_{i=1}^{q}F\left(P_i\left(a_{best}^1\right),...P_K\left(a^K\right),...,P_N\left(a_{best}^N\right)\right) \quad (12)$$

A typical cooperative co-evolution algorithm can be represented as follow:

Algorithm 1: Co-evolutionary Cooperative algorithms

foreach *population* $P_i \in P$ **do**

  $Pi(0) = random;$

  μ_i: individuals in each population ;

  λ_i: number of offspring ;

  θ_r : reproduction parameters (mutation and crossover) ;

  θ_s : selection parameters ;

end

while *termination creteria* **do**

  foreach *population* $P_i \in P$ **do**

    Select evaluators S from P ;

    $F_i(t) = evaluation(P_i(t), S)$;

    $P_i'(t) = reproduction(P_i(t), \theta_r, \theta_s, \lambda)$;

    $F_i'(t) = evaluation(P_i'(t), S)$;

    $P_i(t+1) = selection(P_i(t), F_i(t), P_i'(t), F_i'(t), \mu, \theta_s)$;

  end

end

4. Experimental studies.

To test the effectiveness of the proposed method in this work we used the hourly load data weather conditions collected in 2002 from New England power system archive, Figure 3, 4 and 5 respectively show the annual curve Load in MW, weekly curve load and three similar days of the same month load.

Short term load forecasting mainly depends on the following conditions:

- Days' load
- Day temperature.

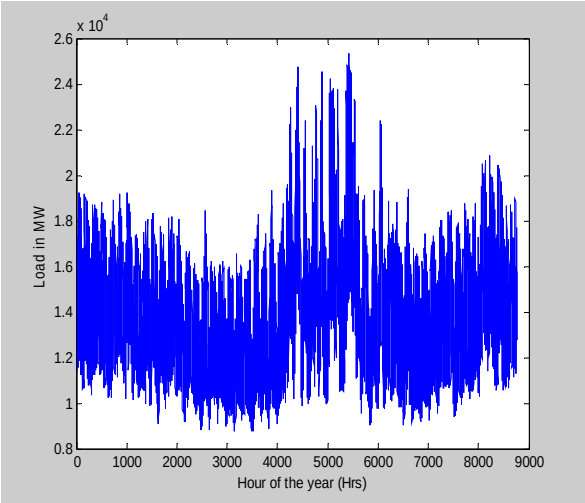

Fig.3 Annual Loads Curve

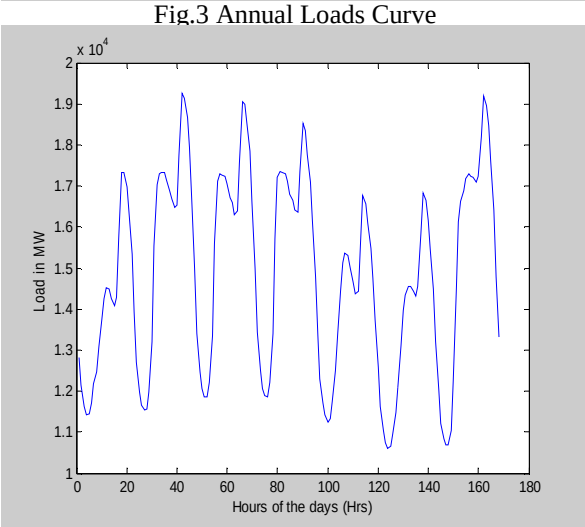

Fig.4 Weekly Loads Curve

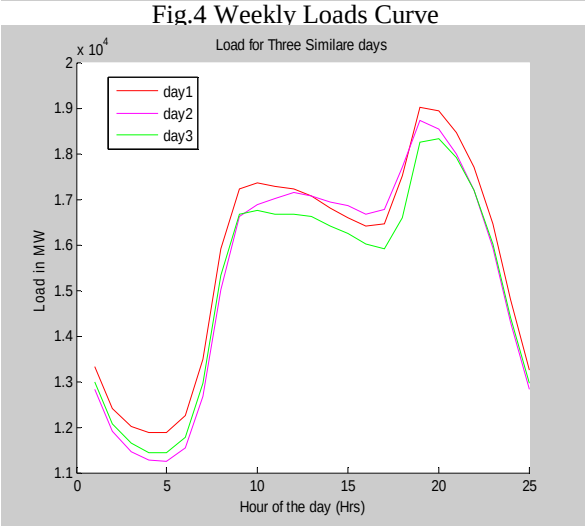

Fig.5 Three Similar Days Loads Curve

The architecture of the neural network used in this work is with 02 inputs, a hidden layer of 10 neurons and one output. The inputs are previous load and actual temperature, the output (forecast load). The training procedure is done by finding an optimal set of W in order to minimize the error between the forecast load and the actual load.

The ANN is trained using Matlab toolbox NSTOOL, the NE is trained using the proposed co-evolutionary algorithm. The SVM is trained by finding the support vector alpha; alpha_ which are calculated as a solution to the quadratic programming problem defined by equation (7) and (8).

We perform simulations for two cases.

Case 1: In the first case we compared three methods: Neuro Evolution, Support Vector Machines and Neural Network to training ours algorithms we use data from three similar days of the same month to predict the next two days so the inputs parameters are the loads and temperatures for each hour of the day and the load as target.

Figures 6 and 7 respectively show the target, forecasted load and also the percentage error in the forecasted load for the data calculated in the three similar days with Neuro Evolution Approach.

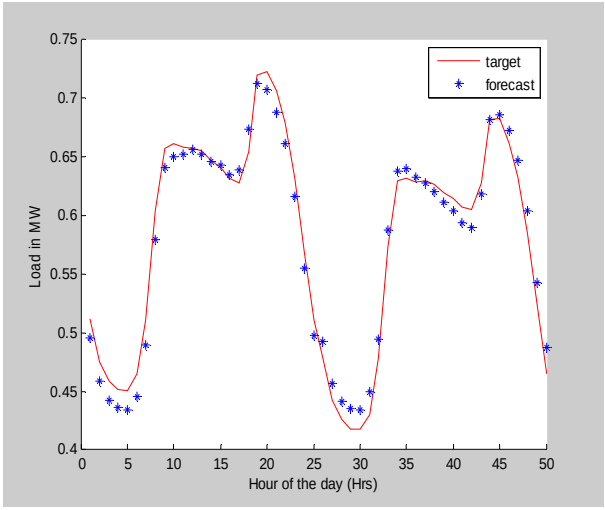

Fig.6 Forecast and Target with Neuro Evolution Approach

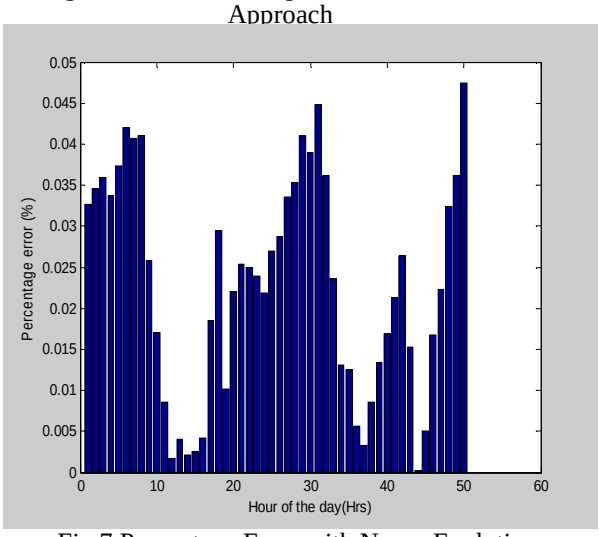

Fig.7 Percentage Error with Neuro Evolution Approach

The simulation based on Neuro Evolution gives the maximum percentage error 4.74% and the mean error 2.29%.

Figures 8 and 9 give the results obtained with Support Vector Machines (SVM) approach.

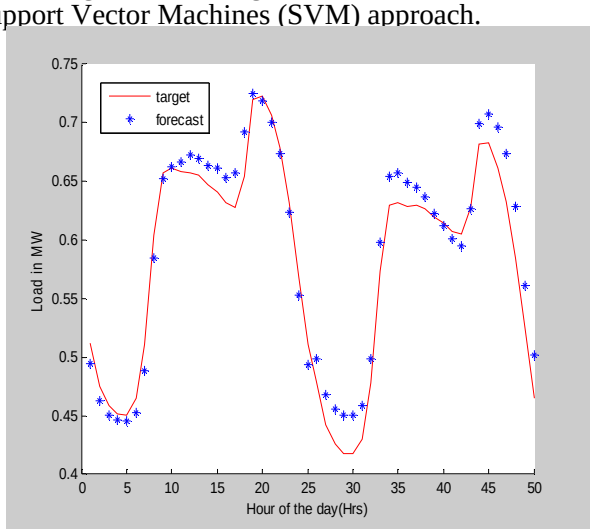

Fig.8 Forecast and Target with SVM Approach

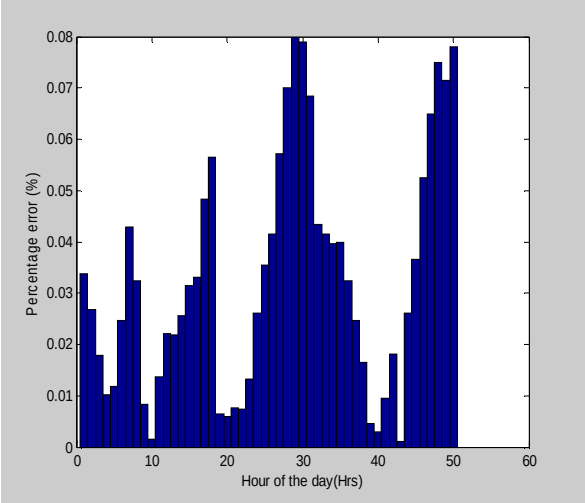

Fig.9 Percentage Error with SVM Approach

The simulation based on SVM gives the maximum percentage error 7.98% and the mean error 3.28%.

Figures 10 and 11 gives the results obtained with Neural Network approach using NSTOOL MTLAB Neural Network Tool Box.

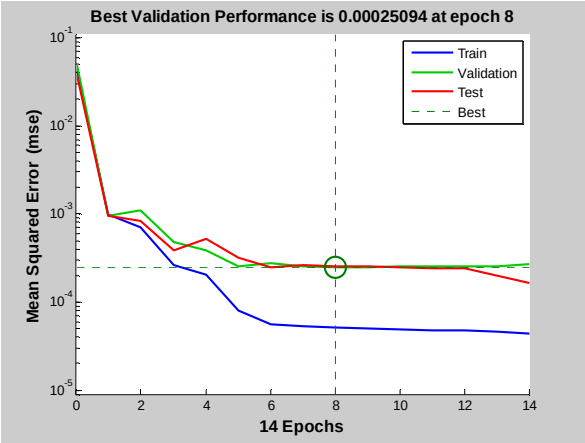

Fig. 10 Validation Performance of Neural Network

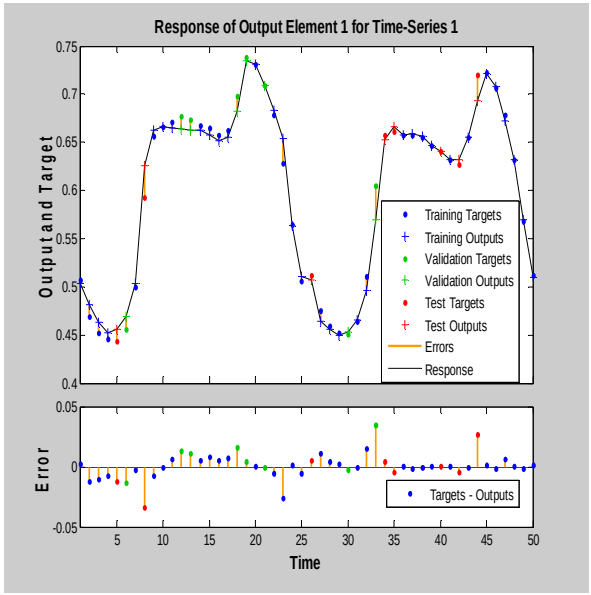

Fig.11 Forecast, Target and Percentage Error with Neural Network Approach

The forecasting results obtained from the proposed methods in terms of mean absolute percentage error and max absolute percentage error, we can conclude that the both approaches Neuro Evolution and Neural Network are better than Support Vector Machine, if we compare the results in this case between Neuro Evolution and Neural Network approach are very close, so we perform a second case concerned only NE and NN.

Case 2: In the second one we made a comparison between a Neuro Evolution and Neural Network methods so to training ours algorithms we use data from two weeks to predict the next week so the inputs parameters is temperatures for each hour of the two weeks and the load as target.

Figures 12 and 13 respectively show the target, forecasted load and also the percentage error in the forecasted load for the data calculated in the week with Neuro-evolution Approach, with 10 neurons in the hidden layer.

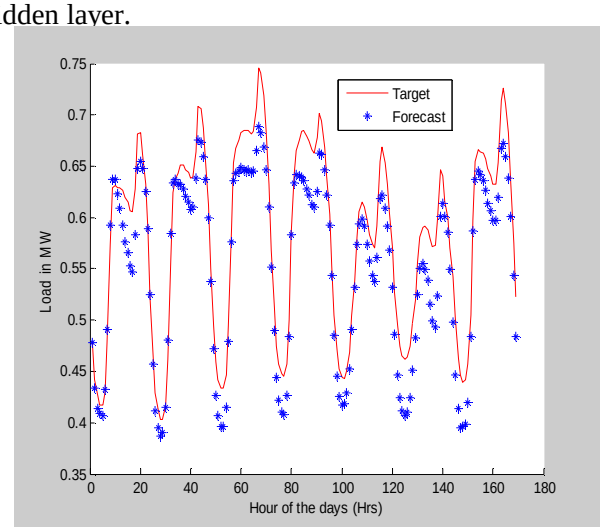

Fig.12 Forecast and Target with Neuro-evolution Approach

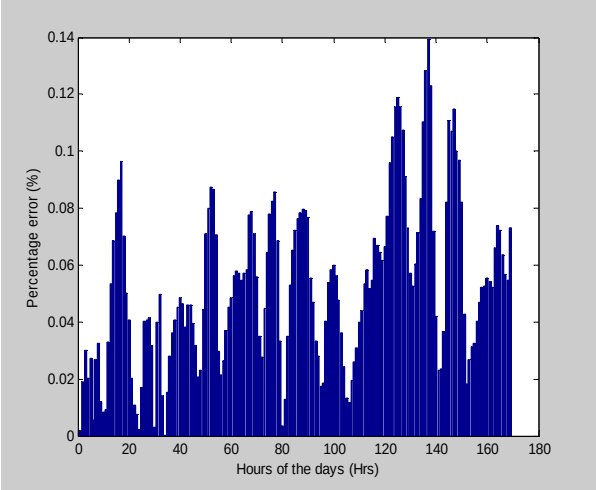

Fig.13 Percentage Error with Neuro Evolution Approach

Figures 14,15 and 16 shows the target, forecasted load and also the percentage error in the forecasted load for the data calculated in the week with Neural Network Approach, with 10,20 and 50 neurons in the hidden layer.

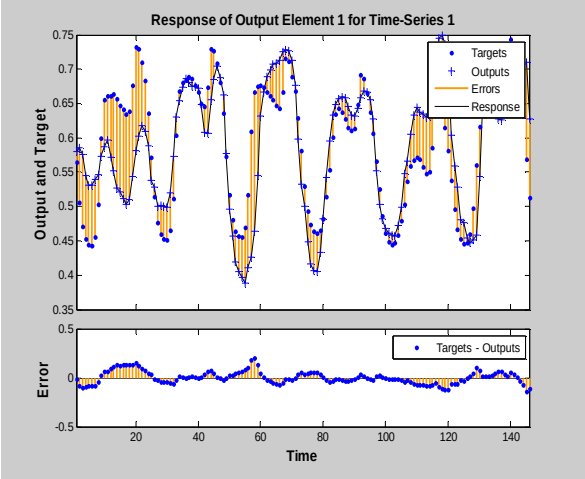

Fig.14 Forecast, Target and Percentage Error with Neural Network Approach for 10 neurons

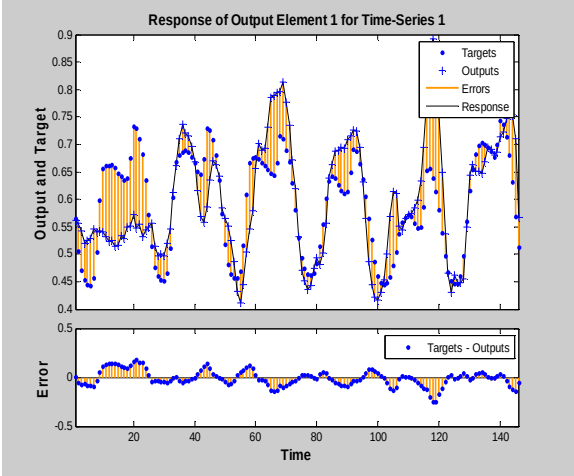

Fig.15 Forecast, Target and Percentage Error with Neural Network Approach for 20 neurons

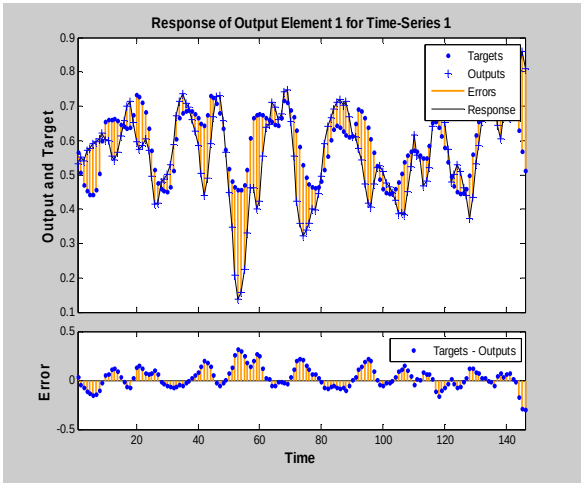

Fig.16 Forecast, Target and Percentage Error with Neural Network Approach for 50 neurons

5 Conclusions

This paper presents a neuro-evolution algorithm for short-term load forecasting problem. To test the effectiveness of the proposed neuro-evolutionary algorithm, we use the hourly data of the aggregate load and weather condition collected in 2002 from New England power system and its performance is compared to ANN and SVR. Using those data, two cases study are designed: in the first case, a day-ahead load forecasting is tested, the algorithms are trained with data of the three previous similar days to forecast the next day.

In this case, the neuro-evolution approach is found to be the most efficient with a mean error of 2:9% and a max error of 4:74%. In the second case, a week-ahead load forecasting is tested, the algorithms are trained with data of the two previous weeks to forecast the next week. By comparing the results obtained by the NN and the NE, we found that, again, the neuro-evolution approach is the most efficient with a mean error of 3:9% and a max error of 14%.

Performance of neuro-evolution, in this application, outperform neural network. this demonstrates that NE has better generalization than ANN. Indeed, changing forecast period from a day to a week does not affect, drastically, the performance of neuro-evolution (the same NE architecture is used in both cases). The loss of neural network performance suggest that the used training algorithm cannot find an optimal set of weights, which explains the decrease in performance when the number of neurons increased to 20 neurons instead of 10 neurons, figure (15).

This behavior is not surprising, knowing the optimization performance of co-evolutionary algorithms. Neuro-evolution captures the most interesting features of both neural network and co-evolutionary algorithms: learning and global optimization, which makes them suited for demanding applications.

References

[1] A. Ladjici, A. Tiguercha, M. Boudour, *Nash equilibrium in a two settlement electricity market using competitive coevolutionary algorithms*, International Journal of Electrical Power and Energy Systems, vol. 57, no. 0,pp. 148 – 155, 2014.

[2] A. Tiguercha, A. Ladjici, M. Boudour, *Suppliers' optimal biding strategies in day-ahead electricity market using competitive coevolutionary algorithms*, 3rd International Conference in Systems and Control (ICSC), 2013 on, pp. 821–826, Oct 2013.

[3] Z. H. Osman, M. L. Awad, T. K. Mahmoud, *Neural network based approach for short-term load forecasting*, in Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES, pp. 1–8, IEEE, 2009.

[4] R.G. Bertrand, A. J. Conejo, *Electricity market equilibrium model with constraints involving prices*, in International Conference on Mathematical and Statistical Modeling in Honor of Enrique Castillo (ICMSM), Ciudad Real (Spain), pp. 28–30, 2006.

[5] P. R. Campbell, K. Adamson, *Methodologies for load forecasting*, in Intelligent Systems, 2006 3rd International IEEE Conference on, pp. 800– 806, IEEE, 2006.

[6] M. Cho, J. Hwang, C.S. Chen, *Customer short term load forecasting by using arima transfer function model*, in Energy Management and Power Delivery, 1995. Proceedings of EMPD '95. 1995 International Conference on, vol. 1, pp. 317–322 vol.1, Nov 1995.

[7] M. M. Ismail, M. Hassan, *Artificial neural network based approach compared with stochastic modelling for electrical load forecasting*, in Modelling, Identification & Control (ICMIC), 2013 Proceedings of International Conference on, pp. 112–118, IEEE, 2013.

[8] A. Escobar, L. Perez, *Application of support vector machines and anfis to the short-term load forecasting*, in Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES, pp. 1–5, IEEE, 2008.

[9] S. Sachdeva, C. M. Verma, *Load forecasting using fuzzy methods*, in Power System Technology and IEEE Power India Conference, 2008. POWERCON 2008. Joint International Conference on, pp. 1–4, IEEE, 2008.

[10] W. Sun, *A novel hybrid GA based SVM short term load forecasting model*, in Knowledge Acquisition and Modeling, 2009. KAM'09. Second International Symposium on, vol. 2, pp. 227–229, IEEE, 2009.

[11] S. S. Haykin, *Neural networks and learning machines*, vol. 3. Pearson Education Upper Saddle River, 2009.

[12] P. Bunnoon, *Mid-term load forecasting based on neural network algorithm: Comparison of models*, International Journal of Computer and Electrical Engineering, vol. 3, no. 4, pp. 600–605, 2011.

[13] N. Cristianini, J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[14] I. Steinwart, A. Christmann, *Support vector machine*, Springer, 2008.

[15] D.f. ZHAO, M. Wang, J.s. ZHANG, X.f. WANG, *A support vector machine approach for short term load forecasting* , Proceedings of the Csee, vol. 4, p. 004, 2002.

[16] X. Pan, B. Lee, *A comparison of support vector machines and artificial neural networks for mid-term load forecasting*, in Industrial Technology (ICIT), 2012 IEEE International Conference on, pp. 95–101, IEEE, 2012.

[17] A. Reyaz, Y.Q. Zhang, R. W. Harrison, *Granular decision tree and evolutionary neural SVM for protein secondary structure prediction*, International Journal of Computational Intelligence Systems, vol. 2, no. 4, pp. 343–352, 2009.

[18] D. E. Moriarty, R. Mikkulainen, *Efficient reinforcement learning through symbiotic evolution*, Machine learning, vol. 22, no. 1-3, pp. 11–32, 1996.

[19] D. Polani, R. Miikkulainen, *Fast reinforcement learning through eugenic neuro-evolution*, University of Texas at Austin, Austin, TX, 1999.

[20] K. Allen, P. Ballen, *Evolving grounded communication in a navigation task using neat*, 2012.

[21] A. G. Pereira, A. Petry, *Data assimilation using neuro-evolution of augmenting topologies*, in Neural Networks (IJCNN), The 2012 International Joint Conference on, pp. 1–6, IEEE, 2012.

[22] K. O. Stanley, D. B. D'Ambrosio, J. Gauci, *A hypercube-based encoding for evolving large-scale neural networks*, Artificial life, vol. 15, no. 2, pp. 185–212, 2009.

[23] A. Karsaz, H. R. Mashhadi, R. Eshraghnia, *Cooperative co-evolutionary approach to electricity load and price forecasting in deregulated electricity markets*, in Power India Conference, 2006 IEEE, pp. 6–pp, IEEE, 2006.

[24] R. P. Wiegand, *An analysis of cooperative coevolutionary algorithms*, PhD thesis, Citeseer, 2003.